Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Document**
  Eindverslag
- **Auteur**
  Mark van Houten (20015081)
- **Datum**
  26 juli 2004
- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag
- **Opleidingsvariant**
  MBIT
- **Periode**
  November 2003 – april 2004
- **Opdrachtgever**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft
- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Referaat

Auteur:
Mark van Houten;
De auteur is een afstuderende student Informatica aan de Haagse Hogeschool, afstudeerrichting MBIT.

Titel rapport:
Ontwikkeling webbased beveiligingssysteem bij de SMRA

Kenmerk rapport:
Dit rapport geeft een beschrijving van de totstandkoming en uitvoering van een afstudeerproject bij de SMRA in Delft. Het rapport heeft als doel de examinatoren en de gecommitteerde in staat te stellen het project te beoordelen.

Datum van publicatie:
April 2004

Trefwoorden:
- Haagse Hogeschool;
- Afstuderen;
- ASP.NET;
- IAD;
- UML;
- Microsoft.NET;
- Object georiënteerd;
- Beveiliging;
- Visual Basic.NET.
- .NET Framework

Voorwoord

Studenten op de Haagse Hogeschool van de afdeling informatica werken in de laatste periode aan een afstudeeropdracht. Deze afstudeeropdracht geeft aan of de student de tijdens de opleiding opgedane kennis kan toepassen in een project bij een bedrijf.

De student die aan deze opdracht heeft gewerkt is Mark van Houten en volgt de afstudeerrichting MBIT (Management en Beheer van Informatie Technologie) op de Haagse Hogeschool.

Dit procesverslag beschrijft de verrichte werkzaamheden van het project “Ontwikkeling webbased beveiligingssysteem“ dat is uitgevoerd bij de SMRA. Aan dit project is gewerkt in de periode van 18 november 2003 tot 13 april 2004.

Door de realisatie van dit document worden de examinatoren in staat gesteld het project te beoordelen. Uit dit document moet de omvang, het niveau en de diepgang van het project duidelijk worden.

Dit document is opgesteld door Mark van Houten.

Den Haag, Haagse Hogeschool, 12 april 2004

Inhoudsopgave

1. Inleiding

Afstuderen is de laatste stap voor een student van de Haagse Hogeschool. Hierbij staat het analyseren en oplossen van een informaticaprobleem en het presenteren van de resultaten daarvan, zowel schriftelijk als mondeling, centraal. Dit document geeft het verloop van het afstudeerproject weer en stelt de examinatoren en de gecommitteerde in staat dit te beoordelen.

Het document is opgebouwd uit drie delen. In het eerste deel is er een omschrijving gegeven van de organisatie waarbinnen dit project heeft plaatsgevonden. Ook wordt de probleemstelling, het doel van de opdracht en de opdrachtomschrijving beschreven. Door middel van een beschrijving van het plan van aanpak is het verwachte verloop van het project aangegeven.

In het tweede deel wordt beschreven hoe de verschillende producten tot stand zijn gekomen. Hierbij wordt vooral gekeken naar “hoe” het proces is verlopen en “waarom” het proces op die manier is aangepakt. Ook worden de belangrijkste beslissingen die zijn genomen beschreven en verantwoord.

In het laatste deel worden opgeleverde de producten geëvalueerd. Tijdens de productevaluatie is er per product gekeken naar de aanpak ten opzichte van het resultaat en worden de producten op kwaliteit geëvalueerd. Ook is er tijdens de productevaluatie gekeken naar de keuzen die gemaakt zijn tijdens procesevaluatie en wordt er bekeken of de eerder gemaakte keuzen juist zijn geweest.

2. Organisatie

2.1 Inleiding

In dit hoofdstuk is de organisatie beschreven waar de afstudeeropdracht is uitgevoerd. Als eerste is de organisatorische inrichting beschreven met een globale omschrijving van de diensten die de SMRA levert. Als tweede is de plek binnen de organisatie omschreven waar de student werkzaam is geweest.

2.2 Organisatorische inrichting

De afstudeeropdracht is uitgevoerd bij de SMRA. De SMRA is een orgaan van de Nederlandse Hervormde Kerk en de afkorting staat voor Stichting Mechanisatie, Registratie en Administratie. De centrale organisatie bestaat sinds 1967 en bestaat uit circa 60 medewerkers.

De kernactiviteit van de SMRA is beheren van een centraal ledenbestand. In dit centrale leden bestand worden kerkelijke gegevens opgeslagen. In dit bestand is op dit moment de kerkledenadministratie van ongeveer 1400 hervormde gemeenten bijgehouden en er worden inmiddels circa zeven miljoen personen beheerd. De SMRA heeft na de belastingsdienst, de grootste database met persoonsgegevens van Nederland.

Het ledenbestand wordt onderhouden vanuit twee belangrijke bronnen: de overheid en de kerkelijke gemeenten. Het onderhoud wordt gedaan door middel van meerdere systemen die gegevens bijhouden op een centrale locatie. Deze gegevens worden bijgehouden voor de Stichting Interkerkelijke Ledenadministratie (SILA) en de bijbehorende administratie wordt de koppeladministratie genoemd. Van elke persoon worden de burgerlijke gegevens (waaronder NAW) opgeslagen samen met een koppeling aan een lokale kerk van een kerkgenootschap.

De burgerlijke gegevens worden aangeleverd door de overheid op grond van een “SILA-stip”. Een SILA-stip geeft in de administratie van de overheid aan dat indien er burgerlijke gegevens van een persoon met deze stip worden gewijzigd, deze wijzigingen moeten worden doorgegeven aan de SILA.

Voor 2,6 miljoen van deze personen wordt er een kerkledenregistratie bijgehouden. Voor deze personen worden de burgerlijke gegevens opgeslagen samen met een aanvullende set kerkelijke gegevens.

Vanuit de centrale registratie kunnen alle aangesloten kerkelijke gemeenten in verschillende vormen worden geïnformeerd over mutaties in de lokale ledenregistratie. Dit wordt naar keuze gedaan d.m.v. formulieren, registratiekaarten of geautomatiseerde mutatie-doorgifte over een modemverbinding naar een lokale PC-toepassing voor ledenregistratie. De lokale kerken leveren ook mutaties aan de SMRA door middel van formulieren of de modemverbinding. Daarnaast levert de SMRA een groot aantal diensten op basis van de ledenadministratie zoals bijvoorbeeld fondswerving (kerkbalans) en distributie van kerkbladen.

Adresgegevens SMRA:

Bezoekadres	: Motorenweg 23
Postcode	: 2623 CR
Plaats	: Delft
Telefoon (receptie)	: 015-2619203
E-Mail	: info@smra.nl
Postbus	: 250, 2600 AG, Delft

2.3 Plek binnen de organisatie

De afstuderende student is werkzaam op de afdeling software ontwikkeling. Ook de projectleiders K.Schaaf en R. Mehlkopf zijn werkzaam op deze softwareafdeling.

De afdeling bestaat uit 5 vaste werknemers met daarnaast vaak extern ingehuurde krachten. Op deze afdeling wordt er voornamelijk gewerkt aan het de ontwikkeling van Baruch, programmatuur op het mainframe en andere programmatuur voor het beheren van het centraal ledenbestand. Baruch is een applicatie die kerkelijke gemeenten gebruiken voor het verwerken van de kerkelijke gegevens. Sinds kort is de ontwikkelaar van Baruch gestopt en heeft de SMRA de ontwikkeling overgenomen.

Op de software afdeling wordt er voornamelijk ontwikkeld in de programmeertalen Natural en Visual Basic 6. Binnen de softwareontwikkelafdeling wordt er standaard geen gebruik gemaakt van software ontwikkelmethoden of technieken.

3. Omschrijving van de opdracht

3.1 Inleiding

In de periode voor het afstuderen is er een opdrachtomschrijving opgesteld. Deze opdrachtomschrijving is tijdens het uitvoeren van de afstudeeropdracht gewijzigd. De gewijzigde opdracht heeft zich gevormd tot een definitieve opdrachtomschrijving van de afstudeeropdracht welke is uitgereikt aan de examinatoren en de gecommitteerde. In dit hoofdstuk is de definitieve opdrachtomschrijving beschreven en is deze nader toegelicht.

3.2 Probleemstelling

Op dit moment worden de persoongegevens doorgevoerd in het centrale systeem door administratieve medewerkers van de SMRA. Voor het doorvoeren van deze wijzigingen wordt er gebruik gemaakt van verschillende systemen. De wijzigingen die moeten worden doorgevoerd worden geleverd door functionarissen van lokale gemeenten, met name ledenadministrateurs. Alleen zij bezitten de bevoegdheid voor het indienen van mutaties.

Het wijzigen van de gegevens wordt gedaan door middel van formulieren of geautomatiseerde mutatie-doorgifte over een modemverbinding vanuit de lokale PC-applicaties die gedeeltelijk geautomatiseerd verwerkt worden.
De kleine kerkelijke gemeenten werken met formulieren. Deze formulieren zijn bewerkelijk en dienen gecontroleerd te worden door de medewerkers van de SMRA. Deze formulieren worden verzonden per post en het duurt één tot enkele dagen voordat de SMRA de mutaties ontvangt. Elke wijziging die door middel van formulieren wordt toegezonden moet door een SMRA medewerker worden verwerkt in het centrale systeem.
De grotere kerkelijke gemeenten werken met PC-applicaties. Het uitvoeren van deze PC-applicaties heeft organisatorische consequenties. Het vereist een lokale gebruikersomgeving met de nodige hardware, software en kennis over de PC-applicatie. Hiernaast zijn er ook vrijwilligers nodig voor de verspreiding de van materialen zoals acceptgiro's.

Voor de kleinere kerkelijke gemeenten is het gebruik van een PC-applicatie geen goed alternatief. Dit vanwege de organisatorische gevolgen voor de organisatie. De SMRA wil een alternatief bieden dat ook geschikt is voor kleine kerkelijke gemeenten.

De SMRA heeft behoefte aan een versoepelde en versnelde manier voor het leveren van diensten. Dit wil men bereiken door de diensten aan te bieden over het Internet. Zo kan de SMRA ook aan de SMRA de kleine kerkelijke gemeenten de mogelijkheid bieden om zelf hun diensten uit te kunnen voeren.

Een ander probleem is dat de SMRA niet beschikt over de gegevens van de gemeentelijke functionarissen. Om contact te zoeken met een functionaris die de ledenadministratie uitvoert binnen een kerkelijke gemeente moet de SMRA op dit moment eerst contact zoeken met de gemeentelijke beheerder.
Ook kan een functionaris tijdens telefonisch contact ook moeilijk of niet worden gevalideerd door een SMRA medewerker wegens gebrek aan informatie over de functionarissen. Dit is wederom wenselijk omdat er wordt gewerkt met persoonlijke gegevens.

3.4 Opdrachtomschrijving

Het is de bedoeling dat de diensten die door de SMRA worden aangeboden aan kerkelijke gemeenten in de toekomst met behulp van Internet worden aangeboden. Om de diensten over het Internet te kunnen aanbieden is er behoefte aan een universeel toepasbaar beveiligingssysteem dat het mogelijk maakt verschillende diensten op een veilige manier beschikbaar te stellen via het Internet. Omdat veel van deze diensten persoonlijke gegevens, waaronder die over de religieuze achtergrond, ontsluiten is het van het uiterste belang dat deze gegevens op een veilige manier kunnen worden ingezien en gemuteerd door alleen bevoegde personen.

Het is de bedoeling dat diensten maar ook onderdelen van diensten kunnen worden aangeboden via het Internet. Om deze reden dient een dienst te worden onderverdeeld in functies. Het beveiligingssysteem definieert en beheert de te ontsluiten diensten in termen van deelcomponenten (functies, URL's) waardoor deze afgeschermd kunnen worden voor niet bevoegde personen.

Alleen de SMRA beheerders hebben de mogelijkheid om diensten of functies aan te maken, te wijzigen en te verwijderen. Binnen dit systeem kunnen de beheerders van de SMRA functies beschikbaar stellen, terugnemen, blokkeren en deblokkeren voor bepaalde kerkelijke gemeenten. Een beheerder in deze gemeente kan deze diensten of functies dan beschikbaar stellen voor een onderliggende functionaris die deze functie dan kan gebruiken. Ook dienen de functies te kunnen worden geblokkeerd op meerdere aggregatie niveau's.

Het systeem dient het mogelijk te maken een deel van het beheer uit te besteden aan gemeentelijke beheerders. Hiervoor is een gelaagd model nodig. Dit omdat de lokale kerken beter op de hoogte zijn van de bevoegde personen dan de SMRA. Ook worden er vele mutaties verwacht op de gegevens van de bevoegde personen. Hiermee wordt bedoeld dat de diensten van functionarissen vaak zullen wisselen. Dit is het gemakkelijkst te beheren als het beheer van functionarissen op het niveau van de kerkelijke gemeente plaatsvindt. Zo wordt de SMRA beheerder in staat gesteld om de gemeentelijke beheerders te beheren en de gemeentelijke beheerders worden in staat gesteld om de onderliggende functionarissen te beheren.
Het gelaagde model geeft nog een aantal mogelijkheden. Als functionarissen contact op willen nemen met hun beheerder dienen zij eerst contact op te nemen met hun gemeentelijke beheerder. Mochten functionarissen alsnog contact opnemen met de SMRA dan dient het systeem de mogelijkheid te bieden om functionarissen te valideren.

Het gelaagde beheersmodel en de nodige intelligentie voor het beheer van bevoegde personen en functionaliteiten maakt dit systeem specifiek voor de SMRA.

Het beveiligingssysteem beveiligt zichzelf. De beheerfuncties van het beveiligingssysteem worden voor zowel de interne als externe gebruikers eveneens door hetzelfde beveiligingssysteem beveiligd.

Het beveiligingssysteem definieert en beheert de te ontsluiten diensten in termen van deelcomponenten (diensten, functies, URL's) waardoor deze afgeschermd kunnen worden voor niet bevoegde personen.

Het nieuwe systeem zal gebruik maken van Microsoft SQL 2000 server en ASP.NET. Er zal een onderzoek moeten worden verricht naar ASP.NET. In dit onderzoek zal het concept van Microsoft.NET worden onderzocht en wordt er onderzocht naar de beveiligingsmethodieken binnen ASP.NET.

Het beveiligingssysteem dient te worden ontworpen, gebouwd en op systeemniveau getest. De onderdelen die niet bij het project worden uitgevoerd zijn hieronder expliciet gegeven.

- Het invoeren van de applicatie is geen onderdeel van de opdracht;
- Van de applicatie worden geen handleidingen opgeleverd;
- Van de applicatie worden geen vormen van opleidingsmateriaal opgeleverd.

3.5 Op te leveren producten

Tijdens het project zullen de volgende producten worden opgeleverd:

- Document: Plan van aanpak;
- Document: Onderzoek .NET;
- Document: Definitiestudie;
- Document: Pilotrapport per pilotdeel;
- Een beveiligingssysteem dat de hierboven genoemde functionaliteiten bevat.

3.6 Uitgangssituatie

Bij de SMRA is er een werkplek ingericht op de softwaretest afdeling met een netwerkaansluiting en de mogelijkheid om te internetten. Voor het uitvoeren van de opdracht wordt de laptop van de student gebruikt.

Het beveiligingssysteem is een nieuw systeem dat nog volledig moet worden ontwikkeld. Bij de start van het project is er een document gerealiseerd met daarin een beschrijving van de opdracht. De specifieke eisen en wensen van het te ontwikkelen systeem waren hierin nog niet duidelijk opgegeven.

De software die benodigd is voor het uitvoeren van de opdracht is vastgesteld en is beschikbaar gesteld door de SMRA. De volgende software is beschikbaar gesteld door de SMRA:

- Visual studio.Net 2003
- Microsoft SQL 2000 server (of andere MS-SQL variant)
- IIS 5.1 (webservice)
- MS-Office pakket
- Windows XP professional
- Windows 2000 server

Het personeel binnen de SMRA heeft nog vrijwel geen ervaring met de Microsoft.NET ontwikkelomgeving.

4. Plan van aanpak

4.1 Inleiding

Door middel van het plan van aanpak is getracht een overeenstemming te bereiken tussen de uitvoerende student en de opdrachtgevers over het te volgen traject, om tot het gewenste eindresultaat te komen. Door overeenstemming te bereiken wordt de kans verkleind dat de opdrachtgever en de uitvoerende student een verschillend beeld hebben van de uit te voeren opdracht. Door dit uit te sluiten wordt voorkomen dat het eindproduct niet aan de verwachtingen voldoet. Ook zijn de afspraken met de opdrachtgevers en de afstuderende student over de uitvoering van het project hierin vastgelegd. Verder wordt het plan van aanpak gebruikt voor het beheerst uitvoeren van de activiteiten.

Als eerste is de projectorganisatie beschreven. Daarna is de totstandkoming van elk onderdeel uit het plan van aanpak beschreven. Het onderdeel opdrachtomschrijving zal niet worden beschreven, omdat deze apart is behandeld in het vorige hoofdstuk. De volledige versie van het plan van aanpak is bijgevoegd als bijlage “A”. Het plan van aanpak heeft de eerste week van de afstudeerperiode in beslag genomen.

4.2 Projectorganisatie

Om de aanspreekpunten te kunnen bepalen en de verantwoordelijkheden van het project te verdelen, zijn de functies vastgesteld. Hieronder is een overzicht gegeven van de personen met de daarbij behorende functies die betrekking hebben op dit project.

Persoon	**Functie, omschrijving**
Jelke Brinksma	Hoofd afdeling softwareontwikkeling, contractpersoon voor personele zaken.
Roel Mehlkopf	Software ontwikkelaar, begeleider van het project en is de persoon voor technische ondersteuning.
Kees Schaaf	Software ontwikkelaar, begeleider van het project en is de persoon voor de functionele aspecten van het project.
Mark van Houten	Afstudeerder op de afdeling software ontwikkeling, de uitvoerende persoon van het project.

4.3 Risico´s en maatregelen

Tijdens het uitvoeren van een project bestaat er altijd de kans dat er zich bepaalde gebeurtenissen voordoen die invloed hebben op de uitvoering van het project. Door rekening te houden met risico’s die zich kunnen voordoen kunnen deze tot een minimum worden beperkt. Er zijn risico’s die bij elk project kunnen optreden. De risico’s die zijn vastgesteld tijdens voorgaande projecten zijn, indien van toepassing, ook bij dit project gegeven. Omdat er met nieuwe methoden, technieken en een nieuwe ontwikkelomgeving gewerkt zal gaan worden is hiernaar eerst enig onderzoek gedaan om zo de risico’s te kunnen bepalen.

Bij elk risico is een maatregel gegeven om het risico te voorkomen of als een risico zich voordoet, de gevolgen tot een minimum te beperken. Hieronder zijn twee voorbeelden gegeven van projectrisico’s en de gestelde maatregelen.

- *Mogelijke tijdsvertraging door ziekte*
  Maatregel: Bij de planning wordt er rekening gehouden met 1 week overhead. Indien er sprake mocht zijn van ziekte kan moet worden bekeken of de overhead als reeds is gebruikt. Mocht dit het geval zijn dan zal de opdracht moeten worden ingekort in overleg met de opdrachtgever(s). Mocht de ziekte langer aanhouden dan 1 week zal ook, in overleg de opdrachtgever(s), de opdracht moeten worden ingekort.

- *Omdat er voor het eerst een iteratieve manier van ontwikkelen wordt gebruikt, kunnen zaken zoals incomplete pilots, workshops, hergebruik en, prioriteiten een verkeerd beeld opwekken.*
  Maatregel: IAD bestuderen voordat er een start wordt gemaakt met het ontwikkelen.

4.4 Methoden en technieken

Om de kwaliteit van het product te waarborgen, is gebruik gemaakt van een systeemontwikkelmethode. Door te werken volgens een methode wordt het project gestructureerd en zal het opgeleverde product moeten voldoen aan de gestelde eisen van de ontwikkelmethode. Aan de hand van de structuur van de methode kan een planning van de uit te voeren activiteiten worden gemaakt, waardoor het risico op overschrijden van de deadline wordt verkleind.

4.4.1 Ontwikkelmethode

Er is voor de ontwikkelmethode IAD gekozen vanwege de iteratieve ontwikkelingsstrategie. Tijdens de module SW-07 is deze ontwikkelmethode besproken en deze methode leek mij een geschikte methode voor dit project. Om deze reden is er enig onderzoek gedaan naar de ontwikkelmethode en is besloten deze methode te gebruiken voor het project.

De IAD ontwikkelmethode heeft een aantal voordelen:
- Bij het project kan het iteratieve gedrag ervoor zorgen dat de complexiteit wordt verminderd;
- Er worden al snel resultaten bereikt die gebruikt kunnen worden voor feedback;
- Aan het begin van elke iteratie is het mogelijk om de doelen, eisen en oplossingen te evalueren of te herzien;
- Door het te ontwikkelen systeem onder te verdelen in kleine delen die elk apart kunnen worden ontwikkeld, ontstaat een overzichtelijke manier van werken.
- De kans dat risico's zich voordoen wordt verkleind omdat er een frequente toetsing plaats vindt en door de meest risicovolle delen in de eerste iteraties op te nemen.

Binnen IAD wordt er volgens een iteratie strategie gewerkt. De iteratiestrategie die voor dit project is gekozen is de strategie: big-bang-invoeren. Er is voor deze strategie gekozen omdat:
- De opleverdata vast staat, en snelle oplevering is daardoor niet nodig;
- Het invoeren van de applicatie is geen onderdeel van de opdracht, maar deze strategie stelt de SMRA in staat na het project het systeem alsnog in te voeren;
- Het apart invoeren van pilots zal waarschijnlijk geen baad bieden voor de ontwikkeling of de organisatie. Dit omdat de losse pilots geen bruikbare delen van het systeem zullen opleveren (voor de organisatie);

Big-bang-invoeren is een iteratiestrategie waarin de eerste 2 fasen (definitiestudie en pilotontwikkeling) iteratief worden uitgevoerd, maar de laatste fase (invoering) niet. De fase invoering vindt pas plaats als de laatste, definitieve pilot is ontwikkeld.

Figuur 1

Omdat het invoeren van het systeem geen onderdeel is van het project wordt deze stap niet uitgevoerd. Doordat er is gekozen voor de iteratiestrategie big-bang-invoeren moet het systeem wel volledig gerealiseerd worden en zo wordt het bedrijf in staat gesteld alsnog het systeem in te voeren in de organisatie.

4.4.2 Technieken

UML

Tijdens de opleiding zijn er UML technieken en technieken van Yourdon aangeleerd. Voor dit project wordt er gebruik gemaakt van UML. Dit omdat UML een techniek is die is gericht op objectgeoriënteerde programmeeromgevingen en ASP.NET een objectgeoriënteerde programmeertaal is.

ERD

Omdat er voor het eerst gebruik wordt gemaakt van UML is onderzocht welke UML tools er beschikbaar waren voor het opzetten van een relationele database. Hiervoor is ook gebuikt gemaakt van het boek: *Warmer, J., en Kleppe A., Praktisch UML, Addison Wesley, 2e editie, 2001.* De volgende conclusies kunnen worden getrokken uit de mededelingen die worden gedaan in het boek:

- De SQL 2000 server die wordt gebruikt bij dit project beschikt niet over een objectgeoriënteerde schil om de database te benaderen. Het nadeel hiervan is dat er een conversie nodig is van een objectgeoriënteerd model naar een model van een andere opslagstructuur.
- Ook past een object model niet op een relationeel tabellenmodel, de structuur is geheel anders.

Mede om deze redenen en omdat de student nog weinig ervaring had met UML is er gekozen om gebruik te maken van de aangeleerde ER modellering techniek. Deze techniek geeft een conceptueel gegevensmodel en wordt algemeen gebruikt voor het structureren en modelleren van de gegevens.
De techniek ER modellering geeft een conceptueel gegevensmodel en wordt algemeen gebruikt voor het structureren en modelleren van de gegevens. Een ERD geeft een grafisch model dat de lay-out beschrijft van de opgeslagen gegevens op een hoog abstractieniveau. Aan de hand van een ERD is er gemakkelijk een implementatiemodel te ontwikkelen waarmee de database kan worden opgezet.

4.5 Standaards en richtlijnen

Binnen het bedrijf worden er geen standaarden en richtlijnen gebruikt voor het rapporteren, ontwerpen en ontwikkelen. Om de integriteit van de documentatie te bewaren, zijn er alsnog standaarden en richtlijnen opgesteld.

Rapportagerichtlijnen

De rapportagetechnieken die zijn aangeleerd tijdens de module AV-03 (rapportagetechniek) zijn gebruikt om te kunnen voldoen aan de vereiste documentopmaak. Door gebruik te maken van de checklist die als bijlage is opgenomen in het plan van aanpak, voldoet de documentatie aan de vereiste opmaak.

Ontwerp- en ontwikkelrichtlijnen

Tijdens het ontwerpen en ontwikkelen van de applicatie is rekening gehouden met enkele eisen die Ghezzi stelt in het boek “The Fundamentals of Software Engineering” (ISBN 0-13-305699-6). Bij dit project zijn de onderstaande eisen nagestreefd:

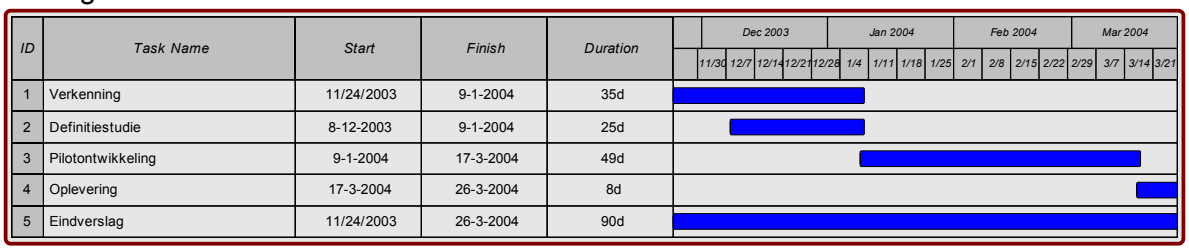
- Correctheid;
- Robuustheid;
- Gebruikersvriendelijkheid;
- Performance.

In het boek van Ghezzi is per eis beschreven hoe er aan deze eis kan worden voldaan. Deze beschrijvingen zijn gebruikt bij het ontwerpen en ontwikkelen van de software.

Om de programmeercode duidelijk en leesbaar te laten zijn voor derden, is een standaard opbouw van de programmeercode gebruikt. Tijdens de module SW-06 zijn richtlijnen hiervoor gegeven, die gelden als standaard opbouw binnen de Haagse Hogeschool.

4.6 Planning

Hieronder is een schema weergegeven met hierop aangegeven wanneer er aan welke fase wordt gewerkt.



ID	Task Name	Start	Finish	Duration
1	Verkenning	11/24/2003	9-1-2004	35d
2	Definitiestudie	8-12-2003	9-1-2004	25d
3	Pilotontwikkeling	9-1-2004	17-3-2004	49d
4	Oplevering	17-3-2004	26-3-2004	8d
5	Eindverslag	11/24/2003	26-3-2004	90d

Figuur 2

Ook hier geld dat de fase definitiestudie en de fase pilotontwikkeling meerdere malen zullen worden doorlopen. Op 9 januari wordt verwacht dat de verkenning is afgerond en dat de eerste volledige versie van de definitiestudie is gerealiseerd.

In de definitiefase zal er een verfijnde versie van het plan van aanpak worden gerealiseerd voor de definitiestudie. Ook wordt er tijdens de definitiestudie een pilot plan opgesteld. Hierin wordt de planning gegeven voor elke pilot.

Tijdens de verkenningsfase zal ook de definitiefase plaatsvinden. Dit omdat deze fasen nauw samenhangen. Ook vinden er in deze periode feestdagen plaats. Om deze reden er is rekening gehouden met een uitloop van 5 werkdagen.

Deze planning geeft de activiteiten weer tegen over de tijd en geeft aan op welk tijdstip er een product opgeleverd moet worden.

5. Onderzoek Microsoft.NET

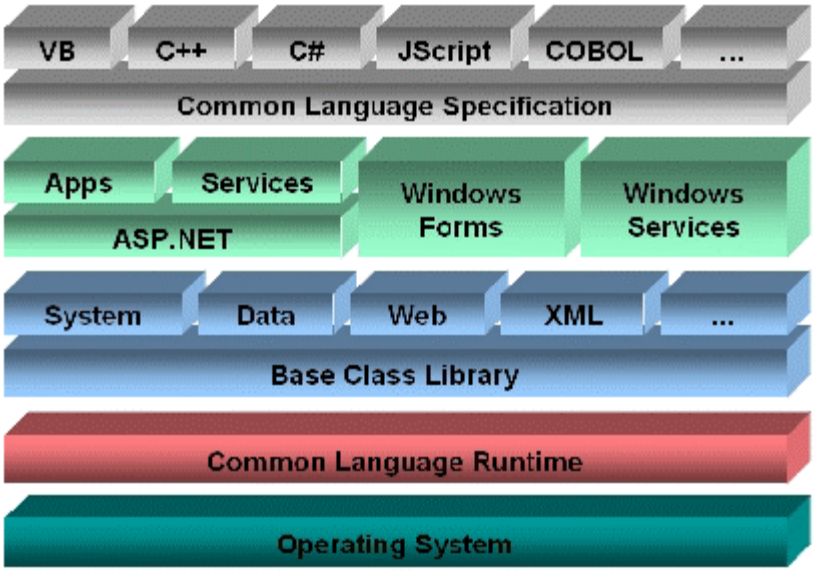
5.1 Inleiding

Omdat zowel de student als de medewerkers binnen de SMRA geen kennis bezitten over de .NET omgeving is er besloten een onderzoek uit te voeren naar de .NET omgeving. Het doel van dit onderzoek is om een goed beeld te verkrijgen over de structuur en de mogelijkheden van de .NET ontwikkelomgeving voor zowel de opdrachtgevers als de uitvoerende student. Op basis hiervan worden de opdrachtgevers in staat gesteld een oordeel vormen over de .NET omgeving en kan het aantrekkelijk zijn om dit in de toekomst meer te gaan gebruiken.

Tijdens het onderzoek is er voornamelijk gebruik gemaakt van diverse websites en boeken. Het onderzoek heeft uiteindelijk een evaluatie rapport opgeleverd. Deze is bijgevoegd bij dit document als bijlage “B”. Het onderzoek naar de Microsoft.NET omgeving heeft ongeveer 3 weken in beslag genomen.

5.2 .NET Framework.

Als eerste onderdeel van het onderzoek is er een globale omschrijving gemaakt over de .NET omgeving. Dit onderdeel heeft ongeveer twee dagen onderzoek gekost. De twee belangrijkste onderdelen van de .NET omgeving zijn de Common Language Runtime (CLR) en de base class library. De common language runtime zorgt ervoor dat er verschillende programmeertalen kunnen worden gebruikt binnen het .NET framework. Omdat de programmeertaal Visual Basic binnen het bedrijf wordt gebruikt is besloten het nieuwe systeem te realiseren in deze programmeertaal. Hieronder is een schematisch overzicht gegeven van de CLR.



Figuur 3

De base class library is een collectie van bruikbare functies die gebruikt kunnen worden bij het ontwikkelen van applicaties. Alle applicaties die werken op het .NET framework kunnen gebruik maken van deze functies. Deze collectie kan worden gebruikt voor zowel GUI (Graphical User Interface) applicaties als “command line” applicaties gebaseerd op ASP.NET, bijvoorbeeld webforms en XML webservices.

Deze twee onderdelen zijn onderzocht en verwerkt in het evaluatierapport: onderzoek Microsoft.NET.

5.3 ASP.NET

Tijdens het vorige deel van onderzoek is het concept van .NET besproken en is heeft men een globaal idee gekregen over de .NET omgeving. In dit onderdeel is er onderzocht naar de mogelijkheden die .NET bieden voor het ontwikkelen van webapplicaties. Hieruit is voorgekomen dat er met behulp van XML webservices en ASP.NET webpagina's over het Internet gegevens kunnen worden uitgewisseld. Bij het onderzoeken van XML webservices werd al snel bekend dat XML webservices alleen services bieden over het Internet dat dit niet geschikt was voor het ontwikkelen van een webapplicatie.

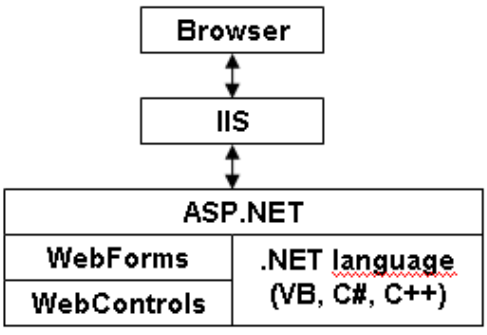
Het concept van ASP.NET is onderzocht en dit bleek geschikt te zijn voor het ontwikkelen van een webapplicatie. Daarnaast is het concept vastgelegd om de opdrachtgevers een idee te geven over ASP.NET. Ook is er vastgelegd hoe er met ASP.NET pagina's kunnen worden ontwikkeld. De belangrijkste onderdelen die hiervan zijn besproken zijn: Namespaces, de structuur van een ASP.NET pagina en webforms.

Omdat het .NET framework bestaat uit duizenden klassen (tot nu toe meer dan 3.400) worden deze klassen georganiseerd in een hiërarchie van "namespaces". Een namespace is een logische groepering van deze klassen. Een goed voorbeeld hiervan is de namespace: System.IO. Deze namespace bevat de klassen die gerelateerd zijn aan het werken met het bestandssysteem.

De structuur van de pagina is onderzocht om kennis te verkrijgen over de ontwikkeling van ASP.NET pagina's. Tijdens het onderzoeken van de structuur van de ASP.NET webpagina is kennis opgedaan over:

- Het instellen van de manier van compileren (van te voren compileren of "at runtime");
- De logica binnen een pagina;
- De controls die worden ondersteund (controls voor het genereren van tekst en HTML binnen een pagina) en hoe deze worden gebruikt;
- Serverside comment (commentaar in de code, niet zichtbaar voor de cliënt);
- Het vormen van de HTML inhoud van de pagina met behulp van tekst en HTML tags.

De basis van ASP.NET ligt in de webforms. Webforms bezitten net als de Windows forms dezelfde eigenschappen en methoden. Maar bij een ASP.NET webform wordt er gebruik gemaakt van UI elementen. Deze UI elementen zetten zichzelf om naar HTML.
Deze webforms zijn onderzocht omdat deze op een nieuwe manier worden verwerkt op de server. In ASP.NET worden de formulieren namelijk aan de kant van de server uitgevoerd. Webforms zijn opgebouwd uit 2 componenten, het visuele deel (het .ASPx bestand) en de code die daarachter wordt uitgevoerd (een aparte class file). Dit in tegenstelling tot ASP of php pagina's. De codebehind pagina (met een .NET language) wordt van te voren gecompileerd en alleen het visuele deel van de pagina wordt pas gecompileerd als hij wordt opgevraagd (on runtime). Hieronder is een schema weergegeven waarin het opvragen van een ASP.NET webform wordt opgevraagd.

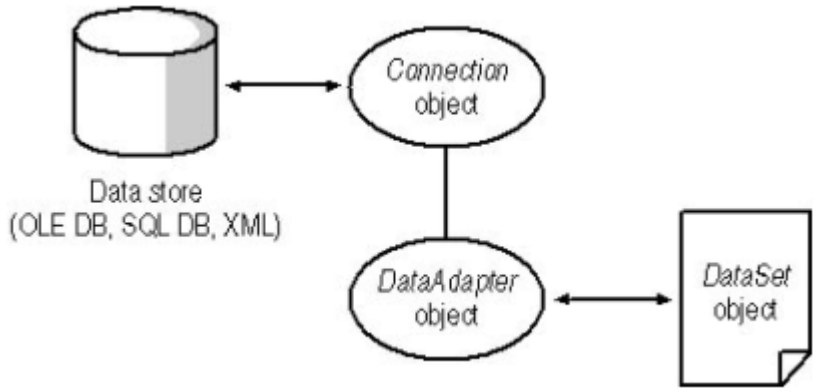


Figuur 4

5.4 ADO.NET

ADO.NET is een onderdeel van de Microsoft.NET omgeving en is binnen .NET de opvolger van ActiveX Data Objects (ADO). In het .NET Framework wordt gegevenstoegang gerealiseerd met een set van klassen onder de naam ADO.NET. ADO.NET is onderzocht om kennis op te doen over het opzetten van connecties met gegevensbronnen voor het muteren van gegevens.

Er is een introductie gegeven door de student over ADO.NET waarin wordt gegeven wat ADO.NET is en hoe ADO.NET werkt. Daarnaast is een stuk gebruikt van de Microsoft website waarin de onderdelen van ADO.NET worden besproken.
Met behulp van deze informatie kan worden bepaald welke mogelijkheden ADO.NET biedt voor het verwerken van gegevens en welke er het meeste geschikt is binnen het beveiligingssysteem. Hieronder is een schema weergegeven met daarin zichtbaar het ADO.NET object model.



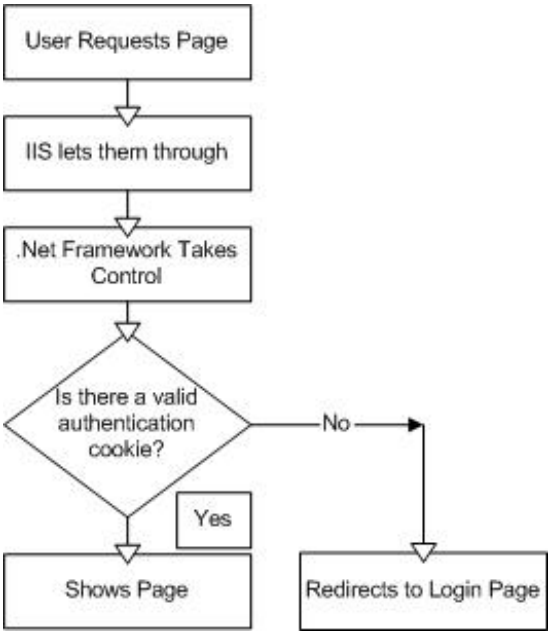
Figuur 5

5.5 Vormen van authentificatie

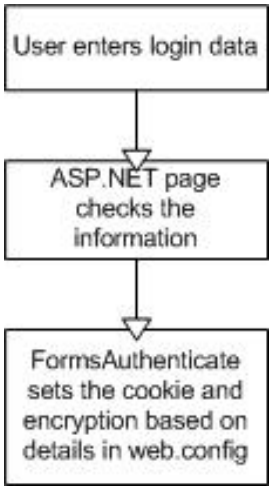
Elke gebruiker moet door het te ontwikkelen beveiligingssysteem worden geauthentificeerd. Binnen ASP.NET zijn er verschillende vormen van authentificatie beschikbaar. Om te kunnen bepalen welke vorm er het meest geschikt is voor toepassing in het beveiligingssysteem zijn deze authentificatie vormen onderzocht.

Om een simpel idee te geven over de werking van authentificatie is eerste de traditionele webbeveiliging (Windows Integrated Authentication) besproken zoals die vaak bij ASP wordt gebruikt.

Binnen ASP.NET kan er worden gewerkt met Windows authentificatie, Passport authentificatie en forms authentificatie. In het onderzoek zijn deze drie vormen van authentificatie onderzocht. Er is gekeken naar het niveau waar de beveiliging ligt en er is gekeken naar het authentificatie (login) proces. Van één van deze 3 vormen van authentificatie zijn op de volgende pagina twee figuren gegeven.



Figuur 6 (beveiliging)



Figuur 7 (proces)

In dit voorbeeld is forms authentication gebruikt. In het linker figuur (figuur 6) is een voorbeeld gegeven van de beveiliging binnen forms authentication en in het rechter figuur (figuur 7) is een voorbeeld gegeven het authentificatie proces van forms authentificatie.

In het linker schema is zichtbaar dat de gebruiker wordt gecontroleerd door het .NET framework. Als de gebruiker nog niet geauthentificeerd is (nog geen authentificatie cookie bezit) wordt hij doorgestuurd naar de Login Page.
In het rechter schema is zichtbaar dat de code in de pagina de gebruiker valideert en hem een cookie toewijst. In het rapport “onderzoek Microsft.NET” zijn alle authentificatie vormen, zowel op het niveau van beveiliging en op het niveau van de processen schematisch uitgewerkt en tekstueel omschreven.

5.6 Vormen van autorisatie

Omdat het beveiligingssysteem pagina’s beschikbaar zal gaan stellen met persoonlijke gegevens mogen niet alle gebruikers deze pagina’s inzien. Nadat een gebruiker is geauthentificeerd moet worden bepaald of de gebruiker toegang heeft (bevoegd is) om een bepaalde bron op te vragen. Dit kan worden bepaald door een autorisatie mechanisme te gebruiken binnen ASP.NET. Om te kunnen bepalen welke vormen van autorisatie er mogelijk zijn binnen ASP.NET is dit onderzocht. Dit is onderzocht door informatie over de autorisatie binnen ASP.NET op Internet op te zoeken.

Omdat er verschillende mogelijkheden zijn om te bepalen of een gebruiker toegang heeft tot en bepaalde bron zijn de verschillende vormen van autorisatie uitgewerkt. Binnen ASP.NET wordt autorisatie beheerd door de volgende componenten:

- Windows Access Control Lists (ACLs)
- Web Server permissies
- URL autorisatie
- .NET Principal Objects
- Roles and Method-Level beveiliging

Al deze componenten zijn in het onderzoeksrapport apart besproken.

Als voorbeeld kan er bijvoorbeeld gebruik worden gemaakt van de GenericPrincipal klasse.

> *Er kan een object worden gemaakt vanuit de GenericPrincipal klasse gebaseerd op zelf geconfigureerde rollen. Dit kan worden gebruikt as er gebruikt wordt gemaakt van een database voor de opslag van gebruikers/rollen. Het principel object kan worden aangesproken in het OnAuthenticate event. Er kan een gespecificeerde tabel worden gerelateerd aan een Windows account die wordt gebruikt bij dit event. Door die informatie te gebruiken, kan er een nieuw principel object worden aangemaakt voor de geauthentificeerde gebruiker.*

5.7 Configuratie van de beveiliging binnen ASP.NET

Nu de beschikbare vormen van authentificatie en autorisatie bekend zijn kan er gekeken worden naar de configuratie hiervan. Dit is met behulp van het volgende boek onderzocht:

> *Walther, Stephen, ASP.NET Uneleashed, SAMS, 1e editie, mei 2003*

Om van een authentificatie mechaniek gebruik te maken dient er ergens te worden vastgelegd dat de website geen anonieme gebruikers toelaat. Dit gebeurt binnen ASP.NET in de web.config. Het web.config bestand is van XML/formaat en moet worden geplaatst in de root van een virtuele map (IIS website). Dit bestand bevat de configuratie van de webapplicatie uit die virtuele map.

Binnen ASP.NET bevindt zich ook een zogenoemd applicatiebestand, onder de naam global.asax. Dit is een optioneel bestand met code voor events op applicatie niveau die worden gestart door ASP.NET of een HttpModule (Dit wordt later in het document nog besproken). Binnen het global.asax bestand kan code worden geplaatst die betrekking hebben op de applicatie. Zo kan er met behulp van dit bestand voor worden gezorgd dat er een bepaalde handeling wordt verricht telkens als een deel van de applicatie wordt opgevraagd of afgesloten.

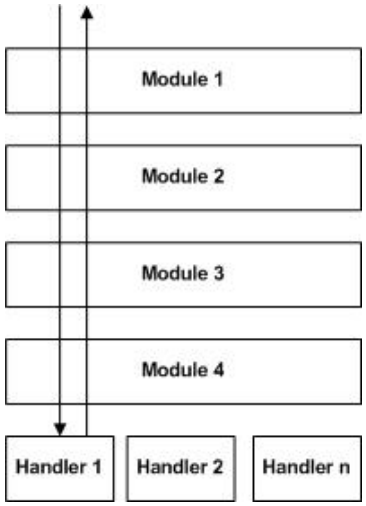
Zodra een gebruiker geauthentificeerd is, is het verstandig om zijn gegevens op te slaan in de applicatie (mits het om gegeven gaat van groot formaat). Dit omdat anders telkens als een gegeven benodigd is van een gebruiker een statement naar het dbms moet worden gestuurd. Om dit probleem te voorkomen is er onderzoek gedaan naar gegevens die aan een pagina, applicatie of kunnen worden toegekend. Dit kan worden gedaan door een View, Application of Session state bij te houden. De Session state houdt de gegevens bij van de gebruikers sessie. Hierin kunnen bijvoorbeeld ook op maatgemaakte datasets worden bewaard (onderdeel van ADO.NET).

Ook is het de bedoeling dat de gegevens gecodeerd over het Internet worden gestuurd. De SMRA is reeds in bezit van een SSL certificaat die hiervoor gebruikt kan worden. Er wordt kort beschreven wat SSL is en hoe dit mechanisme werkt.

5.8 HttpHandlers en HttpModules.

Tijdens het onderzoek naar de beveiliging binnen ASP.NET kwamen de termen httpHandlers en HttpModules vaan naar voren. Om deze reden zijn deze ook onderzocht en opgenomen in het onderzoeksrapport. HttpHandlers en HttpModules zijn ontwikkeld om extra functionaliteiten toe te voegen op het niveau van IIS (in werkelijkheid vlak na IIS).

Binnen ASP.NET wordt er request processing toegepast. ASP.NET request processing is gebaseerd op een pipeline model waarin ASP.NET de http request doorgeeft aan de HttpModules in de pipeline. Elke module die de http request ontvangt heeft volledige controle over de request. Zodra de request alle modulen heeft gepasseerd wordt het opgevangen door een http handler. De http handler verwerkt de request en geeft het resultaat terug in de pipeline. Het volgende figuur beschrijft het pipeline model.



Figuur 8

Als er een request wordt gedaan voor een bepaalde pagina worden de modulen doorgelopen en de bijbehorende HttpHandler wordt uitgevoerd. De HttpHandler kan aan een extensie worden gekoppeld (.smra) en voor elke aanvraag naar een bestand dat met deze extensie eindigt wordt deze handler uitgevoerd. Als er gebruik wordt gemaakt van een http handler moet deze worden geregistreerd in de web.config.

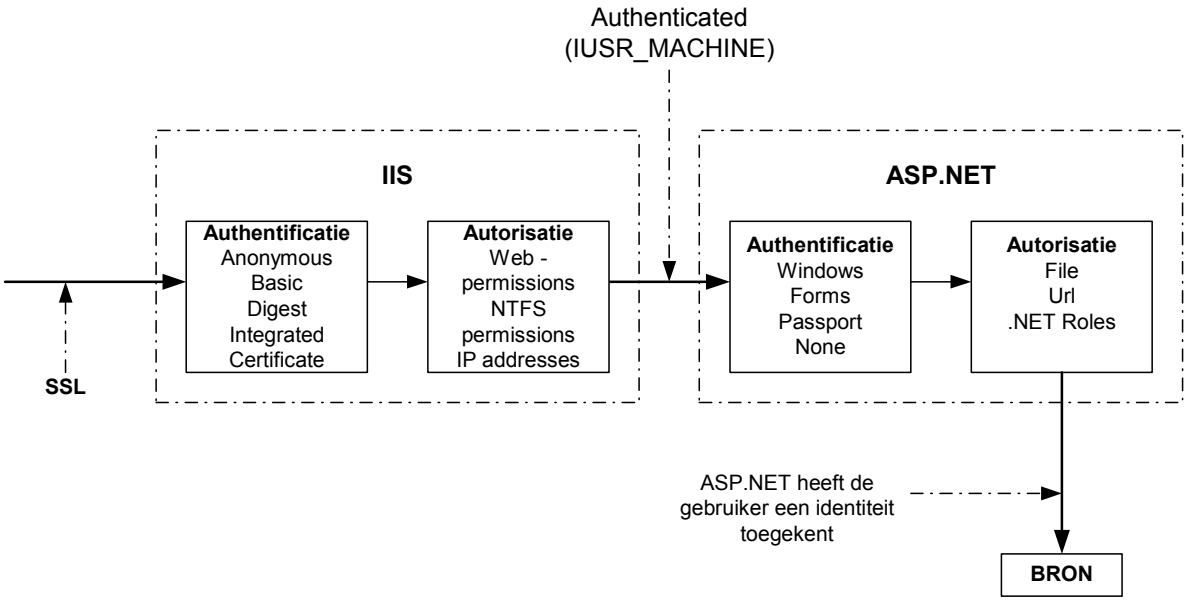
HttpModules zijn de .NET componenten die de System.Web.IHttpModule interface implementeren. Deze componenten voegen zich in de ASP.NET processing pipeline en registreren zich voor bepaalde events. Zodra zo'n event zich voordoet haalt ASP.NET de bijbehorende module erbij voor het uitvoeren van de request. Elk event moet worden geregistreerd. De HttpModule zelf moet ook worden geregistreerd in de web.config.

5.9 Overzicht van de beveiliging

Het onderzoeksrapport van Microsoft.NET werd onoverzichtelijk door de opeenvolging van informatie die tijdens het onderzoek is verworven. Om deze reden zijn de onderzochte onderdelen in schema's geplaatst. De manieren van beveiliging zijn naast elkaar gezet en in schema's verwerkt zodat er kan worden waargenomen hoe de processen verlopen. Ook is er in dit onderdeel bepaald welke onderdelen het meest geschikt zijn voor implementatie bij de SMRA samen met de bijbehorende configuratie.

5.9.1 Overzicht Authentificatie en Autorisatie.

Als eerste is er een overzicht gemaakt van de Authentication/Authorization request flow. Op de volgende pagina is dit schema gegeven (Figuur 9).

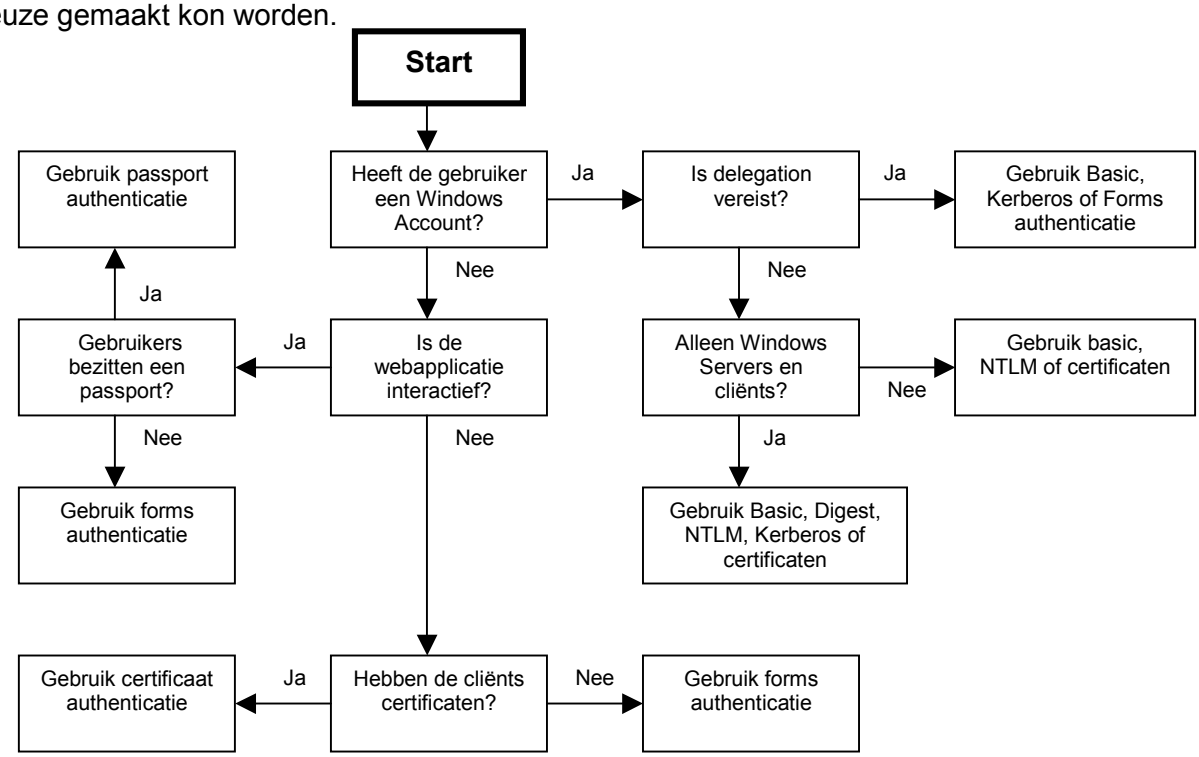


Figuur 9

In dit overzicht is zichtbaar waar de vormen van authenticatie en autorisatie plaatsvinden en welke overheersend zijn (als een gebruiker door IIS niet wordt toegelaten zal het authenticatie proces van ASP.NET in werking treden).

5.9.2 Keuze authentificatie

Hierna zijn de verschillende vormen van authenticatie in een overzicht geplaatst zodat er een keuze gemaakt kon worden.



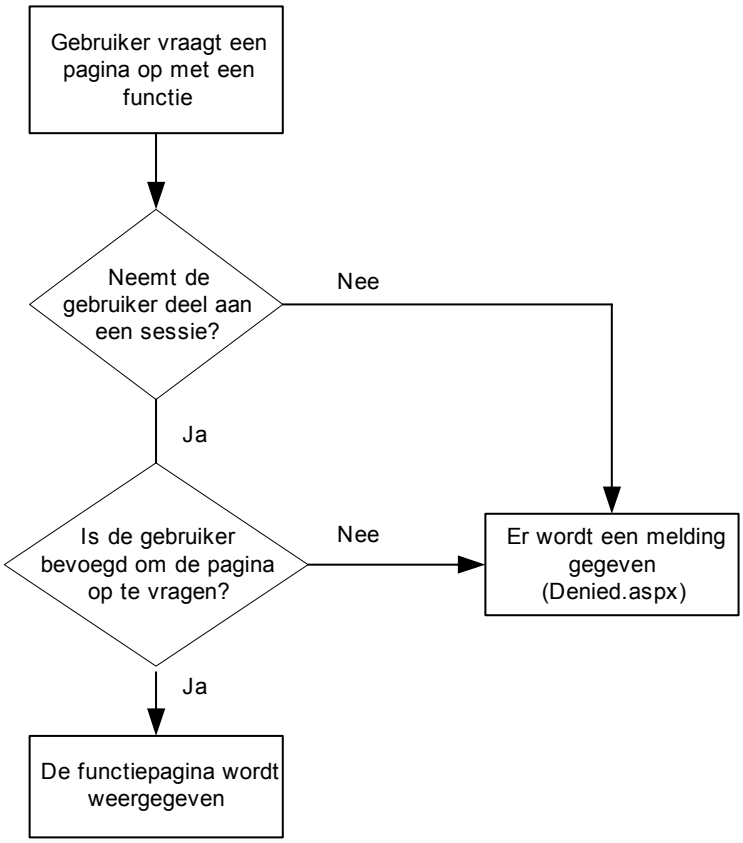
Figuur 10

Aan de hand van dit schema is de keuze gemaakt. Als eerste staat vast dat er geen gebruik wordt gemaakt van Windows accounts en daarnaast wordt er gebruik gemaakt van een

Interactieve webapplicatie. Dit resulteerde in de vraag of gebruikers een passport bezitten. Dit is niet het geval. Er is besloten gebruik te maken van forms authentificatie.

5.9.3 Autorisatie

Binnen de ASP.NET omgeving bevinden zich een aantal mogelijkheden voor gebruikersautorisatie. Na het doornemen van de autorisatie vormen bleek dat er geen standaard authenticatie mechanisme in ASP.NET zit die toepasbaar is binnen dit project. Om deze reden is er een eigen autorisatie mechanisme ontwikkeld in een HttpModule. Deze module controleert bij het opvragen van een pagina met een functie, of de gebruiker voldoet aan de gestelde eisen. De werking van de HttpModule is op de volgende pagina schematisch weergegeven (figuur 11).

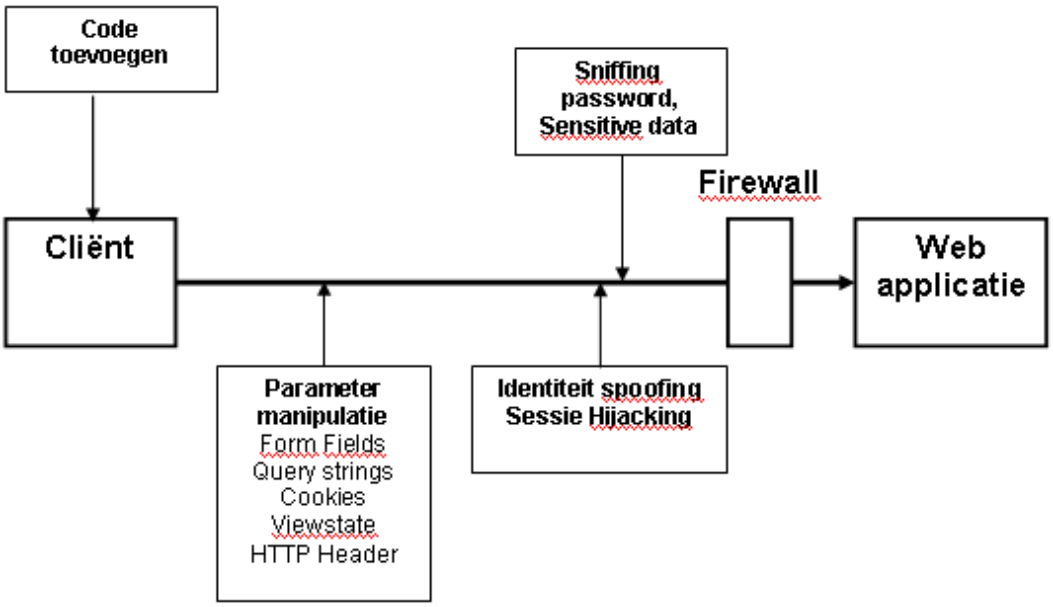


Figuur 11

Er is gekozen om gebruik te maken van een HttpModule omdat deze module bij elke aanvraag van een pagina wordt uitgevoerd. Zo kan er in de configuratie worden aangegeven waar de pagina's met functies zich bevinden en bij het opvragen van deze pagina's zal de HttpModule controleren of de gebruiker de pagina mag opvragen. Als dit niet het geval is, wordt de gebruiker worden doorgestuurd naar een pagina waarop wordt aangegeven dat hij geen toegang heeft tot de betreffende pagina.

5.9.4 Aanvallen op een webapplicatie

Ook in dit onderdeel is er onderzocht naar de mogelijkheden van aanvallen op een webapplicatie. Door dit te analyseren kan er rekening worden gehouden met de "hack" gevoelige punten van een webapplicatie. Hieruit is ook een model naar voren gekomen. Deze is op de volgende pagina weergegeven in figuur 12.



Figuur 12

De “Hack” gevoelige punten zijn op Internet opgezocht en er is onderzocht welke maatregelen genomen kunnen worden om de applicatie hiertegen te beveiligen. In het onderzoeksrapport zijn deze maatregel vastgelegd en omschreven.

5.9.5 Impersonation en delegation

Bij het onderzoeken van Visual studio.NET 2003 waren er problemen met het opzetten van een connectie met de database. Na het raadplegen van veel informatiebronnen bleek het gebruik van impersonation vereist te zijn. Impersonation zorgt ervoor dat een anonieme gebruiker wordt gekoppeld aan het account dat wordt gespecificeerd door het IIS anonimous user account. Door impersonation te gebruiken kunnen er gemakkelijk verbindingen worden gemaakt met bijvoorbeeld een SQL database. Het Windows account gekoppeld aan de anonimous user wordt dan gebruikt voor de toegang op de SQL database en er wordt zo een “trusted connection” opgezet.

5.10 Onderzoek Visual studio.NET 2003

Tijdens deze fase is de ontwikkelomgeving Visual studio.NET ook onderzocht. Deze omgeving is onderzocht omdat de SMRA zelf wil gaan ontwikkelen in deze omgeving. Als de beveiligingsapplicatie in deze omgeving ontwikkeld zal worden, zal het beheren en aanpassen van de applicatie gemakkelijker zijn.

Het onderzoeken van Visual studio was gericht op het leren werken met de ontwikkelomgeving. Om deze reden heeft dit niet een apart document opgeleverd.
De ontwikkelomgeving bleek een zeer uitgebreide ontwikkelomgeving te zijn en in combinatie met de onbrekende kennis van deze omgeving en ASP.NET heeft dit onderdeel bijna 2 weken in beslag genomen (hierbij is ook de start van het leren ontwikkelen in ASP.NET gerekend).

Het moeilijke van het werken met deze ontwikkelomgeving kwam voornamelijk door het grafische deel en het code deel. Zodra er met behulp van het grafische deel iets werd ingevoegd, werd de achterliggende code hiervoor automatisch gegenereerd. Dit zorgde vaak voor een grote hoeveelheid, ingewikkelde code.

5.11 Afronding

Nadat de afzonderlijke hoofdstukken zijn geschreven, is de opmaak van het document is gecontroleerd aan de hand van de checklist die als bijlage is opgenomen bij het plan van aanpak. Hierna is dit document opgeleverd aan de opdrachtgevers.
Uiteindelijk bleek dat de verdere voortgang van het project er steeds meer onderdelen onderzocht moesten worden, zoals onder andere het gebruik van cookies en hun encryptie methoden. Deze zijn niet in het verslag verwerkt omdat de kennis over deze onderdelen nodig was bij het bouwen van de applicatie en niet bij het functionele ontwerp van het beveiligingssysteem.

6. Definitiestudie

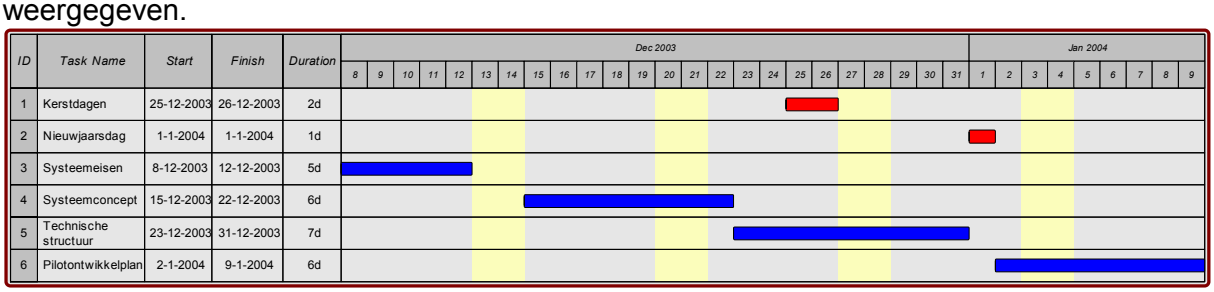
6.1 Inleiding

In dit hoofdstuk wordt de totstandkoming van de definitiestudie beschreven. De definitiestudie had als doel om de systeemeisen vast te leggen en op basis hiervan een systeemconcept te ontwikkelen. Op basis van het systeemconcept is het mogelijk het systeem onder te verdelen in verschillende pilots. Verder wordt er in de definitiestudie gekeken of het project haalbaar is. In totaal heeft het doorlopen van de definitiefase ongeveer 2 weken in beslag genomen. Het document definitiestudie is bijgevoegd als bijlage “C”.

6.2 Plan van aanpak

Het doel van dit onderdeel was het opleveren van een gedetailleerd plan van aanpak gericht op de definitiefase. Ook zijn er afspraken vastgelegd over de inhoud, omvang en uit te voeren activiteiten binnen de definitiestudie. Dit is gedaan door de definitiefase op te delen in verschillende fasen met de daarbijbehorende activiteiten. Voor elke fase is er een schatting gemaakt over het aantal uren dat verwacht wordt nodig te zijn voor het uitvoeren van de fase. Dit is gedaan met behulp van de formule: (minimaal + maximaal + 2 * gemiddeld) / 4 = planning). Op basis van de indeling van de fasen en de ingeschatte uren is er een nauwkeurige planning opgesteld specifiek voor de definitiefase. Deze is hieronder weergegeven.

ID	Task Name	Start	Finish	Duration
1	Kerstdagen	25-12-2003	26-12-2003	2d
2	Nieuwjaarsdag	1-1-2004	1-1-2004	1d
3	Systeemeisen	8-12-2003	12-12-2003	5d
4	Systeemconcept	15-12-2003	22-12-2003	6d
5	Technische structuur	23-12-2003	31-12-2003	7d
6	Pilotontwikkelplan	2-1-2004	9-1-2004	6d



Figuur 14

De Fase Pilot evaluatie is niet in dit onderdeel opgenomen omdat deze fase pas zal worden uitgevoerd nadat er een pilot is gerealiseerd.

Deze planning geeft de activiteiten weer tegen over de tijd en geeft aan op welk tijdstip er een product zal worden opgeleverd.

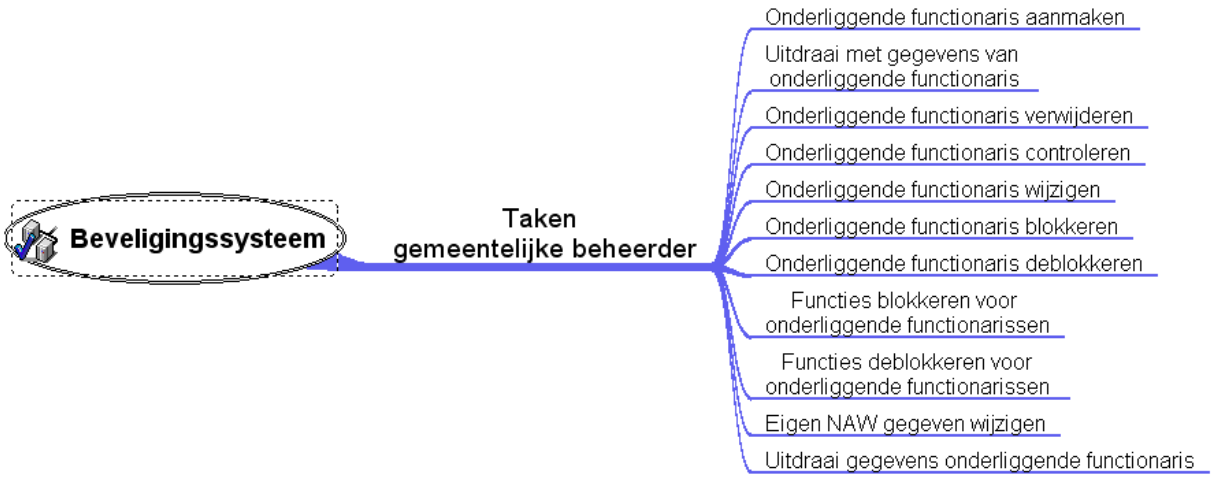
6.3 Pilot evaluatie

Als een pilot wordt opgeleverd wordt over de pilot een presentatie gegeven. Voor deze presentatie worden meerdere personeelsleden uitgenodigd die werkzaam zijn op meerdere afdelingen. De presentaties zijn gegeven met behulp van MS-Powerpoint en zijn zo opgezet dat de gebruikers een indruk krijgen van de kernfunctionaliteiten en de mogelijkheden van de opgeleverde pilot.
Het doel van deze presentaties was het verkrijgen van op- of aanmerkingen, tips of andere wensen over het betreffende pilotdeel. Het verkregen commentaar is bij deze opdracht alleen verwerkt in de documentatie omdat de eisen en wensen van het op te leveren systeem van te voren waren vastgesteld. Dit commentaar kan later gebruikt worden om de functionaliteiten van het systeem uit te breiden of aan te passen.

6.4 Systeemeisen

Voor het bepalen van de systeemeisen is de opdrachtomschrijving gebruikt en zijn er meerdere vergadering geweest. De resultaten zijn met behulp van de “mindmap techniek” vastgelegd. Deze techniek heeft een schema opgeleverd met daarin de eisen die aan het systeem gesteld zijn. Dit is een vrij simpele techniek die gemakkelijk is toe te passen.

Bij het opstellen van de mindmap is bepaald welke gebruikerstypen er gebruik maken van het nieuwe systeem. Per gebruikerstype zijn de functionaliteiten bepaald en vastgelegd als taken. Er zijn algemene functionele systeemeisen opgesteld waarin de overige functionele eisen van de applicatie zijn vastgelegd. Daarnaast zijn er ook de systeemeisen vastgelegd. De uiteindelijke mindmap is groot van omvang geworden en om deze reden is hieronder alleen het deel van de mindmap gegeven met de taken van de van de gemeentelijke beheerder (de volledige versie van dit diagram is wel terug te vinden in de definitiestudie uit de bijlage).



Figuur 15

De eisen die tijdens het gebruiken van de mindmap techniek zijn gekozen, zijn met behulp van een MoSCoW analyse onderverdeeld in prioriteiten. Het gebruiken van de MoSCoW regels was voor de afstudeerder geheel nieuw en er was enig onderzoek nodig om deze regels toe te kunnen passen op de functionele eisen.

Door middel van de MoSCoW regels zijn de eisen ingedeeld als een Must Have, Should Have, Could Have of Would be nice to Have.

MoSCoW staat voor:

- 'Must Have': deze categorie heeft de hoogste prioriteit, wordt gegarandeerd opgeleverd en geldt als de 'motor' van het informatiesysteem.
- 'Should Have': een noodzakelijke eis, maar er is een (tijdelijke) 'work-around' mogelijk.
- 'Could Have': eis met een duidelijke toegevoegde waarde, maar zonder is er nog steeds een bruikbaar systeem.
- 'Would be nice to Have': de opdrachtgever ziet deze eis als laagste prioriteit. Deze eisen worden verwerkt in het systeem afhankelijk van de overgebleven tijd die is gereserveerd voor het uitvoeren van het project.

Met de opdrachtgevers is de prioriteit van elke eis besproken en vastgelegd. Als voorbeeld zijn op de volgende pagina de functionele eisen gegeven van de gemeentelijke beheerder volgens de MoSCoW regels.

NR	Functionele eis	MoSCoW
F19	Kan een onderliggende functionaris aanmaken	**M**
F20	Kan een uitdraai met gegevens maken van een onderliggende functionaris	**M**
F21	Kan een onderliggende functionaris verwijderen	**M**
F22	Kan een onderliggende functionaris wijzigen/controleren	**M**
F23	Kan functionarissen uit de betreffende gemeente blokkeren	**M**
F24	Kan functionarissen uit de betreffende gemeente deblokkeren	**M**
F25	Een gemeentelijke beheerder kan functies blokkeren en deblokkeren voor onderliggende functionarissen	**C**
F26	Kan zijn eigen adresgegevens wijzigen	**C**

De niet functionele eisen zijn vastgelegd in een overzicht zonder gebruik te maken van MoSCoW. Het is namelijk vereist dat alle niet functionele eisen in het systeem zijn verwerkt. Na het afronden van de eerste en tweede pilot zijn er pilotworkshops gehouden die extra eisen en wensen aan het systeem hebben toegevoegd. Deze zijn wegens de gekozen iteratiestrategie ook vastgelegd in de definitiestudie.

6.5 Systeemconcept

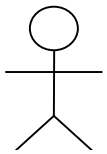
Op basis van de systeemeisen is een eerste functionele, gebruikersgerichte beschrijving van het systeem ontwikkelt. Het systeemconcept is een beschrijving op functioneel niveau van de oplossing. Hierin licht een functionele architectuur besloten die de functionele componenten en hun afhankelijkheden aangeeft.

Het systeemconcept is ontwikkeld aan de hand van de UML techniek “Use Cases”. Use-Cases zijn een middel binnen UML om de functionele eisen die gesteld worden aan het systeem weer te geven. De use-cases beschrijven het gebruik van het systeem uit het oogpunt van de gebruiker.

6.5.1 Actoren analyse

De eerste stap bij het opstellen van het systeemconcept is het bepalen van de actoren. Actoren zijn entiteiten die buiten het systeem staan en direct communiceren met het systeem. Dit kan zowel een mens, een ander systeem of zelfs meerdere personen zijn.

Elke actor is vastgelegd met de daarbijbehorende functionaliteiten die de gebruiker uit wil voeren. Hieronder is een voorbeeld van de actor gemeentelijke beheerder weergegeven.



Gemeentelijke beheerder

<<actor>>
Gemeentelijke beheerder

LogIn()
EigenNawWijzigen()
OnderliggendeFunctionarisAanmaken()
OnderliggendeFunctionarisPrinten()
OnderliggendeFunctionarisVerwijderen()
OnderliggendeFunctionarisWijzigen()
OnderliggendeFunctionarisOpvragen()
OnderliggendeFunctionarisBlokkeren()
OnderliggendeFunctionarisDeblokkeren()
FunctiesWijzigenOnderliggendeFunctionaris()

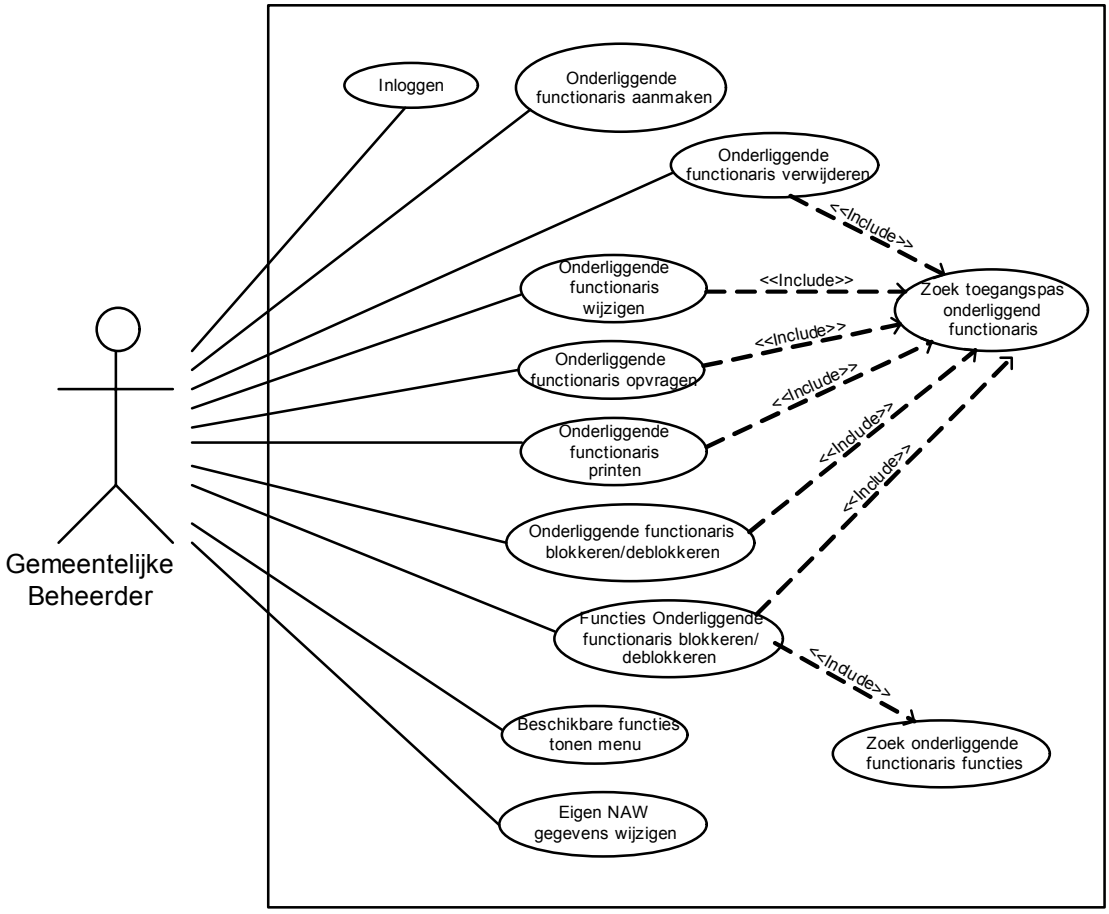
De actoren analyse bleek uiteindelijk overbodig omdat dit in de mindmap is verwerkt. Dit geldt ook voor de functionaliteiten van de actoren.

6.5.2 Use-cases

De tweede stap is het realiseren van de Use-cases. De use-cases geven een beschrijving van interacties tussen de actoren en het systeem, vanuit het oogpunt van de gebruiker. Hieronder is een voorbeeld van een functionaliteit gegeven die wordt uitgevoerd door de gemeentelijke beheerder.

Naam	GemeentelijkeBeheerderAanmaken()
Aannamen	Het scherm "Gemeentelijke beheerder aanmaken" wordt getoond.
Beschrijving	1. De SMRA beheerder voert een nieuwe gebruikersnaam, de bijbehorende NAW gegevens en de bijbehorende gemeente in.. 2. Het systeem genereert een bijbehorend wachtwoord. 3. De SMRA beheerder kan nu opslaan kiezen. Indien hij dit niet kiest treedt er een uitzondering op.[1] 4. De wijzigingen worden opgeslagen en er wordt getoond dat de wijzigingen zijn doorgevoerd. Als de gebruikersnaam al bestaat treedt er een uitzondering op.[2] 5. Use-case gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Word na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen. 2. Er wordt gecontroleerd of de gebruikersnaam niet reeds bestaat. Indien de gebruiker al bestaat worden de gegevens niet opgeslagen en wordt er een melding gegeven dat de gebruikersnaam al in gebruik is. Het scherm keert terug naar de ingevoerde gebruikersgegevens.
Resultaat	Er is een gemeentelijke beheerder aangemaakt.

Na het realiseren van de use-cases is er een use-case-diagram gerealiseerd. Dit diagram voegt niet veel informatie toe maar geeft een duidelijk overzicht van de mogelijkheden die het nieuwe systeem aan de gebruiker geeft. Omdat het te realiseren systeem uit veel functionele eisen bestond en het model onoverzichtelijk werd is er een use-case diagram ontwikkeld per actor. Hieronder is het use-case diagram van de gemeentelijke beheerder gegeven.



Figuur 16

6.5.3 Aggregatie functies

De derde stap is het aggregeren van de functies. Omdat het in de aparte use-case-diagrammen niet duidelijk zichtbaar is welke functies in praktijk dezelfde functionaliteiten zijn, zijn deze geaggregeerd. De functionaliteiten zijn in een matrix verwerkt om zo meer inzicht te krijgen over gelijke functionaliteiten. Een deel van deze matrix is hieronder weergegeven.

Functie	SMRA telefoniste	SMRA Beheerder	Gemeentelijke beheerder	Functionaris
LogIn()	**X**	**X**	**X**	**X**
VraagGebruikerGegevensOp()	X			
EigenNawWijzigen()			**X**	**X**
GemeentelijkeBeheerderAanmaken()		X		
GemeentelijkeBeheerderPrinten()		X		
GemeentelijkeBeheerderVerwijderen()		X		

Het viel op dat er nog andere functies binnen het systeem waren die “vrijwel” synoniem waren. Om deze reden zijn ook deze functionaliteiten in een overzicht vastgelegd. Dit overzicht is hieronder weergegeven.

GemeentelijkeBeheerderAanmaken()	OnderliggendeFunctionarisAanmaken()
GemeentelijkeBeheerderPrinten()	OnderliggendeFunctionarisPrinten()
GemeentelijkeBeheerderVerwijderen()	OnderliggendeFunctionarisVerwijderen()
GemeentelijkeBeheerderWijzigen()	OnderliggendeFunctionarisWijzigen()
GemeentelijkeBeheerderOpvragen()	OnderliggendeFunctionarisOpvragen()
FunctiesGemeentelijkeBeheerderWijzigen()	FunctiesWijzigenOnderliggendeFunctionaris()

6.5.4 Klassendiagram

Bij het realiseren van het klassendiagram is er gebruik gemaakt van het boek:
Warmer, J., en Kleppe A., Praktisch UML, Addison Wesley, 2e editie, 2001.

In dit boek is een werkwijze gegeven voor het ontwikkelen van een klassendiagram. De eerste stap voor het ontwikkelen van een klassendiagram is het identificeren van alle mogelijke kandidaat-klassen. De kandidaat-klassen zijn bepaald door het onderstrepen van de zelfstandige naamwoorden uit de probleembeschrijving en zijn daarna verder uitgewerkt tijdens het uitvoeren van een brainstormsessie. De geïdentificeerde kandidaten zijn in een tabel gezet en er is bepaald welke kandidaat-klassen klassen zijn.
Wanneer is een kandidaat-klasse een klasse? Dit wordt beantwoord door de volgende vragen te stellen bij een kandidaat-klasse.

> *Speelt de kandidaat-klasse een zelfstandige rol in het probleemdomein? Willen we iets van de kandidaat weten of willen we dat hij iets voor mij doet? Heeft de kandidaat-klasse een duidelijke verantwoordelijkheid in het systeem?*

Hieronder zijn in een voorbeeld de kandidaat-klassen Gebruiker en Gebruikersnaam gegeven.

Kandidaat-klasse	Beslissing
Gebruiker	Is klasse
Gebruikersnaam	Attribuut

Door de vraag te stellen bij de kandidaat-klasse gebruiker is het duidelijk dat kandidaat-klasse: *gebruiker* een klasse is. De kandidaat-klasse: *gebruikersnaam* is een attribuut van de klasse: *gebruiker*. Gebruikersnaam heeft geen enkele zelfstandige verantwoordelijkheid en is dus geen instantie van een klasse. De waarde van gebruikersnaam is echter wel interessant.

Deze geeft informatie over de klasse gebruiker en kan later worden meegenomen als attribuut in het klassendiagram.

Na het vinden van de klassen is er een modeldictionary opgesteld. Alle gevonden klassen zijn hierin opgenomen met een daarbijbehorende beschrijving. Hieronder een voorbeeld van de klasse gebruiker gegeven zoals die zou worden weergegeven in een modeldictionary.

klasse	Omschrijving
Gebruiker	Representatie binnen het systeem van een reëel persoon.

Hierna zijn de bijbehorende attributen en operaties bepaald. Een aantal van deze attributen zijn uit de lijst van kandidaten overgenomen. De andere attributen zijn bepaald in overleg met de opdrachtgevers door ons af te vragen welke informatie we van de klassen willen bijhouden. Ook hier is gebruik gemaakt van een vraag waarmee een attribuut kan worden gevonden. De vraag luidt:

> *Wat zijn dingen waar en object van weet? Wat voor informatie zouden we aan het object kunnen vragen?*

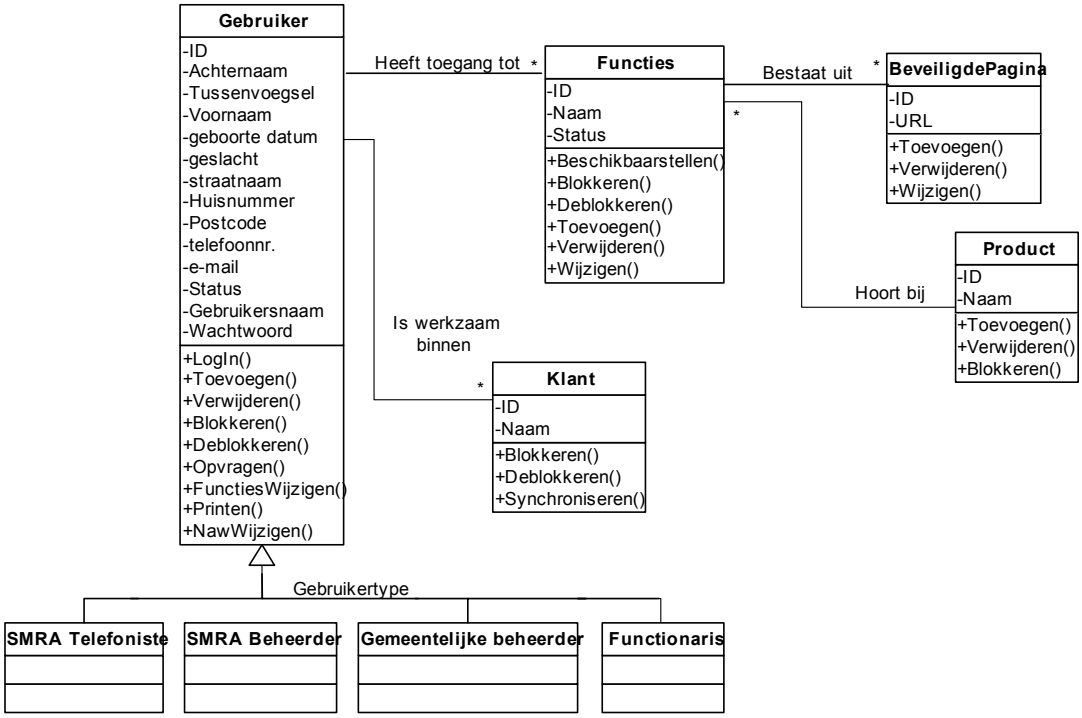
Veel van de attributen zijn pas bepaald in een later stadium tijdens het project. Dit omdat het inzicht van het systeem en zodoende de mogelijkheid nog niet volledig was.
Ook de operaties van een object moeten worden bepaald. Ook hier is gebruik gemaakt van een vraag waarmee operaties kunnen worden bepaald. De vraag om operaties te bepalen luidt:

> *Wat wil ik dat dit object voor mij doet? Wat is de verantwoordelijkheid van dit object en hoe kan ik deze vervullen?*

Ook operaties zijn in een later stadium nog toegevoegd. Hieronder is een voorbeeld gegeven van de eerste versie van de klasse Gebruiker gegeven met de bijbehorende Attributen en operaties.

klasse	Attributen	Operaties
Gebruiker	ID Voornaam Tussenvoegsel Achternaam Geboorte datum Geslacht Straatnaam Huisnummer Postcode Gebruikerstype Gebruikersrol Gemeente_ID	LogIn() Toevoegen() Verwijderen() Blokkeren() Deblokkeren() Opvragen() FunctiesWijzigen() Printen() NawWijzigen()

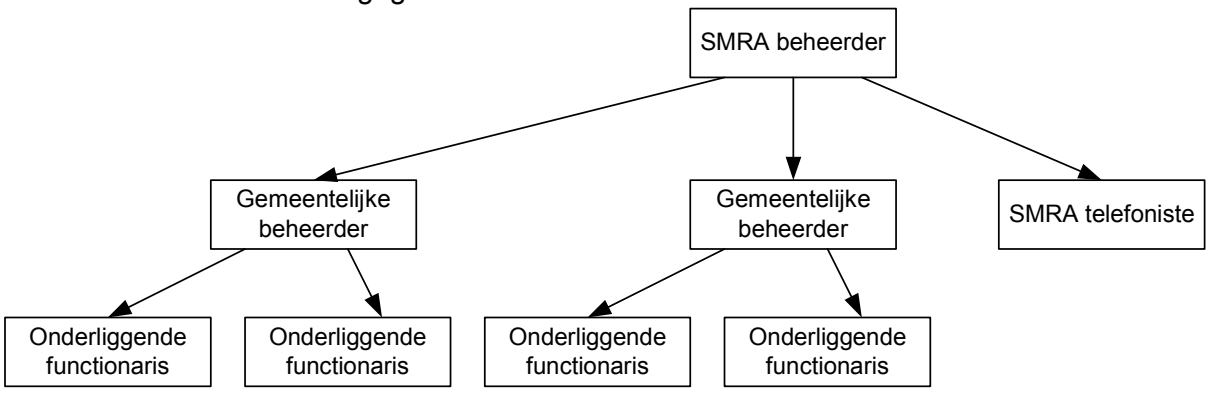
Nadat de voorgaande stappen zijn uitgevoerd is er een klassendiagram gemaakt. Tijdens het opzetten van het klassendiagram zijn de associaties bepaald (in tegenstelling tot de werkwijze van het boek) en verwerkt in het klasse diagram. Dit is gedaan door alle klassen te vergelijken en de klassen te noteren die iets met elkaar te maken hebben. Uiteindelijk is dat allemaal in een model verwerkt en is hieruit een eerste digitale versie van het klassendiagram voortgekomen. Dit diagram is op de volgende pagina weergegeven (figuur 17).



Figuur 17

Omdat dit volgens de opdrachtgevers niet voldoende inzicht gaf over het systeem is er ook een tekstuele beschrijving gegeven van het systeemconcept met hierin enkele modellen. Een belangrijk onderdeel van de opdracht was namelijk het gelaagde beveiligingsmodel. Deze kwam uit de omschrijving met de UML technieken niet naar voren.

Binnen het gebruikersbeheer wordt er gebruik gemaakt van een gelaagd beveiligingsmodel. Dit model is hieronder weergegeven.

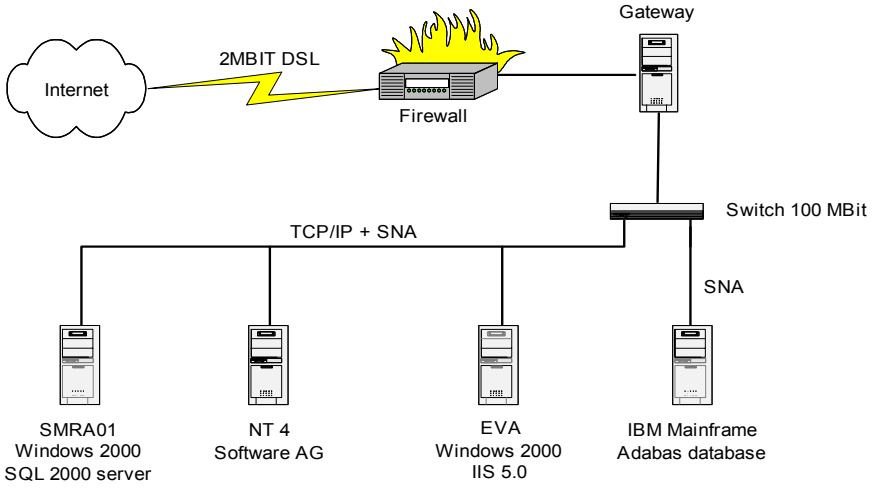


Figuur 18

Door gebruik te maken van dit model kunnen de functionaliteiten van het systeem worden doorgegeven per gebruikertype. Zo kan worden uitgesloten dat bepaalde gebruikers van een gebruikertype geen functionaliteiten kunnen gebruiken waartoe zij niet zijn bevoegd.

6.6 Technische structuur

De technische structuur van de SMRA is bekeken zodat er gecontroleerd kan worden of het te realiseren systeem kan worden toegepast in de huidige structuur. De technische structuur is besproken bij meerdere vergaderingen met de systeembeheerder. Hieruit is het volgende schema voortgekomen (figuur 19).



Figuur 139

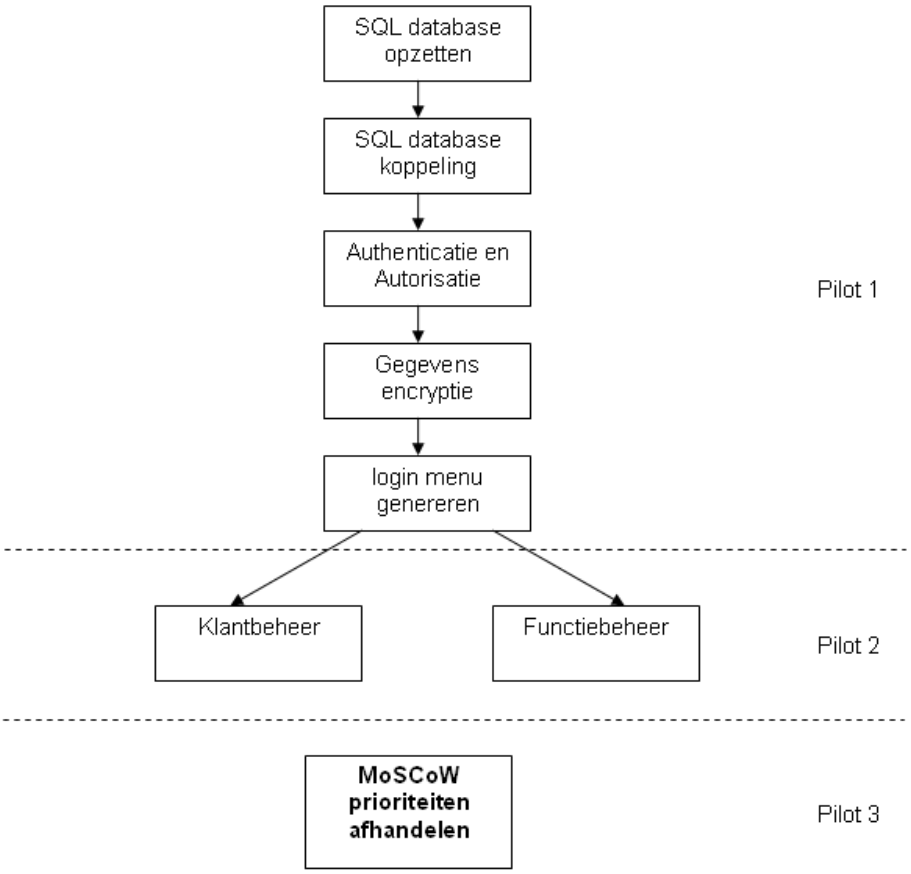
Door de technische structuur te vergelijken met de benodigde faciliteiten is er geconstateerd dat deze faciliteiten in de technische structuur aanwezig zijn. (De benodigde faciliteiten zijn bepaald tijdens het realiseren van het plan van aanpak). Veranderingen binnen het fysieke deel van netwerk zijn niet nodig. Bij de controle van de software versies bleek dat de EVA server gebruikt maakte van IIS 5.0 en moet worden bijgewerkt naar versie 5.1. Ook moet het .NET Framework geïnstalleerd worden op de webserver.

Om te kunnen communiceren met de Adabas database moet er gebruikt worden gemaakt van de Software AG service. Deze service vertaalt de requests van TCP/IP naar SNA formaat. Dit maakt het mogelijk om alsnog mutaties uit te voeren op de gegevens binnen de Adabas database.

6.7 Pilot ontwikkelplan

Dit onderdeel had als doel een pilotplan op te leveren. Een pilotplan wordt gebruikt voor het besturen van de pilotontwikkeling. Er wordt een pilotstructuur gerealiseerd, door gedeelten van het systeemconcept te combineren tot clusters. Door de afhankelijkheden tussen de clusters te analyseren kunnen deze tot zelfstandige pilots worden ingedeeld die apart kunnen worden ontwikkeld.

De systeemeisen en wensen zijn verwerkt tot clusters. Door de clusters in een overzichtelijke structuur te plaatsen is er geanalyseerd welke clusters er samengevoegd worden om een pilot te vormen. In het volgende figuur is deze structuur weergegeven.



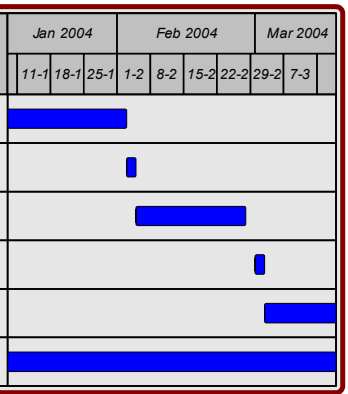
Figuur 20

Om een inschatting te kunnen maken van de tijd die er per pilot nodig zal zijn om deze te ontwikkelen, zijn per pilot de subfasen bepaald en vastgelegd in een overzicht.

Bij het afhandelen van de MoSCoW prioriteiten worden de wensen die nog niet zijn ingevoerd (alle anders dan must have's) in het systeem verwerkt. Dit gebeurt aan de hand van de prioriteiten zoals die zijn toegekend met de MoSCoW regels. De aanpassingen worden doorgevoerd in zowel de definitiestudie als in de betreffende pilotrapporten. Door de MoSCoW prioriteiten als apart onderdeel op te nemen is het mogelijk om na het verwerken van de "must have" eisen van pilot één en twee de overige eisen onafhankelijk van de pilot door te voeren.

Om de hoeveelheid tijd die nodig is voor het ontwikkelen van een pilot te nauwkeurig te kunnen inschatten, is er een urenplanning gemaakt. Ook hier is de formule (minimaal + maximaal + 2 * gemiddeld) / 4 = planning) toegepast. Als volgt is er een overzicht van de planning gegeven. Deze is op de volgende pagina weergegeven in figuur 21.

ID	Task Name	Start	Finish	Duration
1	Pilot 1	9-1-2004	2-2-2004	17d
2	Definitiestudie	3-2-2004	4-2-2004	2d
3	Pilot 2	5-2-2004	27-2-2004	17d
4	Definitiestudie	1-3-2004	2-3-2004	2d
5	Pilot 3	3-3-2004	17-3-2004	11d
6	Eindverslag	9-1-2004	17-3-2004	49d



Figuur 21

6.8 Afronding

Nadat de afzonderlijke hoofdstukken zijn geschreven, is het document definitiestudie gecontroleerd op de opmaak aan de hand van de checklist die als bijlage is opgenomen bij het plan van aanpak. Na de goedkeuring van de opdrachtgevers is het project vervolgd met de fase pilotontwikkeling. Na het uitvoeren van elke pilot is de definitiestudie bijgewerkt.

7. Pilotontwikkeling, pilot 1

7.1 Inleiding

In dit onderdeel is de uitvoering van de fase pilotontwikkeling beschreven. In deze fase is er gewerkt aan het ontwerpen en bouwen van de in het pilotplan opgestelde pilots. Een pilot biedt een operationele subset van de functionaliteit van het uiteindelijk beoogde systeem.

Omdat het onderhoud van de applicatie in een later stadium overgenomen wordt door medewerkers van de SMRA, is de applicatie ontwikkeld met behulp Visual Studio.NET 2003. Het bleek dat Visual studio.NET 2003 een zeer uitgebreide ontwikkelomgeving is en dat dit in combinatie met de nieuwe Microsoft .NET omgeving tijd koste om hierin te leren ontwikkelen. In totaal heeft pilot 1 ongeveer 5 weken in beslag genomen.

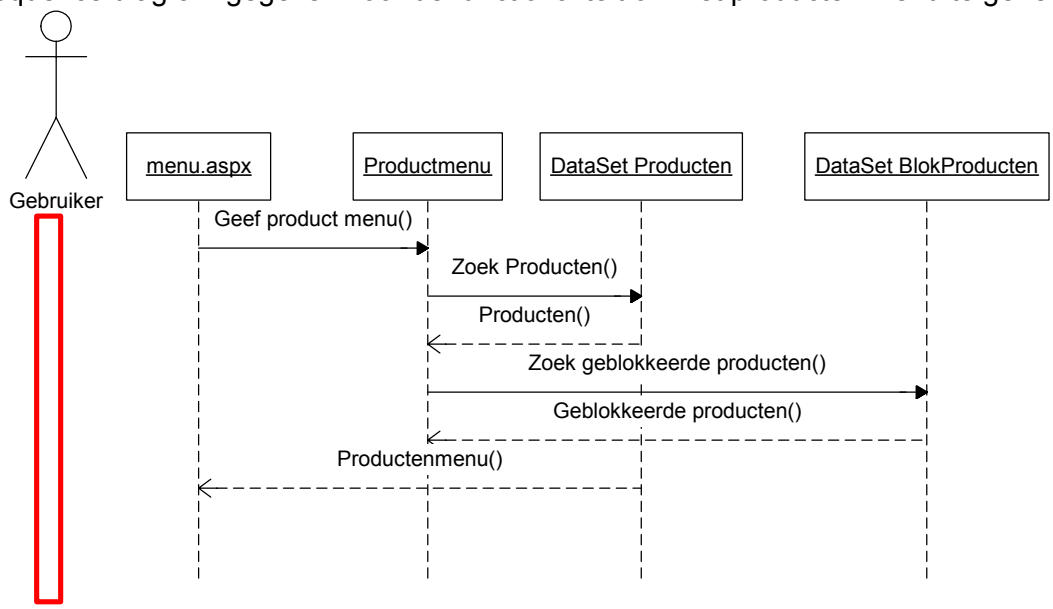
Het resultaat van deze fase was een document: “pilotontwikkeling: pilot 1” dat als bijlage “D” is opgenomen bij dit verslag. In dit hoofdstuk is de totstandkoming van pilot 1 beschreven.

7.2 Globaal functiestructuur

Dit onderdeel van pilot 1 had als doel de functionele inhoud van de pilot te bepalen. De in de definitiestudie besproken functionele architectuur (in het systeemconcept) is hier verfijnd.

7.2.1 Sequence diagrammen

Als eerste zijn er functionele procesmodellen gegeven voor de te ontwikkelen onderdelen. Hierbij is er gebruik gemaakt van sequence diagrammen. Er is gekozen voor het gebruik van deze diagrammen omdat deze op een relatief globaal niveau kunnen schetsen welke onderliggende functionaliteiten er in de onderdelen moeten worden verwerkt. Hieronder is een sequence diagram gegeven voor de functionaliteit om het producten menu te genereren.



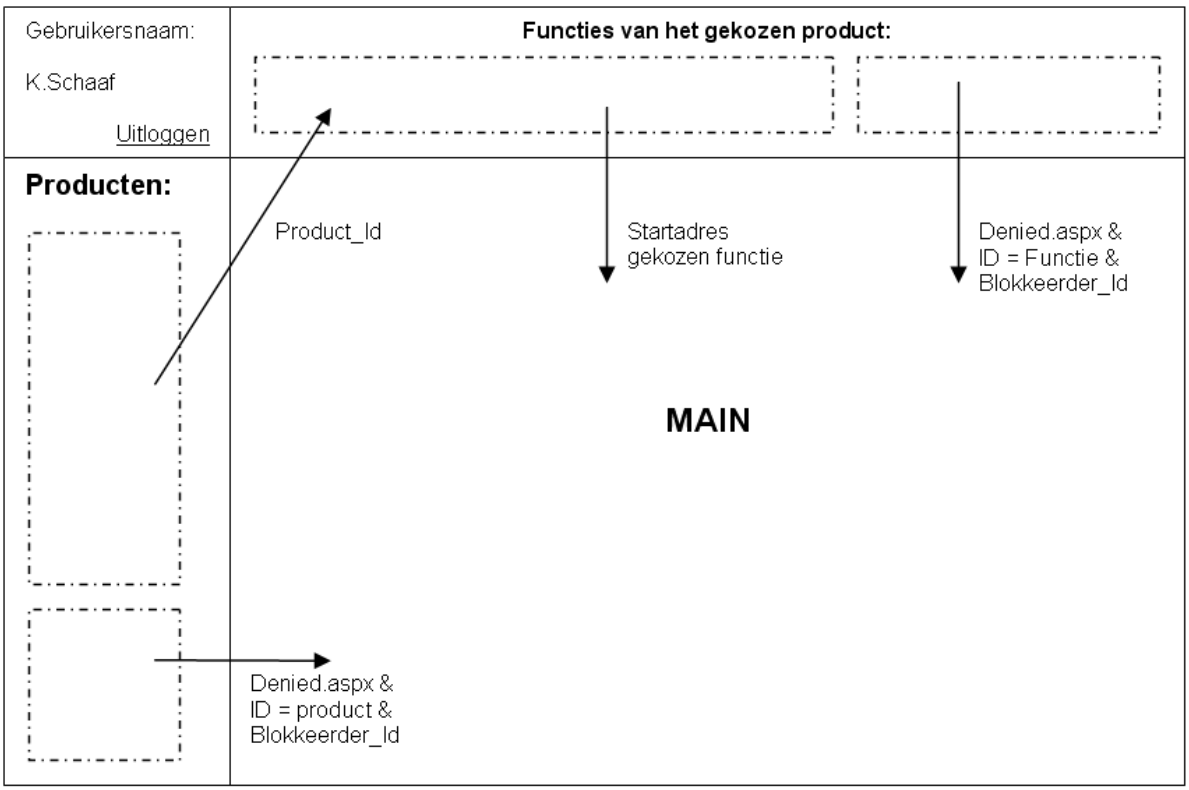
Figuur 22

Zodra een gebruiker is ingelogd op het beveiligingssysteem wordt er voor de gebruiker een persoonlijk producten menu gegenereerd met daarin alleen de producten waartoe hij toegang heeft en de producten die geblokkeerd zijn voor deze gebruiker. Zodra het productmenu wordt geladen worden de datasets “Producten” en “BlokProducten” uitgelezen. (Dit zijn twee datasets die zijn ingelezen tijdens het inloggen.) Met behulp van de data uit

deze datasets kan het menu worden gegenereerd en wordt het gegenereerde menu getoond aan de gebruiker.

7.2.2 Grafische structuur webapplicatie

Omdat er wordt gewerkt met persoonlijke menu's en submenu's dient er gebruik te worden gemaakt van navigatie mechanisme die gebruikerafhankelijk de menu's bepaald. Hieronder is een schematisch overzicht gegeven van de manier waarop er binnen de menu's genavigeerd wordt.

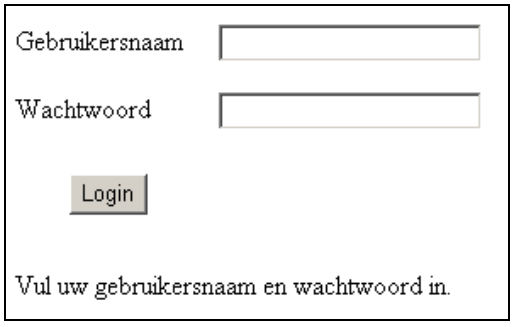


Figuur 23

Door middel van dit schema is er getracht een idee te geven aan de opdrachtgever over de manier waarop er met de applicatie kan worden gewerkt.

7.2.3 Prototype gebruikersinterface

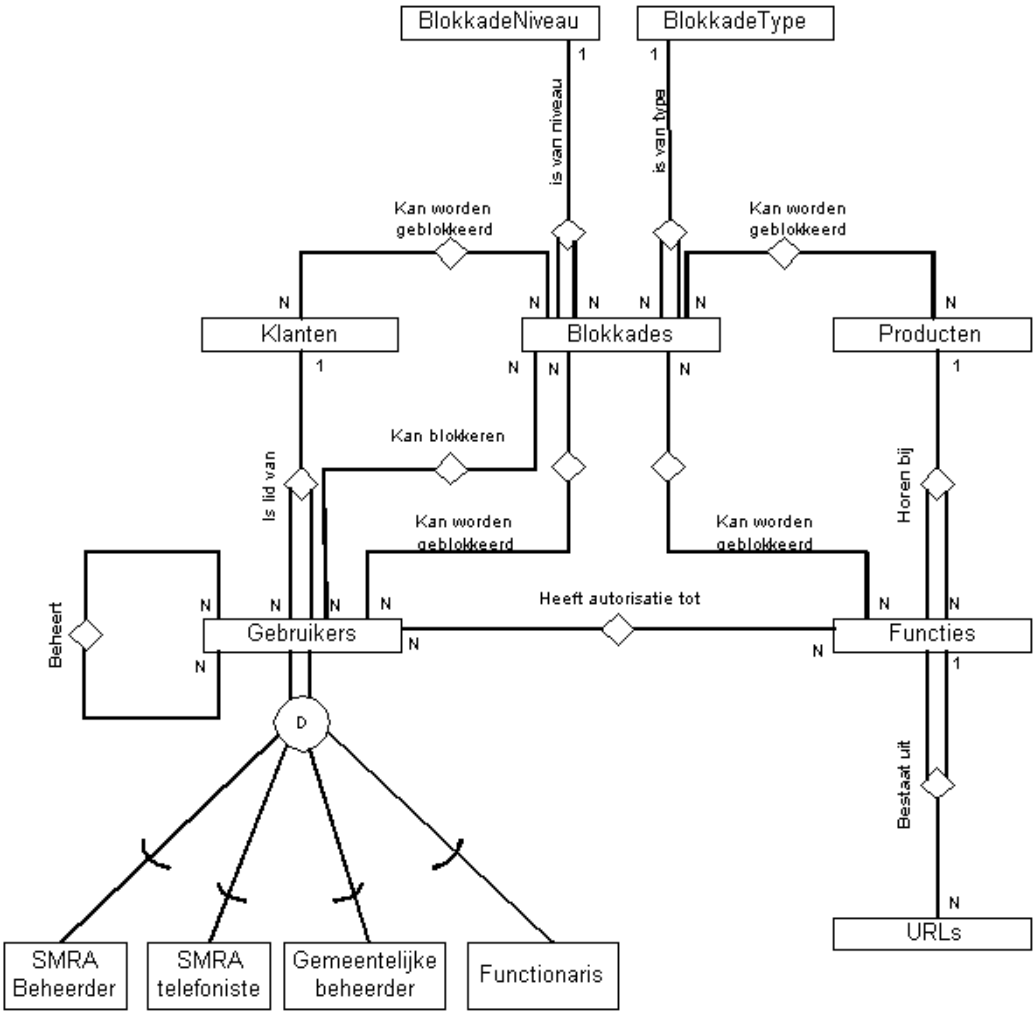
In de globale functiestructuur zijn ook prototypen van de gebruikersinterface gegeven. Dit had tot doel de opdrachtgever een idee te geven over de structuur en de gebruikersvriendelijkheid van de uiteindelijke pilot. In het volgende scherm is een prototype gegeven van de gebruikersinterface van het login scherm.



Figuur 24

7.2.4 Conceptueel gegevensmodel

Ook is het conceptueel gegevensmodel hier aan bod gekomen. Omdat er tijdens de definitiefase al een klassendiagram is ontwikkeld was het vrij eenvoudig om de gegevens te bepalen die in het ERD moesten worden verwerkt. Tijdens het opstellen van het ERD bleek ook dat er een extra attribuut blokkades nodig was om de blokkades in vast te leggen. Er is tijdens een vergadering besloten om als aparte entiteit op te nemen om zo flexibel mogelijk te blijven voor uitbreidingen en om het simpel te houden. Op een later tijdstip tijdens de pilot zijn er ook nog andere wijzigingen gedaan zoals het blokkadeniveau en het blokkadetype apart in een tabel op te nemen. Dit geeft de mogelijkheid om binnen het systeem extra blokkadetypen of niveaus aan te maken zonder dat dit wijzigingen vereist in de programmatuur. Hieronder is het ERD gegeven (Figuur 25).



Figuur 25

Naast het ERD is er ook een overzicht gemaakt van de tabellen met de bijbehorende relaties. In dit model is gemakkelijk te zien uit welke tabellen, attributen en relaties de database bestaat (dit model is alleen opgenomen in het pilotrapport). Het doel van dit model was het geven van een schematisch overzicht van de tabellen en hun velden om zo te kunnen functioneren als helpende hand bij het programmeren.

7.3 Globaal technische structuur

In dit onderdeel van het document zijn de technische aspecten gespecificeerd. Hierin is besproken: de fysieke allocatie, het technisch opslagmodel, het netwerkcommunicatiemodel, de externe interfaces en de configuratie van het beveiligingssysteem. In de fysieke allocatie worden de fysieke locaties gegeven van de gegevens en op welke manier de gegevens worden gebruikt. In het technisch opslagmodel wordt het ERD omgevormd naar een implementatiemodel. Dit model geeft aan hoe de database technisch is opgebouwd. (Dit model is alleen opgenomen in het pilotrapport)

Bij sommige tabellen was het nodig om constraints op te nemen. Constraints zijn beperkingen op tabellen die niet op relationeel niveau kunnen worden aangegeven. Deze constraints zijn opgenomen om inconsistente gegevens op database niveau af te dwingen. Voor de tabel blokkades is er bijvoorbeeld een constraint opgesteld waarbij wordt afgedwongen dat als er een blokkade wordt geplaatst van het type: productblokkade dat het product_Id niet leeg mag zijn en het functie_Id leeg moet zijn. Dit is gedaan door de volgende constraint op te nemen bij de tabel blokkades:

```
([Type_Id] = 1
and [Niveau_Id] is not null
and [Functie_Id] is not null
and [Product_Id] is null
```

Ook voor de andere blokkadetypen en hun relevante niveaus zijn de nodige constraints opgegeven bij de tabel blokkades.

Als Laatste zijn de configuratie- instellingen van het beveiligingssysteem besproken. Dit is een belangrijk onderdeel bij het opzetten van een ASP.NET webapplicatie omdat deze instellingen noodzakelijk zijn om de omgeving in te stellen voor de beveiligingsapplicatie. De configuratie bestanden binnen ASP.NET zijn geschreven in XML. Één van de instellingen die moest worden aangebracht is om gebruik te maken van forms authentication. Dit geeft aan dat de authentificatie niet door IIS wordt gedaan maar door forms authentication van ASP.NET. Dit is aangegeven door de volgende code in de web.config te plaatsen.

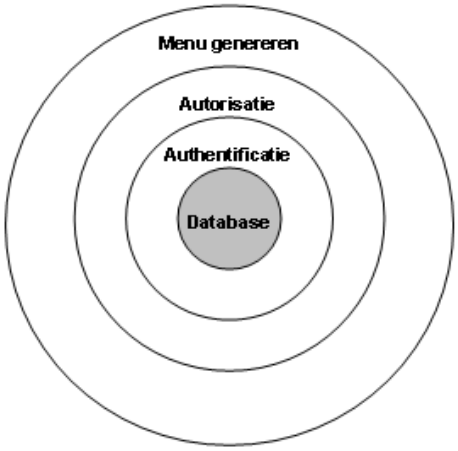
```
<authentication mode="Forms" >
        <forms loginUrl="LogIn.ASPx">
        </forms>
</authentication>
<authorization>
        <deny users="?" />
</authorization>
```

Alle gebruikers die niet bekend zijn binnen de webapplicatie worden doorgestuurd naar de LogIn.aspx pagina. Door de autorization tag te configureren als <deny users”?” /> worden alle onbekende gebruikers doorgestuurd naar de loginUrl om zich te authentificeren.

7.4 Pilot ontwikkelplan

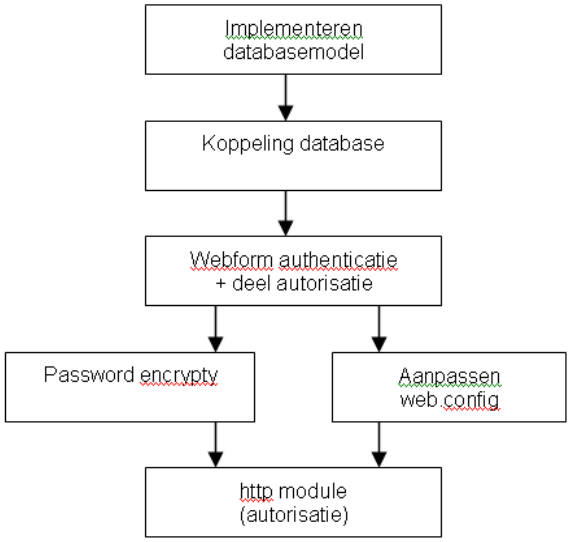
Het pilotplan is ontworpen op basis van de eerder gemaakte globale functionele en technische specificaties. Het pilotplan beschrijft hoe de pilot gaat worden gebouwd. De pilot is opgedeeld in pilotdelen en bouweenheden die achtereenvolgens dan wel parallel kunnen worden ontwikkeld. Tevens worden er voorzieningen getroffen voor het integreren en testen van de pilotdelen.

De pilot wordt opgedeeld in samenhangende componenten die afzonderlijk kunnen worden ontwikkeld, beoordeeld en getest. Binnen de applicatie bevinden zich onderdelen die van elkaar afhankelijk zijn. Om deze reden is er begonnen met de kern van de applicatie: de database. Hieronder is de afhankelijkheid van de onderdelen in een schema weergegeven



Figuur 26

Alle onderdelen in dit schema zijn apart besproken om hun afhankelijkheid te verklaren zodat hiermee rekening kan worden gehouden bij de ontwikkeling. De pilotdelen die hieruit naar voren zijn gekomen zijn hierna nogmaals opgedeeld in bouweenheden die afzonderlijk kunnen worden gebouwd. Aan de hand van de bouweenheden is er een gestructureerd model gerealiseerd waarin is aangegeven in welke volgorde de bouweenheden kunnen worden ontwikkeld. Hieronder is een deel van dit schema weergegeven.



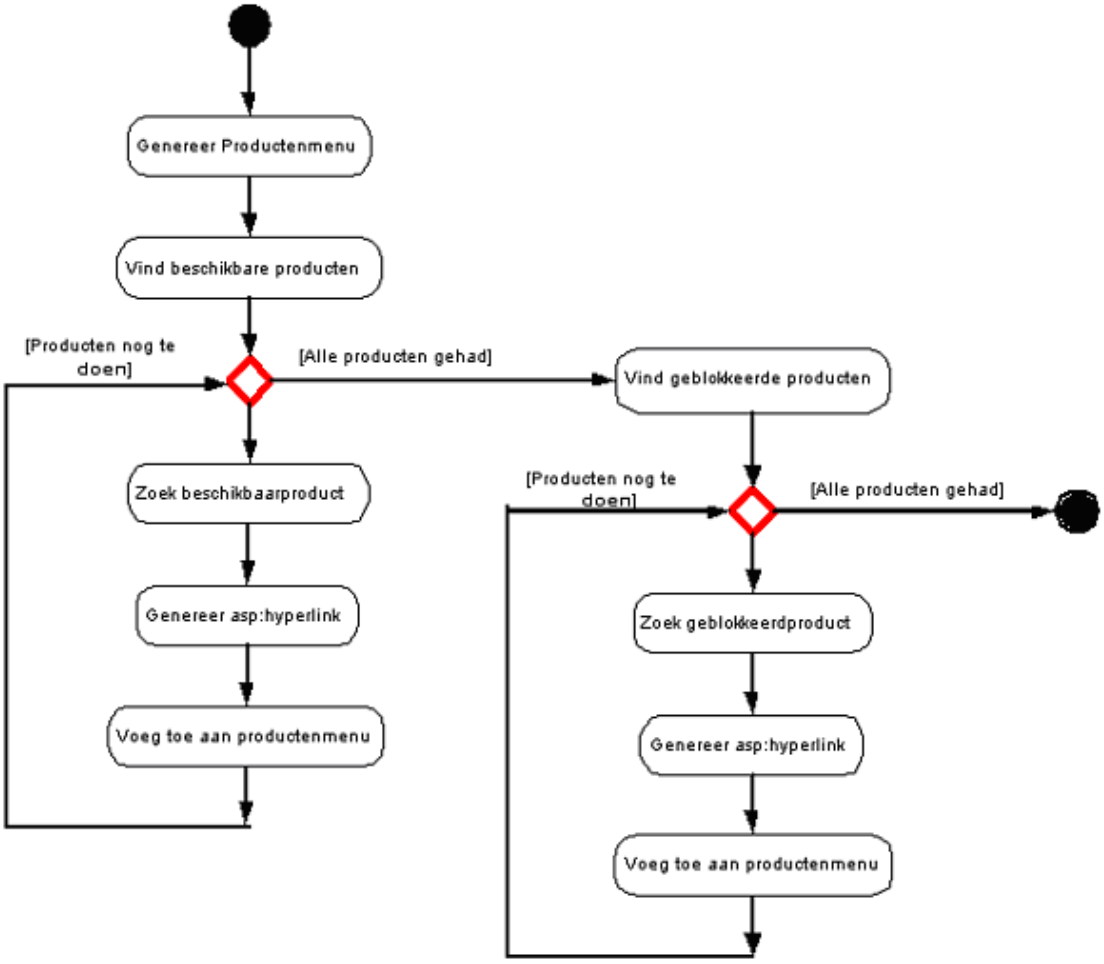
Figuur 27

In dit schema kan worden afgelezen dat er pas kan worden gestart aan de bouw van de authentificatie als de database is opgezet. De authentificatie is namelijk gebaseerd op de gebruikersnaam en het wachtwoord dat is opgeslagen in de database.
Uiteindelijk zal na het bouwen van de software, de software worden getest.

7.5 Ontwerp software bouweenheden

Bij het ontwerp van de software bouweenheden is de technische structuur van deze eenheden bepaald. Om dit te realiseren is er gebruik gemaakt van activiteiten diagrammen. Met behulp van deze activiteiten diagrammen zijn er procesgerichte beschrijvingen gegeven van de software bouweenheden.

Hieronder is een het activiteiten diagram weergegeven van de software bouweenheid om het productenmenu te genereren.



Figuur 28

7.6 Bouw software bouweenheden

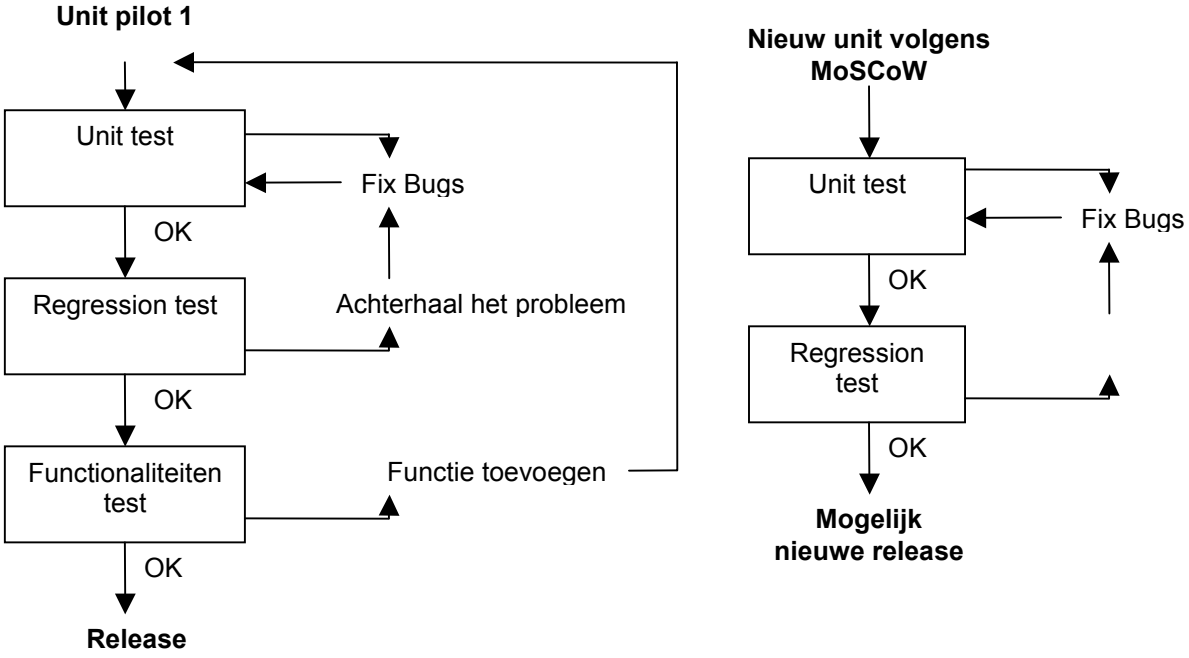
Omdat er voor het eerst door de student werd ontwikkeld in een object georiënteerde omgeving bracht dit vertraging met zich mee. De ontbrekende ervaring over het werken met Visual studio .NET 2003 als het werken met ASP.NET speelde hierbij ook een rol. De grafische interface van Visual studio vereiste vrij veel kennis over ASP.NET om hierin snel te kunnen ontwikkelen. Het ontwikkelen van de software bij de eerste pilot verliep dan ook zeer traag.

Er is begonnen met het maken van een webform (ASP.NET formulier) met een grafisch deel en een code deel, de “login” pagina. Op de login pagina is de authentificatie procedure gebouwd zoals dit is aangegeven in het voorgaande sequence diagram uit paragraaf 6.2. In het sequence diagram is onder andere aangegeven dat de bevoegde functies van een gebruiker worden ingelezen en opgeslagen in een dataset. Doordat er veel relaties liggen tussen een gebruiker en de blokkades werden de SQL queries om gegevens in te lezen ingewikkeld.

7.7 Beoordelen en testen

Uiteindelijk zijn de gebouwde software bouweenheden getest. Er is gebruik gemaakt van een door Microsoft voor geschreven teststrategie. Bij het testen van de webapplicatie is er gebruik gemaakt van unittesting, regressiontesting en een functionaliteitentest. Bij unittesting zijn alle units (functionaliteiten in de software) apart getest. De integratietest is gehouden na het samenvoegen van de units. Deze test had tot doel om te controleren of de units foutloos met elkaar konden samenwerken. De fouten die in de applicatie bleken te zitten zijn direct verholpen.

Hierna is er getest op functionaliteiten. In deze test is er simpelweg gekeken of de betreffende functionaliteiten uit de definitiestudie in het pilotdeel zijn verwerkt, afhankelijk van de MoSCoW prioriteit. De functionaliteiten die in een later stadium zijn toegevoegd zijn elke afzonderlijk getest met behulp van een unit test en er is gebruik gemaakt van een regressiontest. Hieronder is een schematische weergave gegeven van het verloop van het testen.



Figuur 29

Door de gebouwde software te testen worden de in het plan van aanpak opgestelde eisen controleerbaar en beoordeelbaar (correctheid, robuustheid, etc.).

7.8 Aanpassingen externe componenten

Binnen het uitvoeren van deze pilot waren ook een aantal aanpassingen gepland aan externe componenten. De klanten die worden bijgehouden in de SQL database van deze applicatie bevinden zich ook in de Adabas database. De bedoeling was dat deze gegevens gesynchroniseerd zou worden als er een gebruiker wordt aangemaakt. Om hiervoor een functionaliteit te ontwikkelen waren er gegeven nodig uit de Adabas database. Voor het aanspreken van de gegevens was het nodig een functionaliteit te ontwikkelen in de programmeertaal “Natural” welke zou worden gerealiseerd door een SMRA medewerker.
Tijdens het uitvoeren van het traject werd deze functionaliteit steeds uitgesteld omdat deze niet noodzakelijk was voor verdere voortgang van het traject en omdat de

medewerkers van de SMRA weinig tijd hadden om de functie te ontwikkelen. Uiteindelijk is deze functionaliteit niet bij het systeem opgenomen.

7.9 Afronding

Nadat de afzonderlijke hoofdstukken zijn geschreven, is het document: “ Pilot 1” afgerond. De opmaak van het document is gecontroleerd aan de hand van de checklist die als bijlage is opgenomen bij het plan van aanpak. Tijdens deze fase hebben zich een aantal wijzigingen voorgedaan die invloed hebben op onderdelen uit de definitiestudie. Omdat er gebruik wort gemaakt van de iteratiestrategie Big-Bang-Invoeren zijn deze wijzigingen doorgevoegd in de definitiestudie. In de definitiestudie is ook de evaluatie van de pilot beschreven.

8. Pilotontwikkeling, pilot 2

8.1 Inleiding

Na het afronden van de eerste pilot en het aanpassen van de definitiestudie is er gestart aan pilot 2. Tijdens pilot 2 is er gewerkt aan de beheersfunctionaliteiten van het beveiligingsysteem.

Omdat er bij het opstellen van het onderzoeksrapport en bij het realiseren van pilot 1 een behoorlijke vertraging is opgelopen is deze pilot niet geheel afgerond. Alle beheersfunctionaliteiten zijn in het systeem verwerkt behalve degene voor het beheren van de blokkades. De functionaliteiten die wel zijn ontwikkeld zijn dan ook ontworpen, gebouwd en getest. Het ontwikkelen van pilot 2 heeft ongeveer 3,5 week in beslag genomen en is daarbij niet volledig gerealiseerd.

Het resultaat van deze fase was een document: “pilotontwikkeling: pilot 2” dat als bijlage “E” is opgenomen bij dit verslag. In dit hoofdstuk is de totstandkoming van pilot 2 beschreven.

8.2 Globale functiestructuur

Bij het realiseren van de globale functiestructuur is eerst de functionele inhoud van deze pilot bepaald. Ook hier is net als bij de uitvoering van de vorige pilot, het systeemconcept uit de definitiestudie verfijnd.

8.2.1 tekstuele omschrijving

In tegenstelling tot de eerste pilot is er hier geen gebruik gemaakt van sequence diagrammen maar is er gebruik gemaakt van een tekstuele omschrijving van de functionele onderdelen van deze pilot. Dit omdat er in de modellen onvoldoende informatie kan worden meegegeven over de functionele aspecten van de onderdelen en een tekstuele omschrijving een uitkomst leek te zijn. Bij het opstellen van de globale functiestructuur zijn de beheersfunctionaliteiten onderverdeeld in verschillende groepen namelijk:

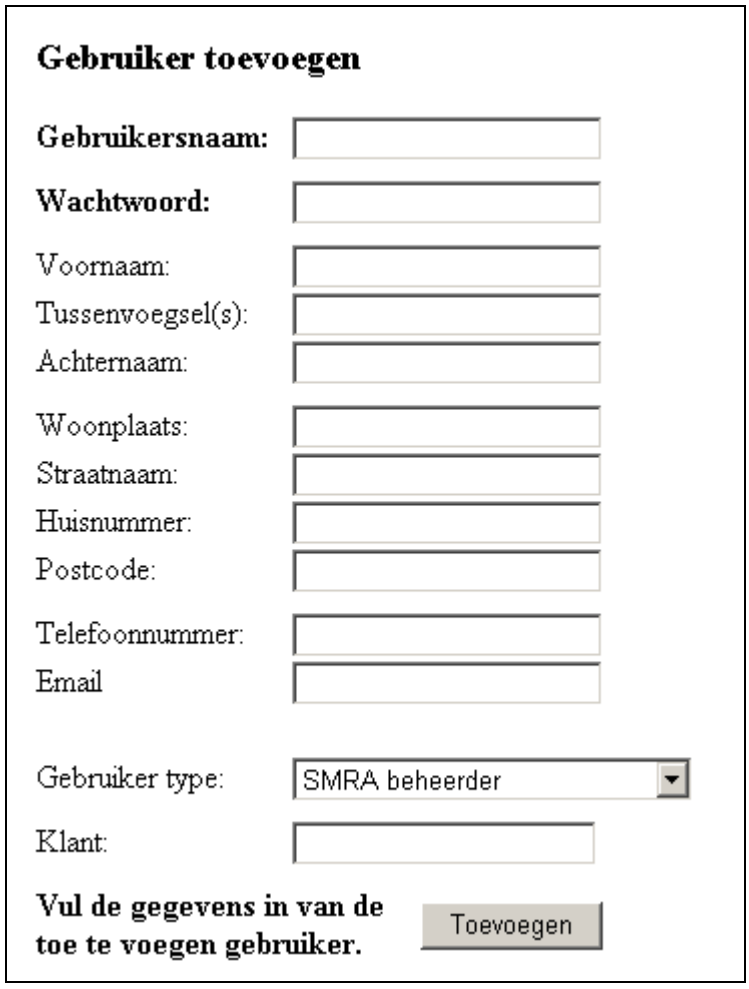
- Gebruikersbeheer
- Functiebeheer
- Blokkadebeheer
- Algemeen
- Controles

Een goed voorbeeld bij de groep gebruikers is dat veel functionaliteiten hetzelfde ogen maar per gebruikerstype anders reageren. Bij het aanmaken van een nieuwe gebruiker kijkt de functie naar het huidige gebruikerstype en wordt het gebruikerstype bepaald dat de gebruiker kan aanmaken en wordt de gemeente of gemeenten bepaald waar de gebruiker in kan worden aangemaakt. Een gemeentelijke beheerder mag namelijk alleen een functionaris aanmaken (gebruikertype 3) in zijn eigen gemeente terwijl SMRA beheerders deze niet mogen aanmaken en bij het aanmaken van een gemeentelijke beheerder de gemeente mogen opgeven.

8.2.2 Prototype gebruikersinterface

Omdat deze tijdens het realiseren van de voorgaande pilot een goed beeld bij zowel de opdrachtgevers als de student op heeft geleverd kon er een goede afstemming worden bereikt over de te realiseren onderdelen. Om deze reden is er binnen deze pilot ook gebruik gemaakt van prototypen van de gebruikersinterfaces.

Op de volgende pagina is een voorbeeld gegeven van een gebruikersinterface die is gebruikt tijdens het realiseren van pilot 2 (figuur 30).



Figuur 30

Dit scherm zou zichtbaar zijn als een SMRA beheerder de functionaliteit gebruiker aanmaken zou oproepen. De mogelijkheden om het gebruikerstype voor de aan te maken gebruiker en de klant is namelijk vrij op te geven door de SMRA beheerder.

8.3 Globaal technische structuur

De globaal technische structuur passend bij deze pilot is grotendeels al gegeven in pilot 1. Dit omdat de functies worden geplaatst onder het beveiligingssysteem dat is gerealiseerd in pilot 1. Er moet echter wel rekening worden gehouden met de fysieke allocatie voor het plaatsen van de functies. Dit is namelijk bepaald in de http module die is ontwikkeld in de eerste pilot.
De http module controleert aanvragen en valideert of de aanvraag gaat om het opvragen van een functie. Om de http module in staat te stellen onderscheid te maken tussen de pagina's van het beveiligingssysteem en de functioneliteiten binnen het beveiligingssysteem dienen toegevoegde functionaliteiten in een specifieke map te worden geplaatst, namelijk:

http://www.smra.nl/beveiligingssysteem/Functies/

Omdat verwacht werd dat dit snel zou resulteren in een onoverzichtelijke hoeveelheid van pagina's binnen deze map is het mogelijk gemaakt om ook submappen op te geven.
Deze submappen kunnen met willekeurige namen worden aangemaakt en het systeem zal nog steeds herkennen dat er een aanvraag wordt gedaan naar een functie waarvoor bevoegdheid is vereist.

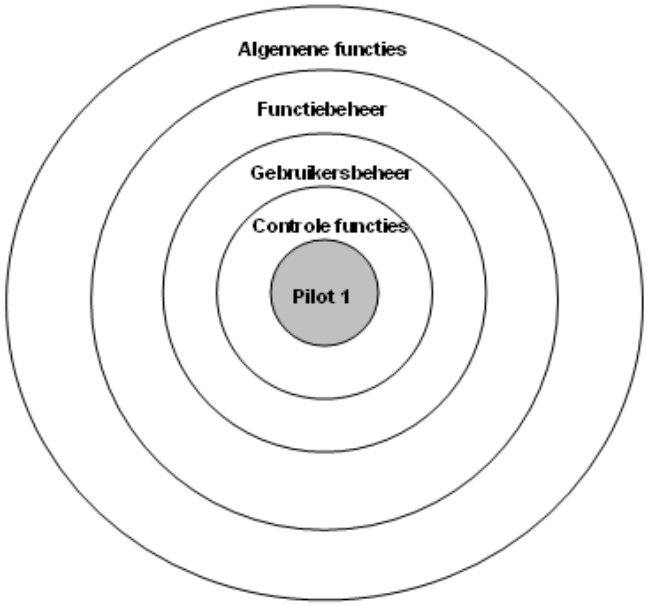
8.4 Pilot ontwikkelplan

Ook dit pilotplan is ontworpen op basis van de eerder gemaakte globale functiestructuur. Het opstellen van een pilotplan gaf tijdens het realiseren van de vorige pilot de mogelijkheid om de bouweenheden van de pilot gestructureerd te kunnen ontwikkelen. Om deze reden is er ook een pilotplan gerealiseerd voor de tweede pilot.

Ook hier worden de bouweenheden van de pilot bepaald en wordt er beschreven hoe de pilot zal worden gebouwd.

8.4.1 pilot delen

De pilot is opgedeeld in samenhangende componenten die afzonderlijk kunnen worden ontwikkeld, beoordeeld en getest. Binnen de applicatie bevinden zich onderdelen die afhankelijk zijn van de onderdelen uit pilot 1. Het was dus noodzakelijk dat pilotfase 1 volledig en succesvol was afgerond. Hieronder zijn de afhankelijkheden van de onderdelen uit pilot 2, in een schema weergegeven.



Figuur 31

Na pilot 1 staan de controle functies in de kern. In dit diagram wil dat in dit geval niet zeggen dat de controle functies eerst moeten worden gerealiseerd voordat er kan worden begonnen aan het gebruikersbeheer maar is er getracht aan te tonen dat de controle functies centraal moeten staan omdat ze door meerdere bouweenheden kunnen worden gebruikt.
Een goed voorbeeld is hiervan de controle of een gemeentelijke beheerder een gebruiker opvraagt uit zijn gemeente. Deze controle functie zal worden gebruikt bij het verwijderen van een gebruiker en zal worden gebruikt bij het wijzigen van een gebruiker.

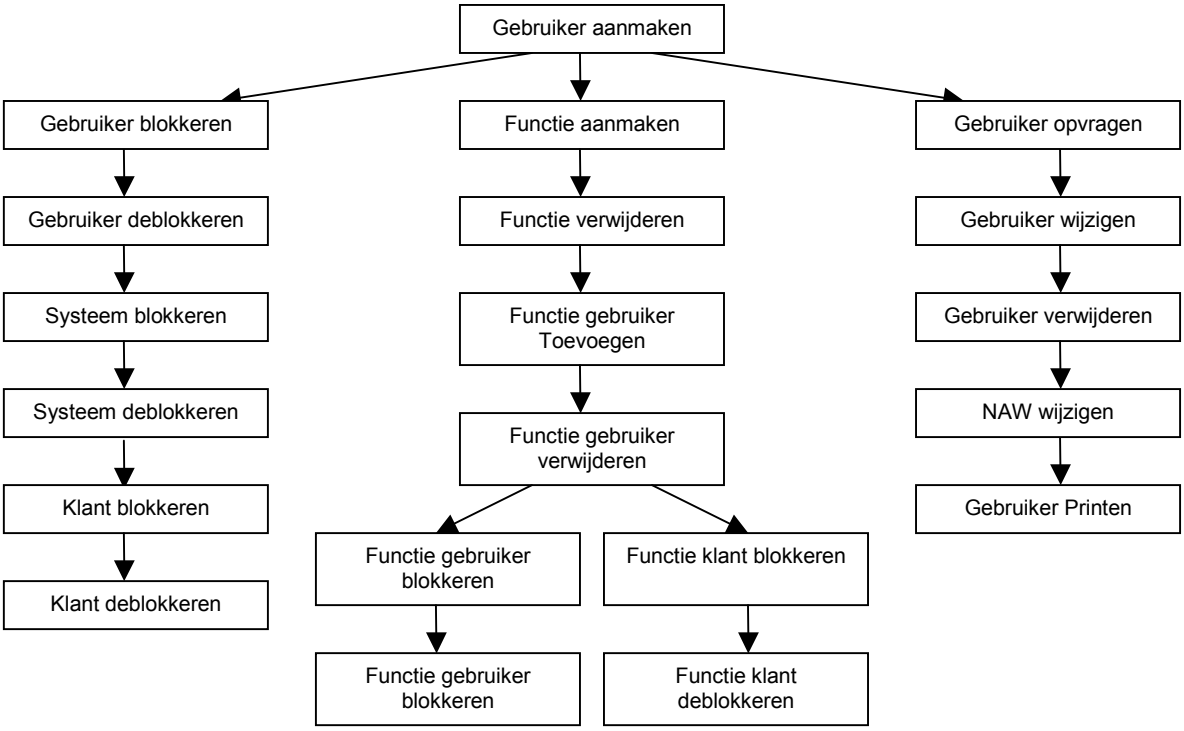
8.4.2 bouweenheden

Na het vaststellen van de pilotdelen zijn de bouweenheden per pilot bepaald. Dit is gedaan door een tekstuele omschrijving van de onderlinge functionaliteiten per bouweenheid. Als voorbeeld het pilotdeel gebruikersbeheer. Dit onderdeel is hier onderverdeeld in de functionliteiten

- GebruikerAanmaken;
- GebruikerWijzigen;
- GebruikerPrinten;
- GebruikerVerwijderen;
- GebruikerOpvragen.

8.4.3 Synchronisatie bouweenheden

Voor de bouweenheden is er een model gerealiseerd waarin de structuur van de ontwikkeling van de bouweenheden is gegeven. Dit model is hieronder weergegeven.



Figuur 32

Aan de hand van dit model zijn de bouweenheden uit pilot 2 ontwikkeld.

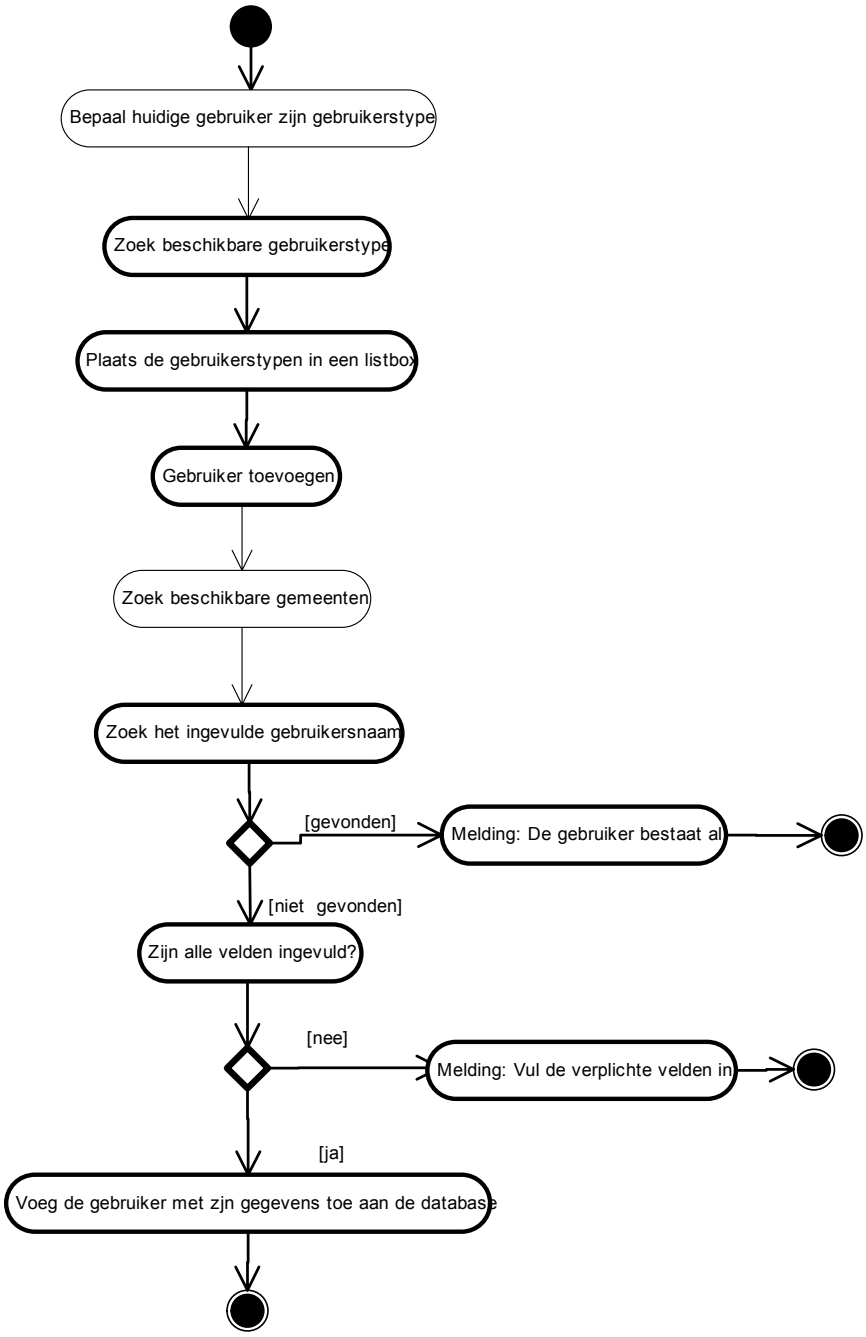
8.4.4 Testen

Ook hier is van te voren vastgesteld dat de gebouwde software bouweenheden worden getest en zal er gebruikt worden gemaakt van de door Microsoft voor geschreven teststrategie zoals bij pilot 1 (paragraaf 7.7).

8.5 Ontwerp software bouweenheden

Bij het ontwerp van de software bouweenheden is de technische structuur van deze eenheden bepaald. Net is bij de vorige pilot is er gebruik gemaakt van activiteiten diagrammen. Met behulp van deze activiteiten diagrammen zijn er procesgerichte beschrijvingen gegeven van de software bouweenheden.

De functionaliteiten uit deze pilot zijn een stuk ingewikkelder dan in de vorige pilot. Dit omdat er veel controles in de functionaliteiten zitten verwerkt. Op de volgende pagina is technische structuur van de functie om een gebruiker aan te maken in een UML activiteiten diagram aangegeven (figuur 33).



Figuur 33

Deze functie bepaald eerst het gebruikerstype van de gebruiker die de functie aanroept. Daarna worden de gebruikerstypen die hij mag aanmaken vastgesteld. Ook wordt aan de hand van het gebruikerstype bepaald in welke gemeente een gebruiker mag worden aangemaakt. Een gemeentelijke beheerder mag namelijk alleen een gebruiker aanmaken in zijn eigen gemeente.
Hiernaast wordt er ook gecontroleerd of de gebruikersnaam al in gebruik is en of alle verplichte velden zijn ingevuld.

8.6 Bouw software bouweenheden

De bouw van de functionaliteiten in pilot 2 vereisten andere kennis dan de functionaliteiten uit pilot 1. De functionaliteiten uit pilot 1 waren erg gericht op de programmeercode en de functionaliteiten uit pilot 2 zijn erg gericht op de ASP.NET controls die op web forms worden gebruikt. Wederom moest er worden uitgezocht hoe er met deze controls kon worden gewerkt en hoe deze kunnen worden toegepast in de programmeeromgeving Visual studio.NET 2003. Ook datagrids kwamen hierbij aan de orde. (Een datagrid is een zeer uitgebreide control die kan worden gebruikt om gegevens weer te geven in een overzicht)

De start van het ontwikkelen van de webforms verliep traag. Nadet er eenmaal een aantal webforms waren de overige webforms snel te realiseren. Wel bleek er tijdens de ontwikkelingen er vele mogelijkheden zijn om de functionaliteiten gebruikersvriendelijker te maken. Omdat de mogelijkheden tijdens het bouwen van de webforms steeds groter werden om de webforms gebruikersvriendelijker te maken en overzichtelijker, was het soms noodzakelijk dat eerder gebouwde functionaliteiten hierop moesten worden aangepast.

8.7 Beoordelen en testen

Na het bouwen van de software bouweenheden zijn er testspecificaties opgesteld voor de unittesten, regressietest en de funcionaliteitentest. 0Hieronder is een testset gegeven met de testspecificaties voor het testen van de functionaliteit om een gebruiker aan te maken.

Omschrijving	Verwacht resultaat	Resultaat
Nieuwe gebruiker aanmaken met alle gegevens ingevuld.	Er wordt een nieuwe gebruiker aangemaakt.	
Gebruiker aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	
Gebruiker aanmaken met alle verplichte velden ingevuld en alle niet verplichte velden niet ingevuld.	Er wordt een nieuwe gebruiker aangemaakt.	
Een gebruiker aanmaken met een al bestaande gebruikersnaam.	Er wordt een melding gegeven dat de gebruiker niet kan worden aangemaakt omdat de gebruikersnaam al in gebruik is.	
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	

Bij het uitvoeren van deze test zijn de resultaten vastgelegd in deze tabel. Indien er een test niet verloopt naar verwacht resultaat of de functionaliteit een bug bevat, is de functionaliteit aangepast. Naast het uitvoeren van de unittesten heeft er ook een regressietest plaatsgevonden en is er met behulp van een eerder opgestelde lijst bepaald of de pilot alle functionaliteiten bevat van pilot 2.

8.8 Afronding

Pilot 2 is niet volledig afgerond wegens gebrek aan tijd. Om deze reden zijn de wijzigingen die nodig zijn in de definitiestudie na het afronden van deze pilot ook niet meegenomen. Om dit document toch af te ronden voor de deadline van de afstudeeropdracht dit document tekstueel afgerond.

In de planning is aangegeven dat de opdracht is onderverdeeld in 3 pilotdelen. Pilot 3 is de laatste pilot van deze opdracht en bevat de functionaliteiten die volgens MoSCoW anders dan “Must haves” zijn. Omdat pilot 2 niet volledig is afgerond is er niet gestart aan het ontwikkelen van de derde pilot.

9. Procesevaluatie

9.1 Inleiding

In dit hoofdstuk zijn de evaluaties van de processen gegeven over de opgeleverde producten. Bij het evalueren van de processen is geëvalueerd of de juiste keuzen zijn gemaakt per proces en of het proces moeilijk of gemakkelijk uit te voeren was. Ook is er aangegeven of dit proces de volgende keer op dezelfde manier zal worden aangepakt.

9.2 Plan van aanpak

Bij het opstellen van het plan van aanpak is de ervaring gebruikt die is opgedaan bij de voorgaande projecten. Het eerste onderdeel dat is gerealiseerd in het plan van aanpak is het onderdeel opdrachtomschrijving. De opdrachtomschrijving was al grotendeels voor de start van het afstuderen gerealiseerd.

De ontwikkelmethode die voor de opdracht zou worden gebruikt was van te voren vastgesteld. De ontwikkelmethode IAD is besproken tijdens een presentatie bij het vak SW-07 en het iteratieve karakter van IAD leek goed te passen bij dit project. Tijdens het opstellen van het plan van aanpak is de IAD ontwikkelmethode bestudeerd en is geverifieerd of de IAD methode daadwerkelijk geschikt was voor de opdracht. Dit is gedaan met behulp van het boek:

Ttitel: IAD, het evolutionair ontwikkelen van informatiesystemen
ISBN: 90-395-0401-6
Auteur: R.J.H. Tolido
Uitgever: Academic Service

Het verschil tussen IAD en SDM is vrij groot. IAD vereist een andere instelling bij het ontwikkelen van software en IAD richt zich ook meer op de contacten tussen de ontwikkelaar en de gebruikers. Het kosten dan ook enige tijd om aan de instelling van IAD te wennen.

Na het onderzoeken van IAD zijn de projectrisico's opgesteld. Dit was een eenvoudig proces. Veel van deze risico's zijn standaard en zij te hergebruiken uit voorgaande projecten. Ook brengen de ontwikkelmethode IAD en de UML technieken de nodige risico's met zich mee omdat deze niet eerder zijn gebruikt bij het uitvoeren van een project. Bij een kort onderzoek naar deze onderdelen kunnen de risico's al snel worden bepaald. In overleg met de opdrachtgevers zijn de maatregelen vastgesteld bij de projectrisico's.

Om het traject van het project vast te stellen is er een planning gemaakt. Het realiseren van de planning was vrij eenvoudig omdat dit voornamelijk is gebaseerd op schattingen. Voor het realiseren van het overzicht is gebruik gemaakt van een "Gannt" model. Dit model is ook in de voorgaande projecten gebruikt en geeft een overzichtelijk overzicht van de activiteiten gepland tegenover de tijd.

De faciliteiten zijn bepaald door vast te stellen welke software en hardware apparatuur er benodigd is voor het uitvoeren van het project. Omdat er van te voren al duidelijk was dat er geprogrammeerd moest worden in ASP.NET met behulp van Visual studio.Net 2003 kon de nodige hardware en software vast worden gesteld door de minimale eisen op te zoeken van beide producten. Dit was zeer eenvoudig vast te stellen.

Bij elk project moet de kwaliteit worden gewaarborgd. Om deze reden is er een apart onderdeel "kwaliteitscontrole" in het plan van aanpak opgenomen.

9.3 Onderzoek Microsoft.NET

Het onderzoek naar Microsof.Net heeft voornamelijk gebruik gemaakt van informatie op het Internet en enkele boeken. Het onderzoek was moeilijk om uit te voeren omdat de .NET omgeving relatief nieuw is en de informatie vaak gericht is op bedrijfsinformatie. Ook is de Microsoft.Net omgeving een zeer uitgebreide omgeving en heeft het onderzoek in eerste instantie vier weken in beslag genomen. Maar tijdens verdere voortgang van het traject moesten er ook nog andere onderdelen worden onderzocht die nodig waren bij het bouwen van de verschillende bouweenheden.

De personeelsleden binnen de SMRA hebben vrijwel geen kennis over de Microsoft.NET omgeving. Dit zorgde ervoor dat alle informatie en kennis door de student zelfs moest worden verworven.

Bij het opstellen van het onderzoek naar Microsoft.Net is er als eerste een korte introductie gegeven over Microsoft.NET. Hierna is de basis van Microsoft.Net besproken en het .NET framework. Van het .NET framework zijn alleen de belangrijkste componenten onderzocht en beschreven in het onderzoeksrapport. Ook is er enig onderzoek gedaan naar XML-webservices. Dit bleek achteraf niet toepasbaar te zijn voor het project.

Hierna is het ontwikkelen van webapplicaties onderzocht in ASP.NET. Voordat er in ASP.NET geprogrammeerd kan worden is er veel kennis vereist. Er zit een vrij ingewikkelde structuur in de webpagina's zijn deze onderdelen onderzocht in het document omschreven.

Er is ook onderzoek gedaan naar ADO.NET. ADO.NET is de opvolger van ActiveX Data Objects en zorgt binnen het .Net framework voor de gegevenstoegang. Er is aan het begin een korte samenvatting gegeven over ADO.NET en hoe ADO.NET werkt. De overige informatie die is verwerkt in dit hoofdstuk komt voornamelijk vanaf de Microsoft website.

De beveiliging van webpagina's is een enorm uitgebreid deel van ASP.NET. Dit onderdeel heeft ongeveer een week gekost om te onderzoeken. Hierbij zijn de verschillende vormen van authentificatie en autorisatie onderzocht. Van de authentificatie vormen zijn schema's gerealiseerd en zijn deze in detail beschreven. Bij de autorisatie zijn worden de verschillende vormen van autorisatie besproken die door .Net worden ondersteund. Elke onderdeel dat mogelijk relevant kon zijn voor het project is hier onderzocht en omschreven.

Het te ontwikkelen systeem zal gebruik gaan maken van SSL. Is SSL wel zo veilig als dat er wordt gezegd. Ook dit is onderzocht door de werking van SSL te onderzoeken. SSL is een ingewikkeld concept en dat maakte dit onderdeel moeilijk om te documenteren.

In tegenstelling tot het oude ASP.NET wordt er binnen een ASP.NET applicatie gebruik gemaakt van configuratie bestanden. Om een applicatie te kunnen ontwikkelen in ASP.NET is kennis over deze configuratie bestanden vereist. Ook deze configuratie bestanden kennen vele instellingen en zorgde ervoor dat er veel tijd nodig was om de nodige configuratie instellengen te vinden.

In een later stadium tijdens het ontwikkelen van de beveiligingsapplicatie bleken de standaard autorisatie mechanismen binnen ASP.NET niet toepasbaar te zijn binnen het te ontwikkelen beveiligingssysteem. Om deze reden is er onderzoek gedaan naar een oplossing hiervoor. Hierbij leken http modulen of http handlers de oplossing te zijn. Een http module of handler werkt op een zeer laag niveau en vereist diverse kennis over Internet Information Services (IIS) en de ASP.NET omgeving.

Tot zo ver was het document een opsomming geworden van informatie. Om deze reden is er een apart hoofdstuk gerealiseerd op basis van het gehele document die een overzicht geeft

van de beveiliging van ASP.NET. In dit hoofdstuk worden de onderzochte onderdelen gerelateerd aan de SMRA en zijn er schema's gerealiseerd waarmee de beveiliging van het te ontwikkelen beveiligingssysteem kan worden bepaald. Doordat er een hoop afhankelijkheden zitten tussen de onderzochte onderdelen was het moeilijk om op dit punt al een conclusie te trekken over de te gebruiken authentificatie en autorisatie mechanismen.

9.4 Definitiestudie

Bij het opstellen van de definitiestudie is er begonnen met het realiseren van een plan van aanpak gericht op de definitiestudie. Dit als aanvulling op de globale planning uit de globale plan van aanpak. Tijdens het opstellen van dit plan van aanpak zijn de onderdelen onderzocht die in definitiestudie moesten worden verwerkt. Aan de hand hiervan is een overzicht ontwikkeld met een tijdsplanning voor de uitvoering van de definitiefase.

Nadat een pilot is afgerond is er in de definitiestudie een evaluatie over de pilot geschreven. Dit was een vrij eenvoudig proces en bleek achteraf geen toegevoegde waarde te bieden omdat met de gekozen iteratiestrategie de gehele definitiestudie aangepast of verfijnd wordt na de afronding van een pilot.

Voor het analyseren van de eisen en wensen is de "mindmapping" techniek gebruikt. Deze techniek is besproken tijdens een voorgaand project maar is echter nooit gebruikt. Dit is ondervonden als een gemakkelijke techniek waarmee snel en overzichtelijk de eisen en wensen van de opdrachtgevers kunnen worden bepaald en vastgelegd.
Ook het indelen van de prioriteiten door middel van de MoSCoW regels geeft duidelijk aan welke eisen de applicatie minimaal zal bezitten. Indien de "Must have" functionaliteiten gemakkelijk binnen de tijd realiseerbaar blijken te zijn, kunnen ook de eisen met een lagere prioriteit in het systeem worden verwerkt. Ook de MoSCoW regels werden beschouwd als een techniek waarmee vrijwel alle eisen kunnen worden vastgelegd op een zodanige manier dat dit voordelen kan bieden voor zowel de opdrachtgevers als de afstuderende student.

Bij het realiseren van het systeemconcept werd voor het eerst gebruik gemaakt van UML. Het systeemconcept is eerst gegeven door een actoren analyse, usecases, usecase diagrammen en een klassendiagram. Uit reactie van de opdrachtgevers bleek echter dat dit een onduidelijk systeemconcept had opgeleverd. Om deze reden is er een ook een tekstuele omschrijving van het systeemconcept toegevoegd.
Omdat er tijdens het opstellen van de eisen en wensen al veel functionele wensen (meer dan 50) waren vastgelegd zorgde dit voor een grote hoeveelheid werk om de use-cases te realiseren en zorgde dit ook voor een onoverzichtelijk use-case-diagram. De use-cases zijn ondervonden als een zeer omslachtige manier voor het omschrijven van het systeemconcept. De use-case-diagrammen zijn later uitgesplitst per actor. Dit zorgde voor een overzichtelijk model maar er kon niet in worden aangegeven welke functies "synoniem" zouden zijn. Om deze reden is er een apart onderdeel opgenomen voor het aggregeren van de functies. Dit was vrij moeilijk om op functioneel niveau vast te stellen en zou zich nog moeten bewijzen tijdens de pilotontwikkeling.

Hierna is de technische structuur omschreven. Voor het bepalen van de technische infrastructuur is "interview" gesprek geweest met de systeembeheerder. Het resultaat van dit gesprek is vastgelegd in de technische architectuur.

Het maken van het pilotontwikkelplan was een leerzaam proces. Het werken met pilots was nieuw voor de opdrachtgevers en de student. Het opdelen van de functionaliteiten in pilots en bouweenheden bleek een goed overzicht op te leveren van de uit te voeren activiteiten tijdens de bouw van het project. Ook het onderzoeken of bouweenheden synchroon konden worden uitgevoerd, was leerzaam. Zo worden de afhankelijkheden van de functionaliteiten

aangegeven in een overzicht en kan de structuur van de pilotontwikkeling eenvoudig worden gerealiseerd.

9.5 Pilot 1

Bij het realiseren van de eerste pilot werd gestart met het opstellen van een globale functiestructuur. Dit is gedaan door middel van sequencediagrammen waarin het gedrag van het systeem werd weergegeven op functioneel niveau. De basis voor deze sequence diagrammen waren de betreffende use-cases uit de definitiestudie.
Een sequence diagram is eenvoudig op te stellenna het doornemen van het hoofdstuk over sequence diagrammen uit het boek: *Warmer, J., en Kleppe A., Praktisch UML, Addison Wesley, 2e editie, 2001.*

De grafische structuur van de webapplicatie was eenvoudig te realiseren omdat deze structuur door de student tijdens een voorgaand project al was gebruikt. Helaas werd verder in het project duidelijk dat er hiervoor webcontrols beschikbaar zijn.

Het conceptueel gegevensmodel was eenvoudig met een ERD te realiseren omdat er in de definitiestudie een klassendiagram was ontwikkeld. In het klassendiagram waren de meeste entiteiten, attributen en relaties bijna direct af te lezen. Weliswaar zijn er binnen het ERD nog een hoop aanpassingen gedaan om het model geschikt te maken voor het beveiligingssysteem. Dit was een vrij moeilijke stap omdat de problemen pas werden ontdekt toen ze nodig waren. Dit vereiste vaak aanpassingen aan onderdelen waarvan eerder gerealiseerde bouweenheden afhankelijk waren. Zelfs tijdens het uitvoeren van pilot 2 zijn er nog veranderingen geweest aan het ERD.

Het pilotontwikkelplan was volledig nieuw voor de student en er was enige moeite voor nodig om op dit niveau al te kunnen bepalen welke functionaliteiten van elkaar afhankelijk zijn. Ook was het moeilijk om te bepalen welke functionaliteiten er ontwikkeld moesten worden en welke functionaliteiten er ter beschikking werden gesteld door de base class library.

Verder was het noodzakelijk dat er een aantal configuratie instellingen werden verricht voordat de software bouweenheden konden worden gebouwd. Omdat de configuratie instellingen waren onderzocht tijdens het onderzoeksrapport naar Microsoft.Net konden deze gemakkelijk worden achterhaald.

De software bouweenheden zijn ontworpen met behulp van UML activiteiten diagrammen. Ook deze diagrammen waren nieuw voor de opdrachtgevers en de student. Bij het gebruik van deze activiteitendiagrammen bleken deze erg te lijken op de bekende “Flowcharts”. Het was vrij gemakkelijk om UML diagrammen te lezen maar deze zijn niet ondervonden als een geschikte tool voor het ontwerpen van software bouweenheden. Dit kwam vooral doordat er slecht in deze diagrammen kan worden aangegeven met welke parameters de functies werken.

Het bouwen van het beveiligingsdeel van de applicatie was het meest ingewikkelde onderdeel van deze pilot. Er werd voor het eerst ontwikkeld in ASP.NET en een groot deel van de te ontwikkelen functionaliteiten zaten op een zeer laag niveau binnen .NET. Aan het bouwen van de bouweenheden in pilot 1 is dan ook 2,5 week meer tijd besteed dan was gepland.

In het eerder opgestelde pilotplan was de structuur voor het testen vastgesteld. Bij het onderdeel testspecificaties zijn er simpelweg testsets opgesteld voor de ontwikkelde bouweenheden. Deze testsets waren eenvoudig te bepalen.
Na het opzetten van de testsets zijn deze uitgevoerd en zijn de resultaten vastgelegd. Bij het uitvoeren van de testsets werden er enkele “fouten” ontdekt die relevant waren voor

meerdere functionaliteiten. Het verhelpen van deze fouten heeft dan ook de meeste tijd in beslag genomen bij het onderdeel testen.

9.6 Pilot 2

In tegenstelling tot de eerste pilot is hier niet gewerkt met sequencediagrammen met een tekstuele omschrijving voor het realiseren van de globale functiestructuur. Dit heeft veel minder tijd in beslag genomen dan dat er sequencediagrammen zouden zijn ontwikkeld en dit leverde alsnog een duidelijke omschrijving op van de functionele structuur van deze pilot.

Het opstellen van de technische structuur was vrijwel overbodig omdat de technische structuur van de applicatie was gerealiseerd tijdens het uitvoeren van de eerste pilot. Wel is er aangegeven hoe het systeem kan herkennen dat aanvragen zijn gericht op een functionaliteit waarvoor bevoegdheid nodig is.

Het bepalen van de pilotdelen en de bouweenheden was voor de tweede pilot zeer eenvoudig. De pilotdelen waren reeds vastgesteld in de opdrachtomschrijving omdat dit deel zich zou richten op de beheersfunctionaliteiten van het systeem. In de definitiestudie waren alle beheersfunctionaliteiten vastgelegd als taken van de gebruikers. Deze waren zo goed als 1 op 1 over te nemen.

Het ontwerpen van de software bouweenheden is in deze pilot ook gedaan aan de hand van UML activiteitendiagrammen. Tijdens het ontwerpen van de functionaliteiten binnen deze pilot waren deze diagrammen beter toepasbaar omdat van de functionaliteiten gemakkelijk kon worden bepaald wat voor activiteiten ze moesten verrichten en omdat de kennis om webforms te ontwikkelen in .NET reeds was opgedaan.

Omdat de deadline in zicht was gekomen is er besloten bepaalde functionaliteiten achterwege te laten de documentatie tot op het laatste punt af te ronden. De testspecificaties die voor deze bouweenheden moesten worden gerealiseerd zouden zeer uitgebreid zijn geweest (wegens de controles op invoervelden, etc.) en zijn daarom simpel gehouden.

De testen zijn uitgevoerd en de resultaten hiervan zijn vastgelegd in het pilotrapport. Bij het testen van deze functionaliteiten werden geen fouten aangetroffen en kon hierdoor snel worden afgerond.

10. Product evaluatie

10.1 Inleiding

In dit onderdeel van het document worden de producten opdrachtomschrijving, plan van aanpak, onderzoek Microsoft.NET, definitiestudie en de pilot's geëvalueerd. De evaluatie richt zich in dit onderdeel op de inhoud van de producten.

10.2 Plan van aanpak

Voor de opdracht was gekozen om gebruik te maken van IAD. Dit is een goede keuze geweest omdat het bij het project vaak voorkwam dat er achteraf wijzigingen plaats hebben gevonden. Ook is er gebruik gemaakt van de pilotworkshops die worden gebruikt binnen IAD. Deze hebben voor een goede relatie en gelijke visie over het systeem geleverd tijdens het uitvoeren van het project. Voordat er volgens IAD gewerkt kon worden moest eerst enig onderzoek worden verricht en aan het iteratieve karakter van IAD moest worden gewend.

Ook is er gekozen om gebruik te maken van UML. Er zijn veel verschillende UML technieken die gebruikt kunnen worden tijdens een project. UML is echter ondervonden als een veel te uitgebreid pakket en alleen de nodige technieken zijn gebruikt. Veel van deze technieken zijn bij dit project als omslachtig ervaren. Bij het volgende project zullen eerst software ontwikkeltechnieken worden onderzocht voordat er wordt gekozen voor UML.

Het vaststellen van de risico's heeft een goede invloed gehad op de relatie tussen de opdrachtgevers en de afstuderende student. Door de risico's vast te leggen en een overeenkomt te bereiken over de maatregelen die worden genomen bij het optreden van een risico worden onenigheden uitgesloten.
Een mogelijke maatregel die niet was opgenomen in het document was de mogelijkheid tot uitstel van de opdracht. Deze maatregel is tijdens dit project genomen wegens invloed van persoonlijke omstandigheden en dit risico had opgenomen moeten zijn in de risicoanalyse.

Er is een zeer globale planning gemaakt tijdens het "plan van aanpak" en tijdens de definitiestudie is deze uitgebreid voor de definitiestudie en de pilotontwikkeling. Door aparte planningen te maken in de definitiestudie kan er een preciezere planning worden gemaakt. Dit omdat er bij de realisatie van het plan van aanpak er nog niet veel kennis was opgedaan over een definitiestudie volgens IAD.

Om de opmaak van de documenten consistent en overzichtelijk te houden is er een checklist opgezet. Deze checklist is als bijlage opgenomen bij het plan van aanpak en heeft onnodige fouten in opmaak aangetoond zodat deze verholpen konden worden. De opmaak van dit rapport voldoet dan ook aan de eisen die daarvoor zijn gesteld.

10.3 Onderzoek Microsoft.NET

Het doel van het schrijven van het rapport over Microsoft.NET was om de opdrachtgevers van de SMRA kennis op te laten doen van .NET en de beveiligingsmogelijkheden hiervan. Op basis van de onderzochte beveiligingsmogelijkheden zou er een beveiligingssysteem kunnen worden ontwikkeld die geschikt zou zijn voor toepassing binnen de SMRA.
Met het opstellen van dit rapport zijn onder andere de volgende doelen bereikt:

- Zowel de student als de opdrachtgevers hebben een indruk gekregen van werking van .NET;
- Het rapport geeft een overzicht van de authentificatie onderdelen van .NET;
- Het rapport geeft een overzicht van de autorisatie onderdelen van .NET;
- Het rapport vormt een basis voor het ontwikkelen van webapplicaties in ASP.NET (onder andere de structuur van ASP.NET pagina's is gegeven);
- Het rapport geeft een overzicht van de beveiligingsmogelijkheden zoals deze zouden kunnen worden gebruikt binnen de SMRA.

De opmaak van het rapport voldoet aan de eisen die daarvoor zijn gesteld in het plan van aanpak.

10.4 Definitiestudie

Het rapport definitiestudie bevat de resultaten van de activiteiten die zijn uitgevoerd tijdens de fase definitiestudie. Het doel van dit rapport was om de eisen en wensen van het te ontwikkelen systeem vast te bepalen en een basis te vormen voor de pilotontwikkeling.
Met het opstellen van het rapport definitiestudie zijn onder andere de volgende doelen bereikt:

- Er is een overzicht gerealiseerd met de eisen en wensen van het systeem;
- Er is een systeemconcept ontwikkeld dat als basis heeft gefungeerd voor het ontwerp in de pilotrapporten;
- Er is een pilotontwikkelplan gemaakt dat is gebruikt als uitgangspunt voor de indeling van de pilotontwikkelingsfase;
- De technische structuur binnen de SMRA is vastgesteld zodat er kon worden vastgesteld dat het systeem binnen de huidige technische structuur van de SMRA kan worden toegepast.

De opmaak van het rapport voldoet aan de eisen die daarvoor zijn gesteld in het plan van aanpak.

10.5 Pilot 1

Dit pilotrapport had als doel de kernfunctionaliteiten van het beveiligingssysteem te ontwikkelen. Dit doel is bereikt bij het realiseren van Pilot 1. De onderdelen die hiervoor nodig zijn geweest zijn:

- Met behulp van prototypen van de gebruikersinterfaces is er een afstemming bereikt tussen de student en de opdrachtgevers over de te realiseren functionaliteiten en hun structuur;
- Een gegevensmodel en het implementatiemodel voor het opzetten van de database;
- De grafische structuur van de webapplicatie is vastgelegd;
- De nodige configuratie instellingen zijn bepaald beschreven;
- Er is een pilotontwikkelplan opgesteld waarin de bouw van de pilot is gestructureerd.
- Volgens de structuur van het pilotontwikkelplan zijn de bouweenheden ontworpen met behulp van UML activiteitendiagrammen;
- Er zijn testspecificaties opgesteld waarmee de gebouwde bouweenheden zijn getest en beoordeeld.

De opmaak van het rapport voldoet aan de eisen die daarvoor zijn gesteld in het plan van aanpak.

10.6 Pilot 2

Dit pilotrapport had als doel de beheersfunctionaliteiten van het beveiligingssysteem te ontwikkelen. Dit doel is bereikt bij het realiseren van Pilot 2. De onderdelen die hiervoor nodig zijn geweest zijn:

- Met behulp van prototypen van de gebruikersinterfaces is er een afstemming bereikt tussen de student en de opdrachtgevers over de te realiseren functionaliteiten en hun structuur;
- De functionaliteiten van pilot 2 zijn vastgesteld;
- Er is een pilotontwikkelplan opgesteld waarin de bouw van de pilot is gestructureerd.
- Volgens de structuur van het pilotontwikkelplan zijn de bouweenheden ontworpen met behulp van UML activiteitendiagrammen;
- Er zijn testspecificaties opgesteld waarmee de gebouwde bouweenheden zijn getest en beoordeeld.

Pilot 2 voldoet niet aan alle eisen en wensen zoals die zijn vastgelegd in de definitiestudie. De functionaliteiten voor het beheren van de blokkades ontbreken. Deze functies zijn zowel niet ontworpen als gebouwd.

De opmaak van het rapport voldoet aan de eisen die daarvoor zijn gesteld in het plan van aanpak.

10. 7 Beveiligingssysteem

Zoals al eerder is aangegeven voldoet het beveiligingssysteem niet aan alle eisen die tijdens de definitiestudie zijn opgesteld. De functionaliteiten die zijn gerealiseerd kunnen worden gebruikt. De beheersfunctionaliteiten om blokkades te plaatsen zijn echter niet bruikbaar. Het beveiligingssysteem biedt echter wel al de ondersteuning om blokkades te plaatsen maar deze kunnen nog niet op een grafische manier binnen de applicatie worden beheerd.

Geraadpleegde literatuur

Hieronder is een overzicht opgenomen van de geraadpleegde literatuur. Deze literatuurlijst heeft betrekking op dit document. Voor een overzicht van de bronnen die zijn geraadpleegd tijdens het onderzoek naar .NET wordt verwezen naar het onderdeel “Geraadpleegde literatuur” achterin het Evaluatierapport “Onderzoek Microsoft.Net” dat als externe bijlage is opgenomen bij dit verslag.

- Warmer, J., en Kleppe A., Praktisch UML, Addison Wesley, 2e editie, 2001.
- Ghezzi, C., e.a., Fundamentals of Software Engineering, Prentice-Hall, januari 1991;
- Reader AV-03 Haagse Hogeschool, nr. 908: Rapportagetechniek;
- Tolido, R.J.H., IAD - Het evolutionair ontwikkelen van informatiesystemen, Academic Service, 2e dr., 1998;
- Walther, Stephen, ASP.NET Uneleashed, SAMS, 1e editie, mei 2003
- Online documentatie over ASP.NET en Microsoft .NET;
  Microsoft .NET : http://www.msdn.microsoft.com
  ASP.NET : http://www.msdn.microsoft.com/asp.net/
  .NET Framework : http://www.msdn.microsoft.com/netframework/

Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Documenten**
  Plan van aanpak
  Onderzoek Microsoft.Net
  Definitiestudie
  Pilotontwikkeling: pilot 1
  Pilotontwikkeling: pilot 2

- **Auteur**
  Mark van Houten (20015081)

- **Datum**
  26 juli 2004

- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag

- **Opleidingsvariant**
  MBIT

- **Periode**
  November 2003 – april 2004

- **Opdrachtgever**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft

- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Voorwoord

Inhoud

Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Document**
  Plan van aanpak
- **Auteur**
  Mark van Houten (20015081)
- **Datum**
  26 juli 2004
- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag
- **Opleidingsvariant**
  MBIT
- **Periode**
  November 2003 – april 2004
- **Opdrachtgevers**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft
- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Voorwoord

Voor u ligt het rapport plan van aanpak ten behoeve van een web based beveiligingssysteem met functietoegankelijkheid- en gebruikersbeheer voor de SMRA. Dit rapport is tot stand gekomen in overleg met dhr R. Mehlkopf en dhr K. Schaaf.

In het plan van aanpak is de opdracht omschreven en is er aangegeven hoe het project zal worden aangepakt.

Delft, SMRA, 2 februari 2004

Inhoudsopgave

1. Inleiding

Dit rapport plan van aanpak is ontwikkeld om de aanpak vast te stellen voor dit project. In dit rapport is de opdracht beschreven en zijn de methoden en technieken bepaald die gebruikt worden om het project uit te voeren.

Het rapport plan van aanpak is de eerste stap voor het ontwikkelen van het beveiligingssysteem. In dit rapport zijn onder andere standaarden en richtlijnen gegeven, projectrisico´s opgesteld, de benodigde faciliteiten bepaald en zijn de kwaliteitseisen opgesteld.

In het rapport bevindt zich ook een planning. Deze planning geeft de activiteiten weer tegen over de tijd en geeft aan op welk tijdstip er een product zal worden opgeleverd.

Aan het einde van het rapport is gegeven waaruit de acceptatietest zal bestaan en hoe deze test zal verlopen.

2. Opdrachtomschrijving

2.1 Inleiding

In dit onderdeel van het rapport is de opdracht en de organisatorische omgeving waarin de opdracht wordt uitgevoerd, omschreven. Tijdens het beschrijven van de opdracht worden de contactpersonen besproken, de probleemstelling(en), de doelstelling(en) en de op te leveren producten worden vastgesteld.

2.2 Contactpersonen SMRA

Dhr. Jelke Brinksma is hoofd van de afdeling software ontwikkeling. Dit is de contractpersoon voor personele zaken.
Dhr. Roel Mehlkopf is begeleider van het project en kan worden gezien als ondersteuner bij het technisch ontwerp.
Dhr. Kees Schaaf is contactpersoon van de SMRA en kan worden gezien als ondersteuner bij het functioneel ontwerp.

2.3 organisatorische omgeving, kader, historie

De SMRA is een orgaan van de Nederlandse Hervormde Kerk en de afkorting staat voor Stichting Mechanisatie, Registratie en Administratie. De centrale organisatie bestaat sinds 1967 en is begonnen met een centraal ledenbestand waarin de lokale kerkledenregistratie van de bijna 1400 hervormde gemeenten wordt bijgehouden. Inmiddels geeft de SMRA operationele ondersteuning aan andere kerken via de Stichting Interkerkelijke Ledenadministratie (SILA). De SMRA is een gespecialiseerde kennisorganisatie met circa 60 medewerkers.

De SMRA beheert intussen een centraal ledenbestand met circa zeven miljoen personen. De SMRA heeft na de belastingsdienst, de grootste database met persoonsgegevens van Nederland. Deze gegevens worden bijgehouden voor de SILA en de bijbehorende administratie wordt de koppeladministratie genoemd. Van elke persoon worden de burgerlijke gegevens (waaronder NAW) opgeslagen samen met een koppeling aan een lokale kerk van een kerkgenootschap. De burgerlijke gegevens worden aangeleverd door de overheid op grond van een “SILA-stip”. Een SILA-stip geeft in de administratie van de overheid aan dat indien er burgerlijke gegevens van een persoon met deze stip worden gewijzigd, deze wijzigingen moeten worden doorgegeven aan de SILA.

Voor 2,6 miljoen van deze personen wordt er een kerkledenregistratie bijgehouden. Voor deze personen worden de burgerlijke gegevens opgeslagen samen met een aanvullende set kerkelijke gegevens.

De registratiegegevens worden onderhouden vanuit twee belangrijke bronnen: de overheid en de kerkelijke gemeenten. In werkelijkheid wordt het centrale ledenbestand onderhouden door meerdere systemen die gegevens bijhouden op een centrale locatie.

Vanuit de centrale registratie kunnen alle aangesloten kerkelijke gemeenten in verschillende vormen worden geïnformeerd over mutaties in de lokale ledenregistratie. Dit wordt naar keuze gedaan d.m.v. formulieren, registratiekaarten of geautomatiseerde mutatie-doorgifte over een modemverbinding naar een lokale PC-toepassing voor ledenregistratie. De lokale kerken leveren ook mutaties aan de SMRA door middel van formulieren of de modemverbinding.

Daarnaast levert de SMRA een groot aantal diensten op basis van de ledenadministratie zoals bijvoorbeeld fondswerving (kerkbalans) en distributie van kerkbladen.

2.4 Probleemstelling

Op dit moment worden de persoonsgegevens doorgevoerd in het centrale systeem door administratieve medewerkers van de SMRA. Voor het doorvoeren van deze wijzigingen wordt er gebruik gemaakt van verschillende systemen. De wijzigingen die moeten worden doorgevoerd worden geleverd door functionarissen van lokale gemeenten, met name ledenadministrateurs. Alleen zij bezitten de bevoegdheid voor het indienen van mutaties.

Het wijzigen van de gegevens wordt gedaan door middel van formulieren of geautomatiseerde mutatie-doorgifte over een modemverbinding vanuit de lokale PC-applicaties die gedeeltelijk geautomatiseerd verwerkt worden.

De kleine kerkelijke gemeenten werken met formulieren. Deze formulieren zijn bewerkelijk en dienen gecontroleerd te worden door de medewerkers van de SMRA. Deze formulieren worden verzonden per post en het duurt één tot enkele dagen voordat de SMRA de mutaties ontvangt. Elke wijziging die door middel van formulieren wordt toegezonden moet door een SMRA medewerker worden verwerkt in het centrale systeem.

De grotere kerkelijke gemeenten werken met PC-applicaties. Het uitvoeren van deze PC-applicaties heeft organisatorische consequenties. Het vereist een lokale gebruikersomgeving met de nodige hardware, software en kennis over de PC-applicatie. Hiernaast zijn er ook vrijwilligers nodig voor de verspreiding de van materialen zoals acceptgiro's.

Voor de kleinere kerkelijke gemeenten is het gebruik van een PC-applicatie geen goed alternatief. Dit vanwege de organisatorische gevolgen voor de organisatie. De SMRA wil een alternatief bieden dat ook geschikt is voor kleine kerkelijke gemeenten.

De SMRA heeft behoefte aan een versoepelde en versnelde manier voor het leveren van diensten. Met de huidige trend is er steeds meer vraag naar automatische mutatieverwerking zonder visuele controle en tussenkomst van SMRA medewerkers. Een andere wens van de SMRA is om in de toekomst de mogelijkheid om diensten aan te bieden over het Internet. Zo kan de SMRA ook aan de SMRA de kleine kerkelijke gemeenten de mogelijkheid bieden om zelf hun diensten uit te kunnen voeren.

2.5 Aanleiding van de opdracht

De SMRA wil applicaties gaan ontwikkelen die op Internet te gebruiken zijn. Omdat er wordt gewerkt met persoonlijke gegevens moeten de gegevens op een veilige manier over Internet worden verstuurd. De SMRA heeft, na de belastingsdienst, de grootste database met persoonsgegevens. Om deze reden speelt veiligheid een belangrijke rol.
Om de applicaties die werken met persoonsgegeven op Internet te kunnen plaatsen is er een beveiligingssysteem nodig. Dit beveiligingssysteem is nog niet aanwezig binnen de SMRA. Omdat er meerdere applicaties gebruik van het beveiligingssysteem zouden maken is er behoefte aan een universeel toepasbaar beveiligingssysteem.

2.6 Opdrachtomschrijving

Het is de bedoeling dat de diensten die door de SMRA worden aangeboden aan kerkelijke gemeenten in de toekomst met behulp van Internet worden aangeboden. Om de diensten over het Internet te kunnen aanbieden is er behoefte aan een universeel toepasbaar beveiligingssysteem dat het mogelijk maakt verschillende diensten op een veilige manier beschikbaar te stellen via het Internet.
Omdat veel van deze diensten persoonlijke gegevens, waaronder die over de religieuze achtergrond, ontsluiten is het van het uiterste belang dat deze gegevens op een veilige manier kunnen worden ingezien en gemuteerd door bevoegde personen.

Het beveiligingssysteem dient in staat te zijn bevoegde personen te beheren. Een deel van het beheer wordt uitbesteed aan gemeentelijke beheerders. Hiervoor is een gelaagd model nodig. Dit omdat de lokale kerken beter op de hoogte zijn van de bevoegde personen dan de SMRA. Ook worden er vele mutaties verwacht op de gegevens van de bevoegde personen. Hiermee wordt bedoeld dat de rollen van functionarissen vaak zullen wisselen. Dit is het gemakkelijkst te beheren als het beheer van functionarissen op het niveau van de kerkelijke gemeente plaatsvindt. Zo wordt de SMRA beheerder in staat gesteld om de gemeentelijke beheerders te beheren en de gemeentelijke beheerders worden in staat gesteld om de onderliggende functionarissen te beheren.

Het beveiligingssysteem definieert en beheert de te ontsluiten diensten in termen van deelcomponenten (functies, URL's) waardoor deze afgeschermd kunnen worden voor niet bevoegde personen.

Binnen dit systeem kunnen de beheerders van de SMRA functies toevoegen, beschikbaar stellen, terugnemen en blokkeren voor bepaalde kerkelijke gemeenten. Een beheerder in deze gemeente kan deze functie dan beschikbaar stellen voor een onderliggende functionaris die deze functie dan kan gebruiken. Ook dienen de functies te kunnen worden geblokkeerd op meerdere aggregatie niveau's.

Het beveiligingssysteem beveiligt zichzelf. De beheerfuncties van het beveiligingssysteem worden voor zowel de interne als externe gebruikers eveneens door hetzelfde beveiligingssysteem beveiligd.

Het gelaagde beheersmodel en de nodige intelligentie voor het beheer van bevoegde personen en functionaliteiten maakt dit systeem specifiek voor de SMRA.

Het nieuwe systeem zal gebruik maken van Microsoft SQL 2000 server en ASP.NET. Er zal een onderzoek moeten worden verricht naar ASP.NET. In dit onderzoek zal het concept van Microsoft.NET worden onderzocht en wordt er onderzocht naar de beveiligingsmethodieken binnen ASP.NET.

2.7 Doelstellingen

Hieronder zijn de doelstellingen gegeven die bereikt moeten worden.

1. Het beveiligingssysteem is in staat bepaalde personen de bevoegdheid te geven voor het opvragen van beveiligde pagina´s.
2. Het beveiligingssysteem is in staat *alleen* bevoegde personen toegang te geven tot de beveiligde pagina´s.
3. Het beveiligingssysteem is in staat de gegevens van beveiligde pagina´s op een veilige manier over het Internet te versturen.
4. De gemeentelijke beheerders, functionarissen en beveiligde webpagina's dienen door het beveiligingssysteem beheerd te kunnen worden.
5. Het systeem dient gebaseerd te zijn op ASP.NET.

2.8 Afbakening

Het beveiligingssysteem wordt ontworpen, gebouwd en op systeemniveau getest.
De onderdelen die niet bij het project worden uitgevoerd zijn hieronder expliciet gegeven.

- Het invoeren van de applicatie is geen onderdeel van de opdracht;
- Van de applicatie worden geen handleidingen opgeleverd;
- Van de applicatie worden geen vormen van opleidingsmateriaal opgeleverd.

2.9 Op te leveren producten

Tijdens het project zullen de volgende producten worden opgeleverd:

- Document: Plan van aanpak;
- Document: Onderzoek .NET;
- Document: Definitiestudie;
- Document: Pilotrapport per pilotdeel;
- Een beveiligingssysteem dat de hierboven genoemde functionaliteiten bevat.

3. Methoden en technieken

3.1 Inleiding

Door bij het ontwikkelen gebruik te maken van een ontwikkelmethode, wordt er aan zo veel mogelijke verwachtingen voldaan van de opdrachtgevers, kan er binnen een korte tijd een concreet resultaat bereikt en worden de benodigde ontwikkeltijd en kosten teruggebracht. Een systeem dat correct is ontwikkeld volgens een ontwikkelmethode is uiteindelijk begrijpelijk, uitbreidbaar, aanpasbaar, herbruikbaar en onderhoudbaar.

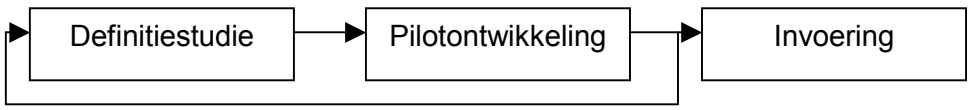
3.2 IAD ontwikkelmethode

Er is voor de ontwikkelmethode IAD gekozen vanwege de iteratieve ontwikkelingsstrategie. Het IAD ontwikkelmodel bestaat uit een aantal fasen die meerdere malen kunnen worden uitgevoerd, ofwel geïtereerd.

De IAD ontwikkelmethode heeft een aantal voordelen:

- Bij het project kan het iteratieve gedrag ervoor zorgen dat de complexiteit wordt verminderd;
- Er worden al snel resultaten bereikt die gebruikt kunnen worden voor feedback;
- Aan het begin van elke iteratie is het mogelijk om de doelen, eisen en oplossingen te evalueren of te herzien;
- Door het te ontwikkelen systeem onder te verdelen in kleine delen die elk apart kunnen worden ontwikkeld, ontstaat een overzichtelijke manier van werken.
- De kans dat risico's zich voordoen wordt verkleind omdat er een frequente toetsing plaats vindt en door de meest risicovolle delen in de eerste iteraties op te nemen.

Binnen IAD wordt er volgens een iteratie strategie gewerkt. De iteratiestrategie die voor dit project is gekozen is de strategie: big-bang-invoeren.



Big-bang-invoeren is een iteratiestrategie waarin de eerste 2 fasen (definitiestudie en pilotontwikkeling) iteratief worden uitgevoerd, maar de laatste fase (invoering) niet. Er worden dus weliswaar diverse pilots na elkaar ontwikkeld, maar geen van die pilots worden ingevoerd voor gebruik in de organisatie. Dit gebeurd pas als de laatste, definitieve pilot is ontwikkeld.

Er is voor deze strategie gekozen omdat:

- De opleverdatum vast staat, en snelle oplevering is daardoor niet nodig;
- Het invoeren van de applicatie is geen onderdeel van de opdracht, maar deze strategie stelt de SMRA in staat na het project het systeem alsnog in te voeren;
- Het afzonderlijk opleveren per pilot zal geen bruikbaar systeem voor de organisatie opleveren.

Omdat het invoeren van het systeem geen onderdeel is van het project wordt deze stap niet uitgevoerd. Maar omdat het systeem uiteindelijk wel volledig gerealiseerd zal zijn, wordt het bedrijf uiteindelijk alsnog in staat gesteld het systeem in te voeren in de organisatie.

3.3 UML technieken

Tijdens de opleiding zijn er UML technieken en technieken van Yourdon aangeleerd. Voor dit project wordt er gebruik gemaakt van UML. Dit omdat UML een techniek is die is gericht op objectgeoriënteerde programmeeromgevingen. ASP.NET is immers een objectgeoriënteerde programmeertaal.

3.4 Database Model

Binnen UML is er geen ondersteuning voor het ontwikkelen van relationele databases. Om deze reden is er gekozen om gebruik te maken van de ER techniek. De techniek ER modellering geeft een conceptueel gegevensmodel en wordt algemeen gebruikt voor het structureren en modelleren van de gegevens.
Een ERD geeft een grafisch model dat de lay-out beschrijft van de opgeslagen gegevens op een hoog abstractieniveau. Aan de hand van een ERD is er gemakkelijk een implementatiemodel te ontwikkelen waarmee de database kan worden opgezet.

4. Standaards en richtlijnen

4.1 Inleiding

Om de integriteit van de documentatie te bewaren, zijn enkele standaards en richtlijnen opgesteld. Deze zijn aangeleerd tijdens de opleiding op de Haagse Hogeschool.

4.2 Rapportagerichtlijnen

De rapportagetechnieken die zijn aangeleerd tijdens de module AV-03 zijn gebruikt om te kunnen voldoen aan de vereiste documentopmaak. Door gebruik te maken van de checklist die als bijlage is opgenomen in het plan van aanpak, voldoet de documentatie aan de vereiste opmaak.

4.3 Ontwerp- en ontwikkelrichtlijnen

Tijdens het ontwerpen en ontwikkelen van de applicatie is rekening gehouden met enkele eisen die Ghezzi stelt in het boek “The Fundamentals of Software Engineering”. Bij dit project zijn de onderstaande eisen nagestreefd:

- Correctheid;
- Robuustheid;
- Gebruikersvriendelijkheid;
- Performance.

In het boek van Ghezzi is per eis beschreven hoe er aan deze eis kan worden voldaan. Deze beschrijvingen zijn gebruikt bij het ontwerpen en ontwikkelen van de software.

Om de programmeercode duidelijk en leesbaar te laten zijn voor derden, is een standaard opbouw van de programmeercode gebruikt. Tijdens de module SW-06 zijn richtlijnen hiervoor gegeven, die gelden als standaard opbouw binnen de Haagse Hogeschool.

5. Risico's

5.1 Inleiding

Door risico's, die zich tijdens het uitvoeren van een project kunnen voordoen, vast te leggen, kan er met deze risico's rekening worden gehouden. Hierdoor wordt de kans dat de risico's zich voordoen verkleind. Bij elk risico is een maatregel aangegeven die moet worden genomen om de invloed van het risico op het project te verkleinen.

5.2 Risico's en maatregelen

Hieronder zijn de risico's weergegeven:

- *Mogelijke tijdsvertraging door ziekte*
  Maatregel: Bij de planning wordt er rekening gehouden met 1 week overhead. Indien er sprake mocht zijn van ziekte moet worden bekeken of de overhead al is gebruikt. Mocht dit het geval zijn dan zal de opdracht moeten worden ingekort in overleg met de opdrachtgever(s). Mocht de ziekte langer aanhouden dan 1 week zal ook, in overleg de opdrachtgever(s), de opdracht moeten worden ingekort.

- *De ASP.NET programmeeromgeving is te ingewikkeld om deze binnen de gestelde periode te beheersen en hiermee de applicaties te ontwikkelen.*
  Maatregel: Tijdens de fase onderzoek wordt er onderzoek gedaan naar de mogelijkheden van ASP.NET. Hiermee kan het niveau van ASP.NET globaal worden bepaald.

- *Het project is te groot om te realiseren tijdens de beschikbare tijd.*
  Maatregel: de opdracht moet, in overleg met de opdrachtgever(s), worden ingekort.

- *Problemen met apparatuur.*
  Maatregel: dagelijks back-ups maken van de applicatie en de documentatie.

- *Faciliteiten niet beschikbaar*
  Maatregel: de benodigde faciliteiten van te voren vaststellen en controleren of deze beschikbaar zijn. Faciliteiten die nog niet beschikbaar zijn worden gelijk bij de desbetreffende persoon aangevraagd.

- *Tekort aan informatie*
  Maatregel: het realiseren van de definitiestudie. Tijdens deze definitiestudie worden zoveel mogelijk gegevens verzameld over de opdracht, zodat een tekort aan informatie voor zo ver mogelijk wordt uitgesloten.

- *De techniek .Net is te ingewikkeld om binnen de gestelde periode te beheersen.*
  Maatregel: De benodigde technologieën van .NET worden onderzocht tijdens de fase onderzoek.

- *Het beveiligingssysteem universeel maken is te ingewikkeld of niet mogelijk tegenover de gestelde eisen.*
  Maatregel: In overleg de eisen zo aanpassen zodat het systeem uiteindelijk wel universeel kan worden gebruikt.

- *De middelware om mutaties op de Adabas database uit te kunnen voeren kan niet worden gebruikt met ASP.NET.*
  Maatregel: Tijdens de fase onderzoek wordt er onderzocht naar de mogelijkheden op dit gebied.

- *De oorspronkelijke doelstellingen vervagen tijdens het project.Omdat er op een iteratieve manier ontwikkeld zal worden, is het waarschijnlijk dat de systeemeisen regelmatig worden bijgesteld. Dit kan resulteren in een 'scope creep', een informatiesysteem die nooit 'af' is.*
  Maatregel: er wordt tijdens de definitie fase een afspraak gemaakt over de systeemeisen. Door tijdens deze fase prioriteiten aan de systeemeisen toe te kennen wordt er een afspraak gemaakt over de uiteindelijke eisen waaraan het systeem moet voldoen.

- *Omdat er voor het eerst een iteratieve manier van ontwikkelen wordt gebruikt, kunnen zaken zoals incomplete pilots, workshops, hergebruik, prioriteiten en timeboxing een verkeerd beeld opwekken.*
  Maatregel: IAD bestuderen voordat er een start wordt gemaakt met het ontwikkelen.

6. Planning

6.1 Inleiding

Een planning is bedoeld om te bepalen uit welke fasen het project zal bestaan en uit welke activiteiten deze fasen bestaan. Per uit te voeren activiteit wordt een schatting gemaakt van de tijd die deze in beslag zal nemen. Dit resulteert in een overzicht waarbij de tijd tegenover de activiteiten wordt gezet, de werkelijke planning.

Tijdens het verloop van het project is zo te controleren of er wordt gewerkt volgens de planning en zodoende te beoordelen of de gestelde deadline niet zal worden overschreden. Mocht er sprake zijn van uitloop dan is dit snel merkbaar en kunnen maatregelen worden genomen.

6.2 Fasen planning

Het project doorloopt een reeks van fases. Hieronder zijn de fases weergegeven met de daarbij behorende belangrijkste activiteiten:

1. **Onderzoek;**
   - .NET Framework.
   - Visual Studio;
   - ASP.NET;
   - Beveiliging binnen ASP.NET;
   - MS SQL 2000 server.
2. **Definitiestudie;**
   - Ontwikkelscenario;
   - Definitie van de systeemeisen;
   - Systeemconcept;
   - Pilotplan.
3. **Pilot ontwikkeling (per pilotdeel);**
   - Pilotontwerp;
   - Feitelijke bouw;
   - Beoordelen en testen.
4. **Oplevering.**
   - Acceptatietest;
   - Overhead.
5. **Eindverslag**
   - Project evaluatie
   - Project verantwoording

Tijdens het gehele traject wordt er gewerkt aan een eindverslag. Dit eindverslag neemt ongeveer 3 weken of 15 dagen in beslag.

Omdat er wordt gewerkt volgens de IAD ontwikkelmethode zal na elk opgeleverd pilotontwikkelrapport, de fase definitiestudie opnieuw worden doorlopen. De definitiestudie hoeft hierbij niet opnieuw te worden gerealiseerd maar wordt tijdens het doorlopen verfijnd of uitgebreid.

In de definitiefase wordt de fase pilotontwikkeling onderverdeeld in verschillende pilots. Het aantal pilots waarin het project wordt opgedeeld geeft aan hoe vaak de fase pilotontwikkeling wordt doorgelopen. Elke keer dat de fase pilotontwikkeling wordt doorgelopen, levert dit een nieuw pilotontwikkelrapport op.

Hieronder is in procenten aangegeven hoeveel tijd er nodig wordt geschat voor het uitvoeren van een fase.

Fase	Geschat percentage
Onderzoek	15%
Definitiestudie	20%
Pilotontwikkeling	55%
Oplevering	10%

6.3 Urenplanning

Aan dit project wordt gewerkt door een afstuderende student. Deze student heeft voor het uitvoeren van deze opdracht 18 weken ter beschikking waarvan er 15 zijn bedoeld voor het project en 3 voor het realiseren van het eindverslag. In de planning wordt het project verspreidt over deze 18 weken en wordt er parallel gewerkt aan het eindverslag.

De eerste week is reeds gepasseerd waarin IAD is bestudeerd en dit plan van aanpak is opgesteld. Er blijven zo nog 14 weken over om in te plannen voor het project. In uren omgerekend is dit 14 x 40 = 560 uur. Aan de hand van de beschikbare uren en de voorgaande tijdlijn is hieronder een indeling weergegeven met hierin aangegeven de geplande uren per project fase.

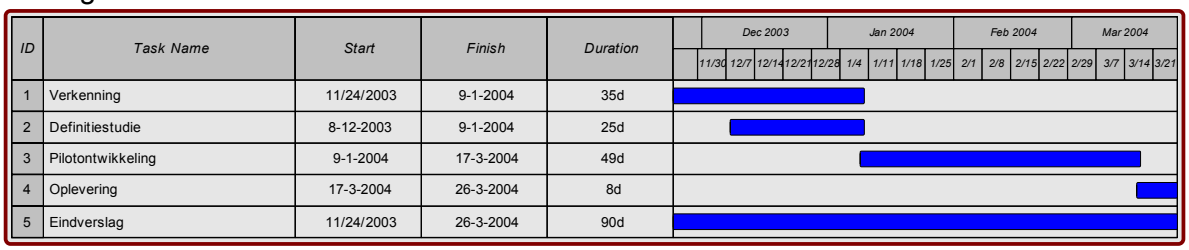
Fase	Minimaal	Maximaal	Gemiddeld	Planning
Onderzoek	77	95	82	85
Definitiestudie	90	138	110	110
Pilotontwikkeling	260	360	300	310
Oplevering	35	65	60	55
Eindverslag	100	140	120	120

(minimaal + maximaal + 2 * gemiddeld) / 4 = planning)

Ook hier geldt dat de fase definitiestudie en de fase pilotontwikkeling meerdere malen zullen worden doorlopen en dat de fase eindverslag zich verspreidt over het gehele traject.

6.4 Overzicht planning

Hieronder is een schema weergegeven met hierop aangegeven wanneer er aan welke fase wordt gewerkt.



Ook hier geldt dat de fase definitiestudie en de fase pilotontwikkeling meerdere malen zullen worden doorlopen. Op 9 januari wordt verwacht dat het onderzoek is afgerond en dat de eerste volledige versie van de definitiestudie is gerealiseerd.

Tijdens het realiseren van de definitiestudie zal er een verfijnde versie van het plan van aanpak worden gerealiseerd voor de definitiefase. Ook wordt er tijdens de definitiestudie een pilot plan opgesteld. Hierin wordt de planning gegeven voor elke pilot.

Tijdens de fase onderzoek zal ook de definitiefase plaatsvinden. Dit omdat deze fasen nauw samenhangen. In deze periode liggen ook enkele feestdagen. Om deze reden er is rekening gehouden met een uitloop van 5 werkdagen.

7. Faciliteiten

7.1 Inleiding

Om het project uit te kunnen voeren zijn er faciliteiten benodigd. De benodigde faciliteiten zijn in dit onderdeel van het document gegeven. Door te controleren of de benodigde faciliteiten aanwezig zijn wordt uitgesloten dat het project op een later tijdstip niet haalbaar blijkt te zijn doordat niet alle benodigde faciliteiten beschikbaar bleken te zijn.

De benodigde faciliteiten zijn in dit onderdeel onderverdeeld in: benodigde software, benodigde hardware en benodigde literatuur.

7.2 Benodigde software

- Visual studio.Net 2003
- Microsoft SQL 2000 server (of andere MS-SQL variant)
- IIS 5.0 (webservice versie 5.0 of hoger)
- MS-Office Word
- Windows XP professional
- Windows 2000 server
- Microsoft .Net Framework

7.3 Benodigde hardware

- Werkstation dat voldoet aan de aangeraden systeemspecificaties die worden gesteld door Visual studio.NET 2003.
- Server die voldoet aan de aangeraden systeemspecificaties die worden gesteld door Windows SQL 2000 server.
- Een werkstation dat toegang heeft tot de Webserver/SQL server en het Internet.

7.4 Beschikbare literatuur:

- IAD: Het evolutionair ontwikkelen van informatiesystemen, Pilot 2;
- ASP.Net in 24 uur, Joe Martin & Brett Tomson;
- Praktisch UML, Jos Warmer & Anneke Kleppe;
- Use cases combined with Booch OMT UML;
- Van der bulcke;
- Reader nr: 741, DB-01, relationele databases;
- Reader nr: 908, AV-03, rapportage technieken

8. Kwaliteitscontrole

8.1 Inleiding

Tijdens het doorlopen van het traject zullen er documenten, ontwerpen en programmeercode worden opgeleverd. De kwaliteit van deze onderdelen zal moeten worden gewaarborgd en daarom worden gecontroleerd. In dit onderdeel wordt beschreven hoe de kwaliteit van deze onderdelen gecontroleerd zal worden.

8.2 Kwaliteitscontrole documentopmaak

De kwaliteit van de opmaak van de documenten wordt behouden door gebruik te maken van de standaards en richtlijnen gesteld in hoofdstuk 4. Om de documentopmaak te kunnen controleren kan Bijlage A worden gebruikt als checklist.

8.3 Kwaliteitscontrole softwareontwerp

In de opdrachtomschrijving is aangegeven welke wensen de opdrachtgever heeft. Deze dienen uiteindelijk te worden teruggevonden in de applicatie. Tijdens de definitiefase worden de functionele eisen (mogelijke gebruikershandelingen met het systeem) vastgelegd met de MoSCoW regels. Met behulp van de MoSCoW regels worden de eisen toegewezen aan verschillende prioriteiten.

De functionele eisen die als Must Have zijn aangegeven zullen dan uiteindelijk ook in de applicatie verwerkt moeten zijn. De overige functionele eisen zullen aan de hand van hun prioriteit verwerkt worden in de applicatie totdat de applicatie aan alle functionele eisen voldoet of totdat de beschikbare periode ten einde is gekomen.

Er wordt gebruik gemaakt van een systeemontwikkelingmethode en verschillende technieken om zo de kwaliteit van het uiteindelijke product te waarborgen.

Tijdens het ontwikkelen van de code zullen de kwaliteitseisen correctheid, robuustheid, gebruikersvriendelijkheid en performance worden nagestreefd. Dit is mogelijk door de eisen uit het boek van Ghezzi, the fundamentals of software engineering, te controleren.

9. Testaanpak

Het testen is verdeeld in 2 onderdelen, de systeemtest en de acceptatietest. Bij het onderdeel systeemtest wordt het informatiesysteem getest op fouten. Het testen wordt gedaan door middel van testsets die zijn opgezet tijdens het ontwikkelen van de pilots. Hiermee wordt de werking van de correctheid, robuustheid en performance getest.

Bij het onderdeel acceptatietest wordt er een eerste versie van de applicatie opgeleverd aan de opdrachtgever(s) voor een acceptatietest. Op basis van deze acceptatietest wordt de applicatie beoordeeld door de opdrachtgever.

9.1 Systeemtest

Bij het uitvoeren van de systeemtest worden er verschillende onderdelen getest. De applicatie dient te worden getest op de werking, functionaliteit en gebruikersvriendelijkheid. Elk onderdeel moet voldoen aan de kwaliteitseisen die gesteld zijn in hoofdstuk 8. Hieronder is elk onderdeel apart beschreven.

Werking
De werking zal worden gecontroleerd d.m.v. zogenaamde testinvoer (testsets). De applicatie dient hierop te reageren zoals wordt verwacht. Om deze test te kunnen doen, zal een testset worden geschreven.

Functionaliteit
De functionaliteiten van de applicaties moeten worden vastgelegd. Dit zal worden gedaan door middel van de MoSCoW regels. Door gebruik te maken van deze regels kan per functionaliteit de prioriteit worden aangegeven. Bij de opgeleverde versie kan stapsgewijs worden gecontroleerd welke functionaliteiten de applicatie bezit. De functionele eisen van de applicatie zullen tijdens de definitiefase worden vastgelegd.

Gebruiksvriendelijkheid
De wachttijden (response) van de applicatie moeten acceptabel zijn. Schermen dienen overzichtelijk te zijn. De projectgroep zal de applicatie testen op aanwezigheid van deze eisen. De gebruikersvriendelijkheid zal worden getest door iemand die niet bij het project betrokken is geweest en die zich voor zou kunnen doen als een gebruiker van het systeem.

9.2 Acceptatietest

Zodra de eerste versie van de applicatie klaar is, zal deze worden getoond aan de opdrachtgever. De opdrachtgever dient op basis van de opdrachtomschrijving een oordeel uit te brengen over het opgeleverde product.
Deze test zal aan het eind het project plaatsvinden.

Bijlage A: Beoordeling rapportagetechniek

Onderdeel	Element	Aanwezig/correct
Titelpagina	Titel (en evt. ondertitel)	ja/nee
	Naam student(en)	ja/nee
	Naam opdrachtgever/docent	ja/nee
	Modulenummer en -naam	ja/nee
	Plaats en datum	ja/nee
	(Naam bedrijf)	ja/nee
Voorwoord	Tussen titelpagina en inhoudsopgave	ja/nee
	Persoonlijk	ja/nee
	Kader/doelgroep/leeswijzer	ja/nee
	Dankbetuiging(en)	ja/nee
	Plaats, datum en namen studenten	ja/nee
Inhoudsopgave	Inhoudsopgave en voorwoord afwezig	ja/nee
	Inleiding (= hoofdstuk 1)	ja/nee
	Niveau's in hoofdstukken (max. 3) aangebracht	ja/nee
	Verband tussen hoofdstukken en paragrafen	ja/nee
	Inspringen per niveau	ja/nee
	Typografisch onderscheid per niveau	ja/nee
	Titels: informatief	ja/nee
	Consequent/gelijkvormig	ja/nee
	Kernachtig geformuleerd	ja/nee
	Literatuurlijst (géén hoofdstuk)	ja/nee
	Bijlagen met nummer/letter en titel	ja/nee
	Paginanummering	ja/nee
Inleiding	Aanleiding (achtergrond/probleem//belang)	ja/nee
	Doelstelling rapport: hoofdvraag	ja/nee
	Werkwijze	ja/nee
	Randvoorwaarden/uitgangspunten	ja/nee
	Structuurbeschrijving/samenvatting hoofdstukken	ja/nee
	Alinea-indeling	ja/nee
	Goede volgorde alinea's	ja/nee
Kern	Elk hoofdstuk begint met inleiding	ja/nee
	Heldere structuur: één onderwerp per alinea	ja/nee
	alinea's beginnen met kernzin	ja/nee
	alineagroepen	ja/nee
	puntsgewijze opsommingen	ja/nee
	Conclusies en aanbevelingen	ja/nee
	Rapportindeling overeenkomstig inhoudsopgave	ja/nee
	Verwijzingen naar de bijlagen	ja/nee
Literatuurlijst	Volgens richtlijnen voor titelbeschrijvingen	ja/nee
Bijlagen	Nummer/letter en titel	ja/nee
	Zelfstandig leesbaar	ja/nee
	Paginanummering loopt door in de bijlagen	ja/nee

Publieksgerichtheid	Leesbaar/begrijpelijk voor opdrachtgever/ belanghebbenden	ja/nee
Verzorgdheid	Spelling correct/ typefouten ontbreken	ja/nee
Taalkundig correct		ja/nee

Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Document**
  Evaluatierapport: Onderzoek Microsoft.Net
- **Auteur**
  Mark van Houten (20015081)
- **Datum**
  26 juli 2004
- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag
- **Opleidingsvariant**
  MBIT
- **Periode**
  November 2003 – april 2004
- **Opdrachtgever**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft
- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Voorwoord

Voor u ligt het evaluatierapport van het onderzoek naar ASP.NET ten behoeve van een Web based beveiligingssysteem bij de SMRA. Dit rapport is tot stand gekomen tijdens een verkenningsfase van het project. Het betreft een onderzoek naar de Microsoft.NET technologieën en ASP.NET die nodig is geweest voor het uitvoeren van het project. Een deel van dit document is dan ook rechtstreekse informatie uit andere informatiebronnen.

De informatie die tijdens dit onderzoek is opgedaan is in dit document vast gelegd. Dit rapport stelt de lezer in staat een globaal idee te vormen over de Microsoft .NET omgeving. Omdat het onderzoek is gedaan voor het ontwikkelen van een beveiligingssysteem bij de SMRA wordt er specifiek gekeken naar de beveiligingsmechanismen die door .NET worden ondersteund.

Delft, SMRA, 29 januari 2004

Inhoudsopgave

1. Inleiding

Dit document richt zich op de .NET omgeving, ook wel het ".NET framework" genoemd. Er zal een korte introductie worden gegeven van Microsoft.NET waarbij wordt omschreven wat .NET is, waar .NET gebruikt kan worden en wat de verwachte toekomst is van .NET. Een onderdeel van de Microsoft.NET omgeving is het .NET framework. Het .NET framework wordt apart besproken. Het framework is de "ruggengraat" van de .NET omgeving. Om deze reden wordt deze apart besproken.

Ook XML webservices zijn een onderdeel van het netwerk. Dit is een vrij nieuwe ontwikkeling en er wordt gekeken naar de mogelijkheden van XML webservices om zodoende te bepalen of webservices kunnen worden gebruikt bij het ontwikkelen van het beveiligingssysteem. In het hierop volgende hoofdstuk wordt er gekeken naar de mogelijkheden van ASP.NET.

Het volgende onderdeel dat zal worden besproken is ADO.NET. Er zal worden beschreven wat ADO.NET is en hoe dit werkt. De verschillende onderdelen binnen ADO.NET zullen aan bod komen zoals: datareaders, datasets en data-adapters. Uiteindelijk zal er worden beschreven hoe ADO.NET binnen ASP.NET gebruikt kan worden.

Hierna zal de beveiliging van ASP.NET globaal worden besproken. Er wordt gekeken naar de onderdelen binnen ASP.NET die zorgen voor de beveiliging en de configuratie van de beveiliging. Ook worden de overige mogelijkheden voor het zelf ontwikkelen van beveiligingsdelen behandeld. Omdat er binnen ASP.NET verschillende vormen van authenticatie kunnen worden gebruikt zijn deze onderzocht en zijn deze vastgelegd. Zowel het beveiligingsmodel als het login proces wordt van elke authenticatie vorm besproken.

Uiteindelijk is er naar de beveiliging binnen het SMRA systeem. De eerder besproken onderdelen uit dit rapport worden in dit hoofdstuk gerelateerd aan het te ontwikkelen beveiligingssysteem bij de SMRA.

2. Introductie Microsoft.NET

2.1 Inleiding

Dit hoofdstuk geeft een globale omschrijving van .NET. Er wordt gegeven wat .NET is en waarvoor het kan worden gebruikt. Verder wordt er gegeven op welke platvormen .NET kan worden gebruikt en wordt er gegeven op welke platformen .NET kan worden ontwikkeld. Ook worden er verschillende programmeeromgevingen bekeken waarin .NET kan worden ontwikkeld.

2.2 Wat is .NET?

Microsoft.Net is een nieuwe omgeving voor het ontwikkelen en uitvoeren van software. De omgeving biedt gemak bij het ontwikkelen van websites en in de omgeving kan in verschillende talen worden geprogrammeerd. .Net zorgt ervoor dat software onderling, zowel over het Internet als lokaal op de computer, op een gemakkelijke manier gegevens kan uitwisselen.

.NET is opgebouwd uit een aantal technologieën voor het verbinden van informatie, mensen, systemen en apparaten. Het is gemakkelijk benaderbaar door middel van XML-webservices. Het .NET-platform maakt het mogelijk op XML gebaseerde toepassingen, processen en websites te maken en te gebruiken als services die samen informatie en functionaliteit delen en combineren op elk willekeurig platform of intelligent apparaat, met als doel maatwerkoplossingen te bieden aan organisaties en individuele gebruikers.

.NET kan worden gebruikt voor het ontwikkelen van Windows software. Maar ook bij het ontwikkelen van software biedt .NET nieuwe mogelijkheden en technieken die gebruikt kunnen worden. Er is een apart onderdeel gericht op het ontwikkelen van websites die gebruik maken van de .NET technologieën, namelijk ASP.NET.

2.3 Besturingssystemen waar .NET op draait

De platvormen die worden ondersteund door .NET zijn: Windows XP, 2000, NT4 met SP6a en Windows ME/98. Windows 95 wordt niet ondersteund.
Sommige delen van het framework werken niet op alle platvormen. ASP.NET wordt bijvoorbeeld alleen ondersteund door Windows XP en 2000.

Om gebruik te maken van de .NET technologie is het nodig om het .NET framework te installeren op de computers waarop het wordt gebruikt. Het .NET framework is de ruggengraat van alle software die gebruik maakt van .NET functionaliteit. Op de platvormen Windows 95/98/ME kan geen .NET software worden ontwikkeld.

ASP.NET kan worden gebruikt binnen een (Internet Information Services). IIS wordt niet ondersteund door Windows XP home edition en kan om deze reden niet worden gebruikt voor het hosten van ASP.NET. Er kan gebruik worden gemaakt van de ASP.NET Web Matrix Server en kan worden gebruikt om ASP.NET te hosten op een Windows XP home machine.

2.4 Software ontwikkeling met .NET

Binnen de omgeving .NET kan er worden geprogrammeerd in verschillende programmeertalen. Microsoft biedt voor het .NET framework de compilers voor C#, C++, VB en JScript. Andere producenten hebben aangekondigd compilers te ontwikkelen voor talen als COBOL, Eifel, Perl, Smalltalk en Python.

Er zijn verschillende tools die gebruikt kunnen worden voor het ontwikkelen van .NET applicaties. Hieronder zijn de verschillende ontwikkelomgevingen opgesomd met de daarbijbehorende informatie.

- .NET Framework SDK
  Het SDK is een gratis ontwikkelomgeving and biedt compilers voor C++, C# en VB.NET.

- ASP.NET WebMatrix
  Dit is een gratis ontwikkelomgeving van Microsoft en bied een grafische gebruikersinterface. Deze omgeving is te downloaden vanaf de Microsoft website. Bij het download bestand zit een simpele webserver inbegrepen die kan worden gebruikt in plaats van IIS om ASP.NET applicaties te kunnen hosten. (Geschikt voor Windows XP Home)

- Microsoft Visual C# .NET Standard 2003
  Dit is een pakket is een kleine versie van Visual studio die ongeveer € 120,- kost. Deze versie is gelimiteerd tot het gebruik van 1 programmeertaal en minimale "wizard" ondersteuning. Er is ook een versie beschikbaar voor C#, VB.NET en C++.

- Microsoft Visual studio .NET professional 2003
  Dit pakket ondersteunt alle Microsoft programmeertalen en heft uitgebreide "wizards".

2.5 Toekomst van .NET

Licatec over:

"Toekomst van .NET

Nu de mist wat is opgetrokken begint het langzamerhand duidelijk te worden hoe wij Microsoft .Net moeten plaatsen. Een aantal jaren geleden was de .Net strategie nogal verwarrend, en gebruikte iedereen binnen Microsoft te pas en te onpas de .Net term om aan te geven hoe modern hun product wel niet was. Gelukkig heeft Microsoft de boel opgeschoond, de meeste verwarrende .Net benamingen zijn weer verdwenen. Hieronder geven wij onze mening over en onze ervaring met .Net

Het is een prachtige ontwikkelomgeving

Dot Net is primair een prachtige ontwikkelomgeving. Microsoft heeft van de strijd tegen IBM in de OS/2 tijd (eind jaren 80) geleerd dat de club die de mooiste ontwikkelomgeving heeft de harten van de programmeurs wint, en daarmee haar platform verrijkt met mooie software. De .Net ontwikkelomgeving is ook veel krachtiger dan de bestaande Java ontwikkelomgevingen die beschikbaar zijn. Het grootste voordeel is dat je niet alleen server, maar ook cliënt applicaties kan maken. Java ontwikkelomgevingen zijn meer gericht op server toepassingen. Ook heeft Java slechts een beperkte groep aanhangers, het wordt alleen maar wordt gebruikt binnen sommige grote organisaties, die alleen voor intern gebruik software ontwikkelen. Microsoft heeft dus de slag om de programmeeromgeving van de toekomstgewonnen, maar wie had anders verwacht?

Het lijkt wel heel erg op J2EE

Microsoft heeft .Net ingezet omdat binnen grote bedrijven steeds meer systemen werden ontwikkeld in de Java omgeving J2EE. Microsoft moest snel met een antwoord komen, en heeft daarom heel goed gekeken naar J2EE, en op bijna onbeschaamde wijze alle functionaliteit hiervan soms letterlijk overgenomen. Maar men is veel verder gegaan dan J2EE en .Net bevat dan ook veel meer functionaliteit, vooral gericht op Cliënt toepassingen.

Het is een platform, en dit houdt de ontwikkeling tegen
Je moet om .Net op een computer te draaien de .Net runtime installeren, dit is een heel groot bestand van 18Mb. Eigenlijk is deze runtime een uitbreiding van het operating system, zodat een object georiënteerd operating system ontstaat. Maar het zal nog vele jaren duren voordat een stabiele versie hiervan op een voldoende percentage computers in de wereld is geïnstalleerd. Tot dat tijdstip zal de markt erg terughoudend zijn met het leveren van .Net applicaties, het aantal computers waarop het draait is immers beperkt. Het mooie van het .Net platform is wel dat het in principe op alle soorten computers kan worden geïmplementeerd, van handhelds tot Sun Servers. Op langere termijn kan Microsoft hierdoor zijn markt verbreden.

Het is nog niet stabiel
Geen enkele techneut had verwacht dat .Net meteen na introductie stabiel zou zijn. Dat blijkt in de praktijk ook realiteit te zijn, alleen eenvoudige Web applicaties, waarbij slechts een deel van het .Net platform wordt gebruikt zijn stabiel. Maar zodra je iets ingewikkelders met .Net wil doen blijken er snel kleine en grote problemen te ontstaan. Wij hebben bijvoorbeeld een .Net applicatie gemaakt, maar helaas verdwenen op de meest onverwachte momenten spontaan de knoppen van het invoerscherm. Gelukkig was dit een eenvoudig systeem, dat wij snel in Delphi konden herschrijven, maar het geeft aan hoe moeilijk het in de praktijk vaak is om systemen met nieuwe technologie te realiseren. Met Delphi weet je zeker dat je altijd een werkend systeem kan krijgen omdat de sources van de libraries beschikbaar zijn. Maar met .Net is dit onmogelijk, dus als je eenmaal vast zit dan is er verder niets meer aan te doen.

Het zit niet in Office 2003
Officieel is de ontwikkeling van Visual Basic gestopt door Microsoft, dus wij hadden gehoopt dat in Office 2003 je C# of Basic.Net macro's kon programmeren. Maar het nieuwe Office 2003 bevat nog steeds het oude getrouwe Visual Basic. Wat hiervan de reden is zal over enkele jaren wel worden gepubliceerd in één van de vele boeken die regelmatig over Microsoft zullen verschijnen. Het kan technisch moeilijk zijn, men kan het de gebruikers niet willen aandoen, maar het meest waarschijnlijke is een ordinair een machtsconflict tussen de Office groep en programmeertalen groep van Microsoft.

Met .Net haal je een personeelsprobleem in huis
Natuurlijk verkoopt Microsoft .Net en de ontwikkelomgeving als iets waarmee je snel systemen kan realiseren. De waarheid is echter dat om met Basic.Net te werken je behoorlijk goed moet zijn in object georiënteerd programmeren. Iemand die goed is in Java of in Delphi behoeft alleen wat andere termen te leren, en binnen een paar weken is het een even goede Basic.Net programmeur. Iemand zonder deze ervaring moet een nieuw begrippenkader, een nieuw paradigma, en een geheel nieuwe manier van programmeren leren. Alleen de betere meer intelligente programmeurs zijn hiertoe in staat, de rest zal afhaken. Een organisatie die overgaat op .Net haalt tegelijkertijd hierdoor een personeelsprobleem in huis."

(Bron: http://www.lizatec.com/LIZATEC/TECHTRENDS/TOEKOMSTNET)

3. .NET Framework

3.1 Inleiding

Het .NET Framework is een van de grond af opnieuw ontwikkeld platform om zowel webontwikkelaars als ontwikkelaars van traditionele programma's in staat te stellen hun toepassingen efficiënter te bouwen en ze flexibeler te laten werken. De twee belangrijkste componenten van het .Net framework zijn: de "common language runtime" en de "class library". Een ander onderdeel dat tijdens het project van toepassing zal zijn is het ontwikkelen van een server side applicatie. In dit deel van het document worden deze onderdelen globaal besproken.

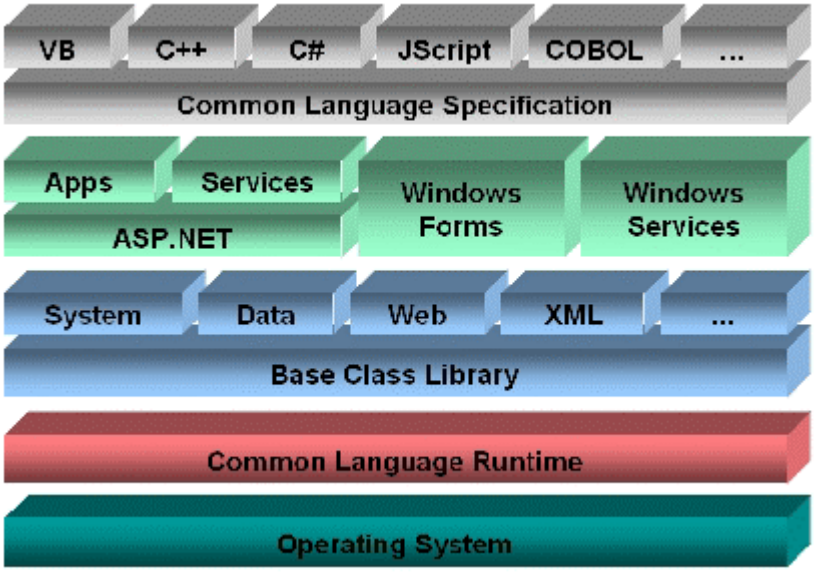
3.2 Common Language Runtime (CLR)

De basis van het .NET Framework wordt gevormd door de Common Language Runtime (CLR). De CLR zorgt voor het uitvoeren veilig en efficiënt uitvoeren van applicaties. De CLR beheert het geheugen, controleert of de applicatie wel voldoende rechten heeft om bepaalde acties uit te voeren, en of er niets gedaan wordt waardoor het systeem instabiel kan worden. De CLR werkt bovenop het besturingssysteem, zodat applicaties niet direct met het besturingssysteem communiceren. Op die manier worden applicaties afgeschermd van de complexiteit van het besturingssysteem.

Applicaties die onder het .NET Framework werken, zijn te onderscheiden in vier verschillende soorten:

- Windows Forms applicaties (desktop applicaties en client-server applicaties)
- Windows Services (diensten)
- ASP.NET applicaties (webapplicaties)
- ASP.NET Web Services (diensten die via het web aan te roepen zijn door andere systemen)

Al deze applicaties worden gemaakt met een van de programmeertalen die het .NET Framework ondersteunen. Dit zijn alle talen die vastliggen in de Common Language Specification. Een schematische weergave van het .NET Framework is hieronder gegeven.



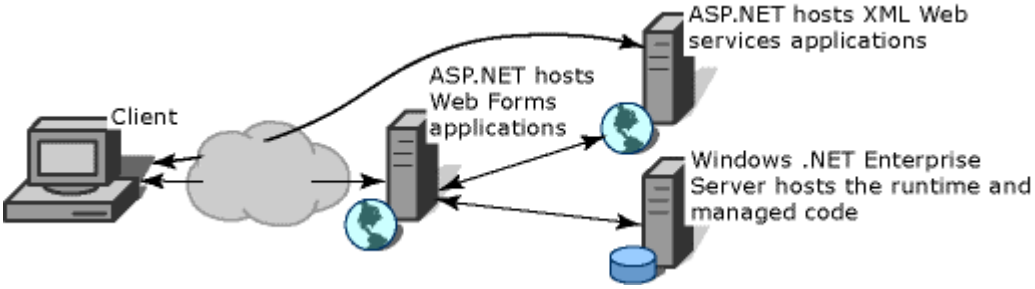
Onder deze structuur liggen de systeemservices (Operating system). Bovenop deze systeemservices bevind zich de common language runtime. De CLR laadt en voert de code uit die is geschreven die zich richt op de CLR. Ook biedt de CLR geïntegreerde beveiliging.

3.3 Class libraries

De class libraries is een object georiënteerde collectie van bruikbare functies die gebruikt kunnen worden bij het ontwikkelen van applicaties. De library biedt de functies die anders door het besturingssysteem worden geboden. Op die manier worden applicaties afgeschermd van de complexiteit van het besturingssysteem. Alle applicaties die werken op het .NET Framework maken zo gebruik van deze functies, in plaats van die aangeboden door het besturingssysteem. Deze collectie kan worden gebruikt voor zowel GUI (Graphical User Interface) applicaties als “command line” applicaties gebaseerd op ASP.NET, zoals bijvoorbeeld Web Formulieren en XML webservices.

3.4 Server-side applicatie ontwikkeling

Het volgende Figuur toont een standaard netwerkschema met daarin managed code draaien op de verschillende server omgevingen. Server omgevingen zoals IIS en SQL server kunnen standaard operaties uitvoeren terwijl de geschreven applicatie de managed code uitvoert.



ASP.NET is een host omgeving die de mogelijk bied aan ontwikkelaars om met behulp van web-based applicaties gebruik te maken van het .NET framework. ASP.NET is een complete architectuur voor het ontwikkelen van websites en andere Internet gedistribueerde objecten die gebruik maken van managed code. Zowel web formulieren als XML webservices gebruiken IIS en ASP.NET om applicaties te publiceren. Ook bieden zij beide een collectie classes in het .NET framework.

XML webservices worden op dezelfde manier gedistribueerd als standaard websites. Componenten van XML webservices beschikken niet over een gebruikersinterface en zijn niet bedoeld voor het gebruik met Internet Explorer of Netscape Navigator. XML webservices bieden software componenten die gebruikt kunnen worden door andere applicaties, zoals standaard cliënt applicaties, webbased applicaties of andere webservices.

4. XML Webservices

4.1 Inleiding

In dit onderdeel van het document wordt er gegeven wat XML webservices zijn en waarvoor deze kunnen worden gebruikt. Ook wordt er beschreven hoe toepassingen met elkaar kunnen worden verbonden met behulp van XML webservices.

4.2 Wat zijn XML-webservices?

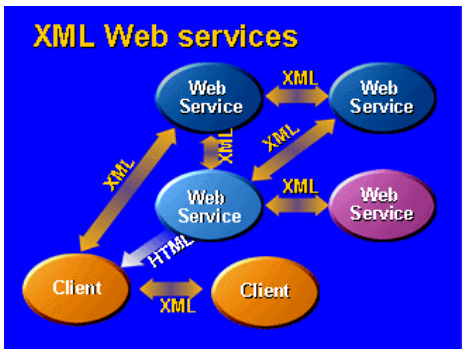
XML-webservices stellen toepassingen in staat met elkaar te communiceren via het Internet, ongeacht het besturingssysteem of de programmeertaal. Zij kunnen op elk platform worden geïmplementeerd en zijn gedefinieerd door openbare standaardisatie. XML webservices kunnen ook andere functies vanuit andere toepassingen aanroepen. Ze wisselen gegevens uit in XML. XML is een universele vorm voor het indelen van informatie. Doordat zij gegevens kunnen uitwisselen via XML, zijn zij in staat onafhankelijk van elkaar te werken en elkaar zijn gegevens te kunnen gebruiken.
De hoofdpunten van XML webservices zouden als volgt kunnen worden aangegeven:

- XML-webservices stellen toepassingen in staat gegevens uit te wisselen;
- XML-webservices kunnen over platformen en besturingssystemen heen worden aangeroepen en zijn programmeertaalonafhankelijk;
- .NET is het platform voor XML-webservices van Microsoft.

4.3 Verbinden van toepassingen met XML webservices

Websites zijn bedoeld voor het presenteren van informatie aan een gebruiker: zij vormen het communicatiemedium dat servers gebruiken om met gebruikers te communiceren. XML-webservices bieden services die direct met andere services kunnen communiceren. Services die intern dan wel extern worden gehost kunnen via internet met elkaar communiceren door middel van XML en SOAP berichten.

Services kunnen hun gegevens beschikbaar stellen in XML zodat ze kunnen worden ingelezen door andere applicaties. Deze koppelingen kunnen met behulp van XML-webservices op eenvoudige wijze tot stand worden gebracht. XML-webservices stellen de toepassingen in staat informatie uit te wisselen via het Internet, ongeacht het besturingssysteem of de back-endsoftware waarmee de toepassing werkt. Hieronder is, in een overzicht, een voorbeeld gegeven met hierin de communicatie tussen verschillende webservices en cliënts.



4.3.1 SOAP

SOAP staat voor Simple Object Acces Protocol. Dit is protocol wordt gebruikt om informatie, binnen een webservice request of response, te coderen voordat het wordt verstuurd over een netwerk. Ze werken onafhankelijk van elk besturingssysteem of protocol en kunnen worden verstuurd met behulp van verschillende Internet protocollen, bijvoorbeeld HHTP.

4.3.2 UDDI

UDDI staat voor Universal Description, Discovery and Integration. Dit is een gedistribueerde directory die het mogelijk maakt om webservices op Internet bekend te maken en elkaar op te kunnen zoeken. Dit is goed te vergelijken met het gebruiken van een gouden gids.

4.3.3 WSDL

WSDL staat voor Web Services Description Language. Dit is een volgens XML opgemaakte taal voor het beschrijven van de mogelijkheden die een webservice biedt. WSDL is een onderdeel binnen UDDI en is de taal die wordt gebruikt door UDDI.
WSDL bied op deze manier een standaard interface voor een webservice.

5. ASP.NET

5.1 Inleiding

In dit onderdeel van het document wordt er beschreven wat ASP.NET is en waarvoor dit gebruikt kan worden. Ook wordt er gegeven wat asp.net pagina´s zijn en hou deze zijn opgebouwd.

5.2 Wat is ASP.NET

Active Server Pages (ASP) is een webontwikkelplatform dat gecreëerd is door Microsoft. Met ASP kunnen webpagina's dynamisch worden ontwikkeld. Hiermee wordt bedoeld dat de websites aan de hand van de acties van de gebruiker, verschillende gegevens kan weergeven. Zo kan er bijvoorbeeld gebruikt worden gemaakt van een database.
In een ASP pagina staat HTML code met scripts. Gewoon HTML wordt uitgevoerd aan de kant van de cliënt en ASP wordt uitgevoerd aan de server klant. Het resultaat van de uitgevoerde scripts wordt naar de browser gestuurd. Met dit script kunnen bijvoorbeeld de gegevens uit een database worden gelezen en weergegeven in een web browser.

ASP.NET is de opvolger van ASP en onderdeel van het .NET framework. Het maken van ASP.NET applicaties lijkt meer op het maken van Visual Basic applicaties. Eén van de voordelen van ASP.NET is dat er geprogrammeerd kan worden in verschillende talen. De talen die standaard door ASP.NET ondersteund worden zijn:

- Visual Basic .NET (VB.NET)
- C#
- JavaScript (JScript.NET)

Om ASP.NET pagina's te kunnen bekijken is alleen een webbrowser nodig. Om ASP.NET pagina's te maken en uitvoeren, is er specifieke software nodig. Er is een IIS webserver nodig en het .NET framework is nodig om de ASP.NET pagina's te kunnen uitvoeren.

In het klassieke ASP waren er 6 objecten en hoefde er minder te worden geleerd. In ASP.NET (Framework versie 1)zijn er ongeveer 4,000 klassen en 78.000 methode en eigenschappen beschikbaar.

5.3 ASP.NET pagina's

ASP.NET pagina's zijn webpagina's zijn pagina's die eindigen op de extensie .aspx. Deze webpagina's maken gebruik van de ASP.NET Controls. De controls genereren de inhoud van de webpagina die de gebruiker krijgt te zien en kan gebruiken. ASP.NET controls kunnen worden gebruikt samen met HTML. Er kan zo een website worden ontwikkeld met een statisch deel (HTML) en dynamisch deel (ASP.NET).

5.3.1 Namespaces

Omdat het .NET framework bestaat uit duizenden klassen (tot nu toe meer dan 3.400) worden deze georganiseerd in een hiërarchie van "namespaces". Een namespace is een logische groepering van klassen. Een goed voorbeeld hiervan is de namespace System.IO. Deze namespace bevat de klassen die gerelateerd zijn aan het werken met het bestandssysteem.

De namespaces zijn ingedeeld in een boom structuur met als stam de namespace: "System". Deze namespace bevat alle klassen voor de basis datatypen zoals strings en arrays.

Binnen een ASP.NET webpagina is er de mogelijkheid gegeven om te verwijzen naar een specifieke klasse. Om een klasse specifiek te laten verwijzen naar een bestandssysteem

bestand (de file klasse) wordt dit op de volgende manier aangegeven: “System.IO.File”. Hierbij verwijst System.IO naar de namespace en File naar een specifieke klasse.

5.3.2 ASP.NET pagina structuur

Hieronder zijn de elementen van de ASP.NET pagina structuur opgesomd.

- Directives
- Code declaration blocks
- ASP.NET controls
- Code render blocks
- Server-side comment
- Server-side include directives
- Literal text en HTML tags

Directives
Een directive bestuurt de manier waarop een ASP.NET pagina is gecompileerd. Een directive wordt in de pagina geplaatst tussen de karakters <%@ “directive” %>. Een directive kan overal in een pagina worden geplaatst. Er zijn 2 soorten directives namelijk page en page imports directives. Een page directive wordt gebruikt om de standaard programmeertaal binnen de ASP.NET pagina aan te geven zoals bijvoorbeeld: <%@ Page Language=”VB”%>.
Met een Page import directive kunnen er bepaalde namespaces worden geïmporteerd in een ASP.NET pagina. Een ASP.NET pagina wordt standaard geladen een standaard set klassen. Het kan dus nodig zijn om specifieke klasse te importeren. Dit wordt gedaan met de Page Import directive. Een voorbeeld van een Page import directive kan zijn: “<%@ Import Namespace=”System.Web.Mail” %>”. Deze zorgt ervoor dat de klasse SmtpMail kan worden gebruikt omdat deze zich bevindt in de System.Web.Mail namespace.

Code declaration blocks
Binnen de code declaration blocks wordt de logica van de applicatie genoteerd tezamen met alle globale variabelen, subroutines en functies. Deze logica wordt weggeschreven binnen de <Script Runat =”Server> </Script> tags.

Aan de <Script> tag kunnen 2 attributen worden meegegeven. Het eerste attribuut is de taal (als er geen taal wordt gegeven wordt degene gebruikt die in de page directive is opgegeven).

<Script Language=”VB” Runat =”Server> </Script>

Het tweede attribuut is de tag SRC. Deze kan worden gebruikt om een extern bestand te gebruiken voor de inhoud van het code blok.

<Script Runat =”Server SRC=”SourceApplicationLogic.aspx”> </Script>

ASP.NET Controls
ASP.NET controls kunnen worden gebruikt voor de tekst en HTML inhoud van een pagina. Deze controls moeten worden gegeven tussen een <form Runat=”Server”> tag. Voor sommige tags zoals en <ASP:Label Runat=”Server” /> is dit niet nodig. Binnen een ASP.NET webpagina kan er zich maar één <form Runat=”Server”> bevinden.

Code render blocks
Als er binnen HTML of tekstdelen van een pagina code moet worden uitgevoerd kan er gebruik worden gemaakt van code render blocks. Hiervan zijn 2 verschillende typen

mogelijk. Dit zijn “inline code” en “inline expressions”.

Inline code voert een statement of een serie statements uit. Deze vorm van code wordt tussen de volgende karakters gegeven:

```
<%  %>
```

Inline expressions geven de waarde weer van een variabele of methode. Inline expressions worden tussen de volgende karakters gegeven:

```
<%= %>
```

Server-side comment
Binnen ASP.NET pagina’s kunnen er server-side comments worden toegevoegd (opmerkingen). Server-side comments worden tussen de volgende karakters gegeven:

```
<%--  --%>
```

Dit wordt gebruikt om commentaar toe te voegen tussen code. Dit commentaar kan niet worden gezien in een webbrowser maar alleen bij het inzien van de broncode.

Server-side include directives
Het is mogelijk een bestand in een ASP.NET pagina toe te voegen door een form te gebruiken van de server-side include directives. Als je een bestand wilt toevoegen uit dezelfde map of uit een submap kun je het volgende gebruiken:

```
<!-- #INCLUDE file=”includefile.aspx” -->
```

Maar er kan ook een include directive worden toegevoegd uit een virtuele map. Hierbij moet de volledige virtuele mapnaam worden opgegeven.

```
<!-- #INCLUDE virtual=”/virtualdirectory/includefile.aspx” -->
```

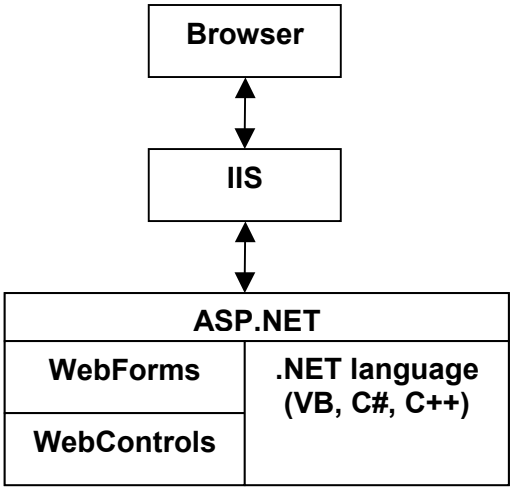
Er kunnen geen variabelen worden meegegeven in de aanroep van een include directive.

Literal text en HTML tags
Het laatste element dat kan worden gebruikt in een ASP.NET pagina is HTML inhoud. Het statische deel van de webpagina wordt opgebouwd uit standaard HTML tags en tekst. HTML wordt net als de rest van de elementen ook gecompileerd. De HTML inhoud van een pagina wordt gepresenteerd door de LiteralControl klasse. De Text eigenschap van deze klasse kan worden gebruikt om het HTML gedeelte van een ASP.NET webpagina te manipuleren.

5.4 Webforms

De basis van ASP.NET ligt in de Web Forms. Web Forms bezitten net als de windows forms dezelfde eigenschappen en methoden. Maar bij een ASP.NET web form wordt er gebruik gemaakt van UI elementen. Deze UI elementen zetten zichzelf om naar HTML.

WebForms zijn opgebouwd uit 2 componenten, het visuele deel (het .aspx bestand) en de code die daarachter wordt uitgevoerd (een aparte class file). Dit in tegenstelling tot ASP pagina's. De codebehind pagina (met een .NET language) wordt van te voren gecompileerd en alleen het visuele deel van de pagina wordt pas gecompileerd als hij wordt opgevraagd (on runtime). De mogelijkheid is er wel om binnen een ASPX pagina server side code te integreren. Dit gaat ten koste van de performance en overzichtelijkheid bij het ontwikkelen. Hieronder is weergegeven welke componenten er nodig zijn om een ASP.NET webform te ontwikkelen.



De browser bevind zich aan de client side en vraagt de webpagina op. IIS ontvangt de request voor de pagina en laad de webform pagina. ASP.NET is nodig om deze pagina te compileren en de pagina wordt obgebouwd met behulp van WebForms en Webcontrols. De .NET language runtime voert de achterliggende Code uit en ASP.NET vertaald de pagina naar een webpagina (HTML pagina). IIS stuurt de pagina terug naar de browser van de client en de browser toont de webbpagina op het scherm. Dit zorgt ervoor dat vrijwel elke webbrowser die HTML ondersteund deze pagina kan weergeven.

6. ADO.NET

6.1 Inleiding

In dit onderdeel wordt beschreven wat ADO.NET is en hoe dit werkt. Ook komen de verschillende onderdelen waaruit ADO.NET is opgebouwd, aan bod.

6.2 Wat is ADO.NET?

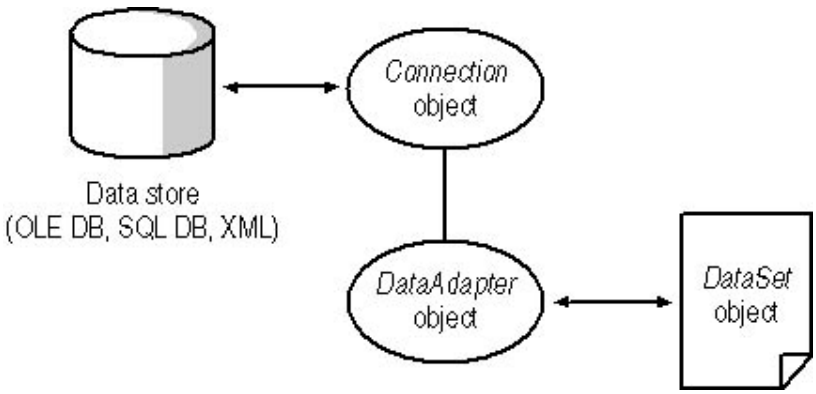
ADO.NET is binnen .NET de opvolger van ActiveX Data Objects (ADO). In het .NET Framework wordt gegevenstoegang gerealiseerd met een set van klassen onder de naam ADO.NET. ADO.NET is niet ADO dat is aangepast voor de .NET-infrastructuur maar is een nieuw programmeermodel voor gegevenstoegang dat een volledig begrip en een andere instelling vereist. Het is een volledig gedistribueerd model en het is geheel gebaseerd op XML (het Recordset- object is verdwenen). Onder ADO.NET zijn de functies van het Recordset-object verdeeld over drie nieuwe objecten: DataReader, Dataset en DataAdapter. Deze splitsing is het gevolg van het feit dat de functionaliteit, en hiermee ook de complexiteit van de Recordset in het ADO-model in de recente versies is toegenomen.

6.3 Hoe werkt ADO.NET

Er zijn 3 lagen om data te benaderen in ADO.NET.

- De physical layer
- De data provider
- De data set

De physical layer kan een OLE database, SQL database of zelfs een XML bestand zijn waarin de data is opgeslagen. De dataprovider bevat het connection object die de connectie tot stand brengt, beëindigd etc. Maar hij bevat ook de command objects die een representatie van de data in het geheugen kunnen plaatsen. De data layer zorgt voor de abstractie tussen de fysieke dataopslag en de data set waarmee wordt gewerkt. De data in de dataset wordt binnen de architectuur gezien als niet gerelateerd aan de data. De data set wordt gezien als een aparte, onafhankelijke databron. Hieronder is een figuur weergegeven met daarin het ADO.NET object model.



Binnen ADO.NET zijn er twee verschillende typen database connecties.
Er kan gebruik worden gemaakt van het OleDbConnection object om verbinding te maken met een lokale database. OLE database verbindingen gebruiken het OleDbDataAdapter object om commando´s uit te voeren en data in te lezen.
Ook kan er gebruik worden gemaakt van het SqlDbConnection object om verbindingn te maken met een database server. SQL database verbindingen maken gebuikt van het SqlDbAdapter object om commando´s uit te voeren en data in te lezen.

Ook zijn XML pagina´s direct toegankelijk vanuit data sets door gebruik te maken van de DataSet methoden: ReadXML en WriteXML. XML bestanden zijn een statische representatie van data sets. ADO.NET gebruikt XML voor het versturen van data over het Internet.

6.4 DataReaders, Dataset en DataAdapter

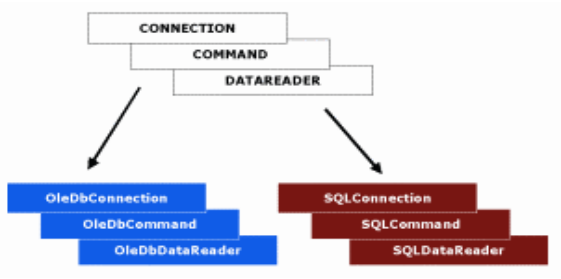
Microsoft over:

> *"Het DataReader-object is het object waarmee efficiënt de records van het resultaat van bijvoorbeeld een database query kan worden doorlopen.*
> *Het object Dataset heeft als functie een client-side cache te creëren van een of meer gerelateerde Recordsets.*
> *Het DataAdapter component heeft als functie om alle data-uitwisseling met een onderliggende database te verzorgen. Het DataSet-component bevat zelf geen specifieke database-functies en kan bijvoorbeeld ook gebruikt worden voor het laden, bewerken en updaten van data in XML-bestanden."*

6.5 De classes Connection, Command en Datareader

Microsoft over:

> *ADO.NET bevat drie abstracte classes, namelijk Connection, Command en DataReader. Voor het bouwen van .NET-applicaties zal de ontwikkelaar gebruik maken van de hiervan afgeleide OleDB of SQL classes. Voor toegang tot data waarvoor een OleDB provider beschikbaar is wordt er gebruikt gemaakt van instanties van de classes OleDbConnection, OleDbCommand en OleDBDataReader. Voor het benaderen van data in een SQL Server database, kun je als alternatief instanties van de SQLConnection, SQLCommand en SQLDataReader classes gebruiken die een betere performance geven, omdat deze direct werken met SQL Server's TDS (Tabular Data Stream) verbindingslaag (zie figuur 1).*
> *De DataReader zal veelal worden gebruikt om op een snelle en efficiënte manier de data die het resultaat is van een database query of stored procedure te lezen. Een typisch voorbeeld hiervan is het vullen van een listbox bij het opbouwen van een scherm met waarden uit de database. De ontwikkelaar is zelf verantwoordelijk voor het schrijven van de code en voor het lezen van de waarden uit de opgehaalde records waarmee de controls worden gevuld.*



6.6 Het hart van software ontwikkeling: DataSets

Microsoft over:

"Het hart van een softwareoplossing die gebruik maakt van ADO.NET is de al eerder genoemde DataSet. Een dataset is een in-memory store van databasegegevens. Een dataset bevat een willekeurig aantal gegevenstabellen, instanties van het ADO.NET DataTable-component, dat elk correspondeert met een databasetabel of view. Een dataset legt een gedistribueerde (disconnected) view vast van databasegegevens. Behalve de data in de tabellen kan een dataset ook informatie bevatten over constraints en relaties tussen de data in de tabellen. Dit maakt het mogelijk om behalve sequentieel ook hiërachisch te navigeren door de data in de DataSet.
ADO.NET moedigt ontwikkelaars aan dit gedistribueerde model te gebruiken. Gedistribueerde gegevens zijn schaalbaarder, omdat ze niet doorlopend een beroep doen op de database-servers. Wat je met name oplost met een gedistribueerd model is het feit dat er minder verbindingen actief zijn op de database, dat er geen locks aanwezig zijn en dat er minder network roundtrips worden doorlopen. Met name dit laatste valt positief uit voor de schaalbaarheid van een applicatie. Een ander groot voordeel van datasets is dat deze ADO en XML met elkaar verenigen. XML neemt twee beperkingen weg bij ADO. XML is geschikter om gegevens te verplaatsen over verscheidene platforms en door firewalls dan ADO Recordsets. Ook is XML in het algemeen in staat om meer gegevenstypen te beschrijven dan ADO. Zijn de gegevens echter tabulair, dan werken de beperkingen van ADO in jouw voordeel. ADO is beter 'getuned' om met deze vorm van gegevens te werken dan een algemeen XML Document Object Model (DOM).
Datasets zijn het resultaat van de samenvoeging van ADO en XML. Een Dataset bevat een of meer brokken XML in tabelvorm, genaamd datatabellen. Deze datatabellen kunnen afzonderlijk worden behandeld. Ook kunnen onderlinge relaties worden gedefinieerd tussen de datatabellen. Deze relaties geven de ontwikkelaar de gelegenheid ADO data-shaping toe te passen zonder de grillige SHAPE-taal te hoeven beheersen. Datasets hebben geen directe koppeling met databases of SQL. Het enige wat een dataset doet, is datatabellen manipuleren, uitwisselen of verbinden met gebruikersinterfaces. Er zijn twee standaardmanieren om datatabellen op te nemen in een dataset. De eerste is via het Datasetcommand-object dat de resultaten van een databasequery omzet in XML. Hiermee worden database-updates tot stand gebracht die doen denken aan de ADO client cursor engine, maar dan beheersbaarder. De tweede manier is om direct te werken met XML. Het dataset-object bevat methoden voor lezen en schrijven van XML-gegevens en schema's, en kan nauw samenwerken met een XMLDataDocument-object. De XMLDataDocument-class in het .NET Framework erft van de algemene XMLDocument-class en breidt het uit met specifieke functionaliteit voor de manipulatie van datatabellen. Hierdoor ontstaat een brug tussen een dataset en de generieke XML-document."

6.7 Typed Datasets: classes creëren die afstammen van de dataset

Microsoft over:

> *”Het .NET Framework ondersteunt overerving, waardoor het mogelijk is classes te creëren die afstammen van de dataset, en tevens functionaliteit toevoegen. Deze classes worden typed datasets genoemd en kunnen overal worden gebruikt waar een dataset wordt gebruikt. Dit betekent dat typed datasets alle mogelijkheden ondersteunen van de datasets die .NET bezit, waaronder binding en XML-interoperabiliteit. Een XML-schema, opgeslagen in een XSD-bestand, is het enige dat nodig is voor de creatie van een typed dataset. Visual Studio.NET zal vervolgens een dataset-subclass aanmaken die de ontwikkelaar van de juiste broncode voorziet. Deze code zorgt ervoor dat de data in de dataset te benaderen is via logische, en strong typed properties. De nieuwe class zal over 'strongly typed' eigenschappen beschikken die elke kolom in de gegevens vertegenwoordigen. De ontwikkelaar kan eenvoudig nieuwe eigenschappen en methoden toevoegen als de code eenmaal is gegenereerd. Bovendien is Intellisense beschikbaar voor de desbetreffende typed dataset. Kortom, ADO.NET gaat zijn beloften voor een nieuw programmeermodel voor gegevenstoegang waarmaken.”*

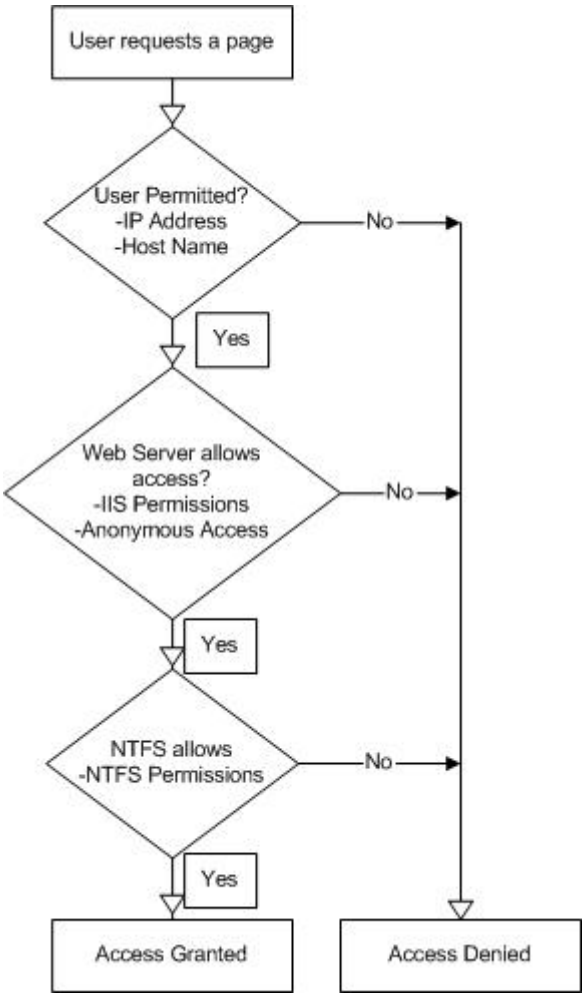
7. Vormen van authenticatie

7.1 Inleiding

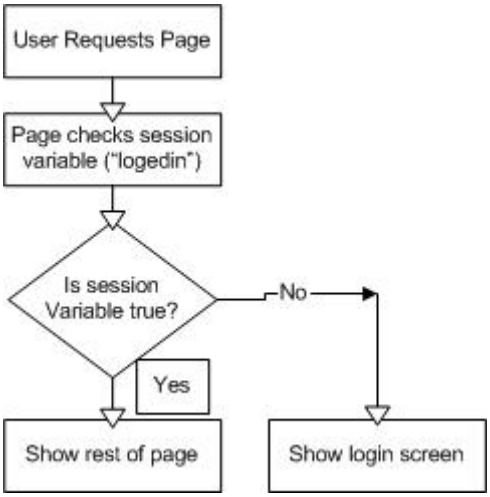
Authenticatie kan op verschillende manieren gebeuren. De eerste vorm van authenticatie die wordt besproken is de traditionele web beveiliging (Windows Integrated Authentication). Binnen ASP.NET zijn er verschillende vormen om gebruikers te authentiseren. In dit onderdeel worden deze vormen besproken zodat de meest bijpassende vorm van authenticatie kan worden gekozen voor het beveiligingssysteem.

7.2 Traditionele web beveiliging

Deze vorm van beveiliging voldoet voor vele systemen. Een nadeel van dit systeem is dat de individuele permissies binnen Windows moeten worden beheert (Windows user accounts). Als een gebruiker een pagina opvraagt, voert IIS de volgende stappen uit:



Om een beveiligde webpagina te ontwikkelen met eigen gebruikersbeheer kan er een database worden gebruikt.



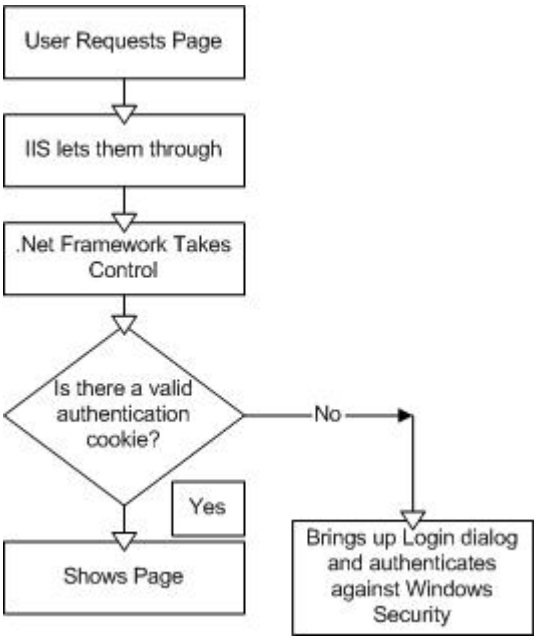
Deze layout vereist dat op elke pagina waarop de beveiliging nodig is, de beveiliging wordt geïntegreerd in de pagina. Dit is op te lossen door een beveiliging's script te ontwikkelen die in elke pagina wordt aangeroepen. Het moet wel nog steeds worden toegevoegd in elke pagina.

7.3 ASP.NET security

Binnen ASP.NET zijn er verschillende vormen van authenticatie. In dit onderdeel wordt de manier waarop deze beveiligingsmechanieken werkt schematisch weergegeven.

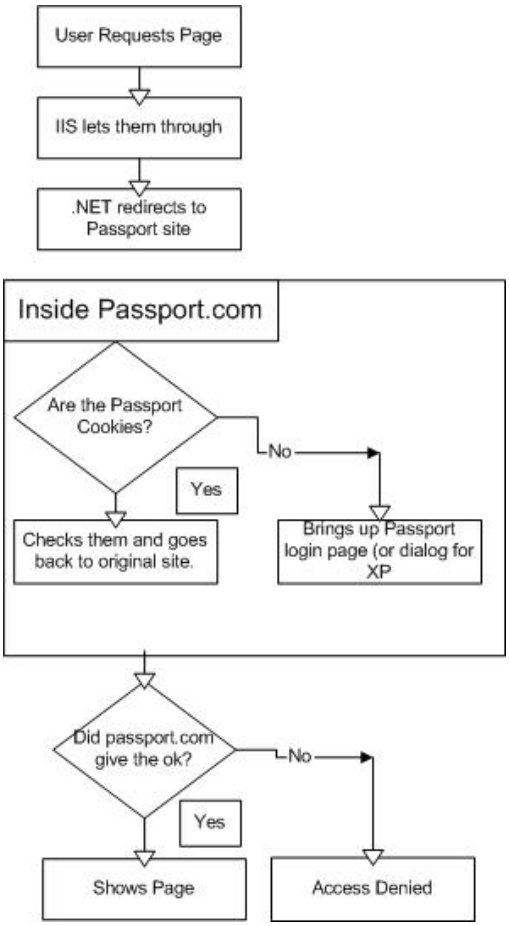
7.3.1 Windows Security

De windows security is vrijwel identiek aan IIS Integrated Windows Security.



7.3.2 Passport

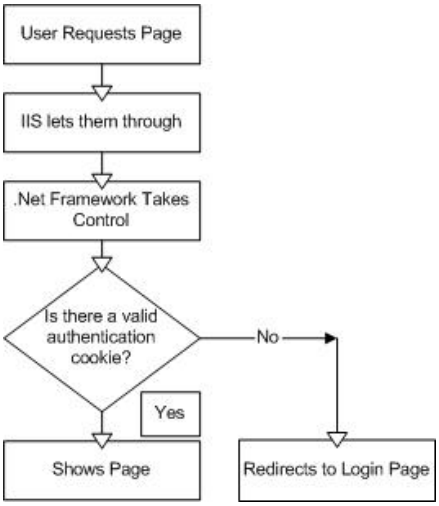
Binnen ASP.NET kan er gebruik worden gemaakt van een Passport. Het vereist daarom ook dat er en business account beschikbaar is met een bruikbaar passport.



Weliswaar hangt deze methode af van de passport.com (een site die pasporten biedt) pasport response. Een pasport response kan worden vervalst.

7.3.3 Forms Based Security

Deze methode stuurt de gebruiker door naar een form om in te loggen en een cookie te plaatsen.



By het gebruik van Form based security is het niet nodig om nieuwe Windows accounts aan te maken voor gebruikers. Hieronder is een voorbeeld gegeven van een web.config configuratie voor form based security.

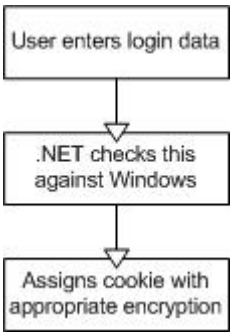
```
<configuration>
  <system.web>
    <authentication mode="Forms">
      <forms name="Auth" loginURL="login.aspx" protection="All" timeout="10" />
    </authentication>
  </system.web>
</configuration>
```

Als er iemand wordt geautoriseerd wordt hem een cookie toegewezen genaamd “Auth”. De pagina waar iemand op in logt is genaamd “login.aspx”. Als iemand een pagina probeert op te vragen en hij is nog niet geautoriseerd wordt hij automatische verwezen naar de login.aspx pagina.
Hieruit volgt een validatie cookie met encryptie (tripple-DES of DES). Dit kan expliciet worden aangegeven met de eigenschappen All, None, Encryption of Validation.
Hierna kan de timeout worden opgegeven. Dit is standaard 30 min.

7.4 Het logon Process

In dit onderdeel wordt het login process van de verschillende beveiligingsvormen binnen ASP.NET schematisch weergegeven.

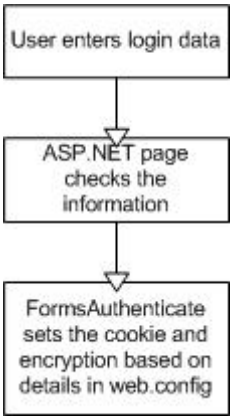
7.4.1 Windows Based



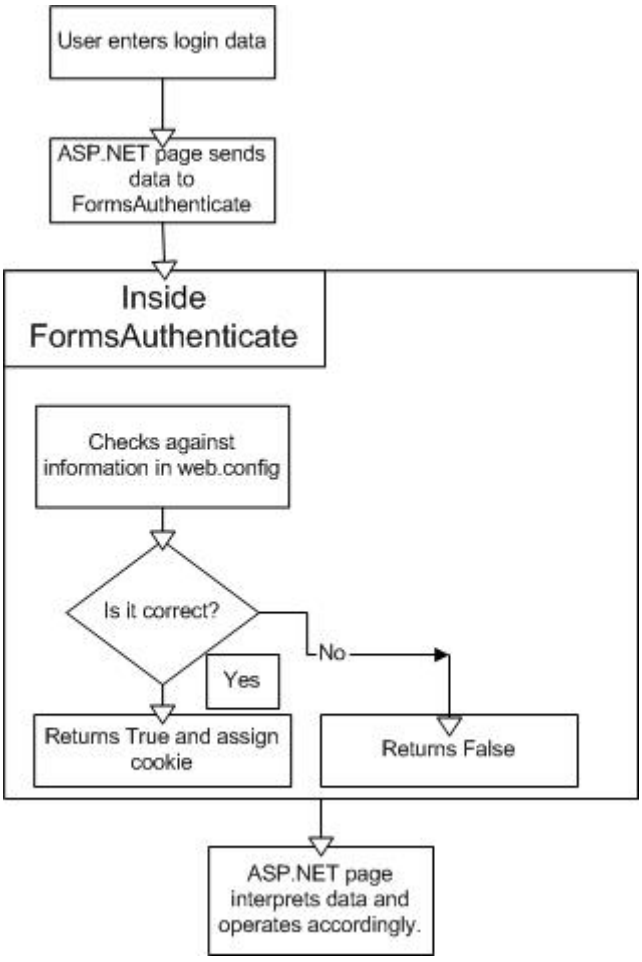
7.4.2 Forms based Security

Deze methode kan op 2 manieren worden toegepast.

1) Zonder credentials in web.config



2) Met credentials in web.config

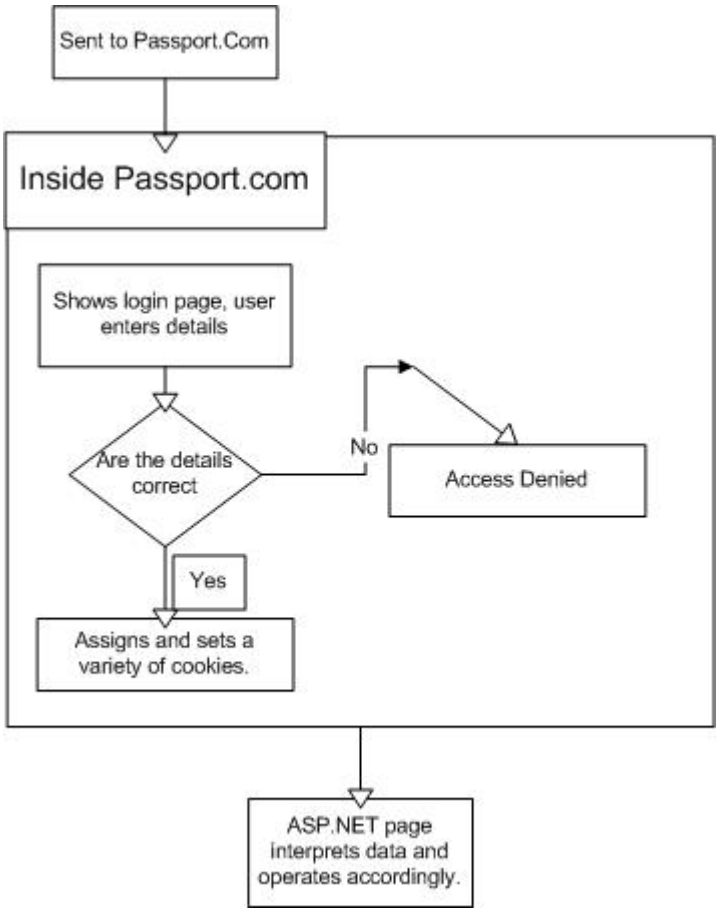


In dit schema is zichtbaar dat de FormsAuthenticate de gebruiker valideert en een cookie toewijst. Dit omdat de gebruikers zijn opgegeven als credentials in de web.config.

7.4.3 Passport

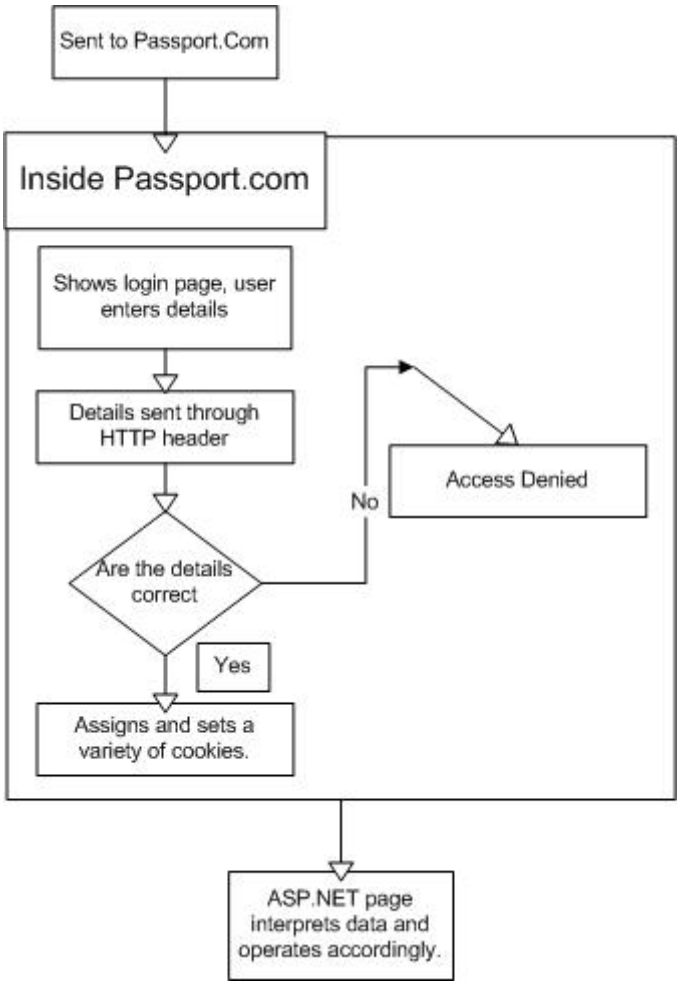
Deze methode werkt anders dan de voorgaande methode. Ook hier zijn 2 mogelijke manieren om deze methode te gebruiken.

1) Zonder Windows XP



Ook hier hangt het af van de passport response. De cookies laten de gebruiker toe op elk deel van de website en bezitten informatie zoals timeout, etc. Deze waarden worden allen gegenereerd door de paswoordsite.

2) Met Windows XP (IE6)

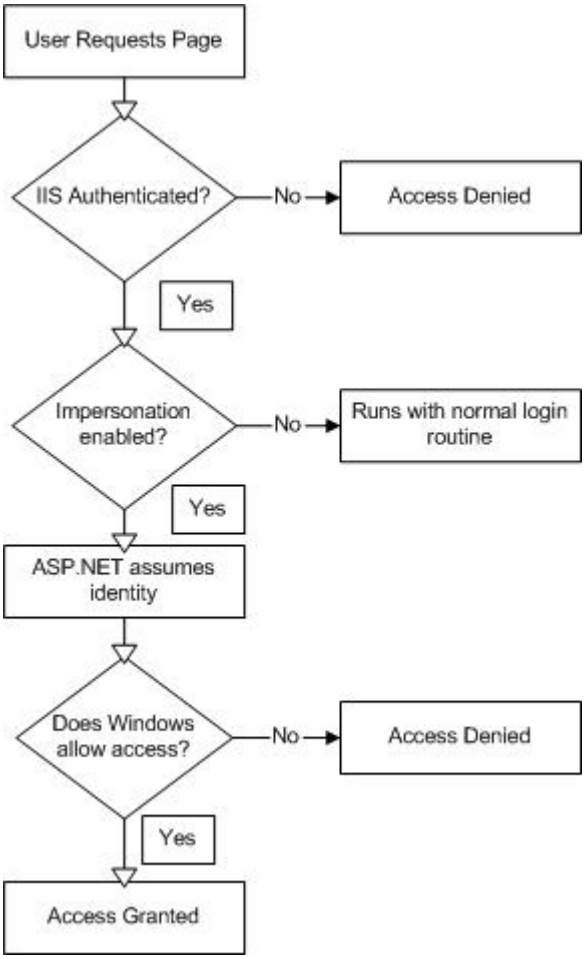


Op deze manier worden de details gestuurd door de “passport authentication”. Hiervan zijn geen verdere details bekend over de werking. Ook het gebruik van cookies varieert.

7.5 Impersonation

Impersonation zorgt ervoor dat ASP.NET de toegang tot bepaalde pagina's kan bepalen op de gebruikers identiteit. Normaliter zou ASP.NET een gebruiker toegang geven tot het lokale systeem en zodoende toegang geven tot alle bestanden en folders. Als Impersonation wordt toegepast, gebruikt ASP.NET de rol die IIS hem toewijst. Als ASP.Net zich voordoet als een gebruiker kan Windows de beveiliging bieden voor de applicatie.

Hieronder is een schema gegeven met daarin weergegeven de werking van de methode.



8. Vormen van autorisatie

8.1 Inleiding

Nadat een gebruiker is geauthentificeerd moet worden bepaald of de gebruiker toegang heeft (bevoegd is) om een bepaalde bron op te vragen. Dit proces wordt autorisatie genoemd. ASP.NET werkt samen met IIS om authenticatie en autorisatie te leveren voor applicaties.

Er zijn verschillende mogelijkheden om te bepalen of een gebruiker toegang heeft tot en bepaalde bron. De beveiliging van ASP.NET is gebouwd op de beveiligingsinfrastructuur binnen IIS. Elke communicatie tussen de cliënt en de applicatie passeert IIS en elk Windows proces dat invloed heeft op de persmissies van de gebruiker. Als er gebruikt wordt gemaakt van NTFS, onderhoud Windows een ACL (Acces Control List) voor elke bron waarover hij controle heeft. De ACL permissies gaan over alle andere vormen van autorisatie heen.

Hiernaast gebruiken ASP.Net applicaties ook configuratie bestanden voor autorisatie informatie. Alle ASP.NET applicaties erven hun beveiligingsconfiguratie uit de machine.config. Weliswaar kan elke applicatie instellingen van deze overschrijven op applicatie niveau in het bestand web.config. Het is belangrijk hierbij rekening te houden dat ASP.NET een ISAPI extension is. Dit houdt in dat hij alleen controle heeft over de resources die gebonden zijn aan Aspnet_isapi.dll. (Voor meer informatie: http://msdn.microsoft.com/library/en-us/cpguide/html/cpconConfigurationFiles.asp)

8.2 Autorisatie beheer

Binnen ASP.NET wordt autorisatie beheerd door de volgende componenten:

- Windows Access Control Lists (ACLs)
- Web Server permissies
- URL autorisatie
- .NET Principal Objects
- Roles and Method-Level beveiliging

In de volgende kopjes worden deze componenten apart besproken.

8.2.1 Windows Access Control Lists

Met Windows ACL's kunnen er bestandssysteem permissies worden uitgedeeld op specifieke applicatie bestanden. Deze oplossing werkt het beste als de applicatie gebruik maakt van Windows user accounts. Om gebruik te maken van Windows ACL's moet het bestandssysteem NTFS zijn. (Voor meer informatie: http://msdn.microsoft.com/library/en-us/security/security/access_control_lists.asp)

8.2.2 Web Server Permissions

Met behulp van IIS kunnen er ook permissies zoals “read acces” of “directory browsing”, worden gespecificeerd op directories binnen de website. In tegenstelling tot NTFS zijn webserver permissies voor elke gebruiker geldig die gebruik maken van Web of FTP sites. NTFS beheert fysieke mappen op de server en IIS beheert de permissies van de virtuele mappen. (Voor meer informatie: http://www.microsoft.com/windows2000/en/server/iis/htm/core/iiwspsc.htm)

8.2.3 URL Authorization

De URlAutorizationModule klasse legt de relaties tussen gebruikers, rollen en elementen met URL namespaces die gedeclareerd zijn door een URL. Deze module kan worden gebruikt om slectief toegang of geen toegang te geven aan specifieke gebruikers voor het opvragen van elementen van de URL namespace. Als voorbeeld, de toegang kan worden gebaseerd op de rol van een gebruiker. Het volgende voorbeeld geeft toegang tot verschillende domein gebruikers, terwijl de rest geen toegang wordt gegeve.

```
<!-- Web.config file -->
<configuration>
  <system.web>
    <authentication mode="Windows" />
    <authorization>
      <allow users="domain1\user, domain2\user2, domain1\user3" />
      <deny users="*" />
    </authorization>
  </system.web>
</configuration>
```

(Voor meer informatie: http://msdn.microsoft.com/library/en-us/cpguide/html/cpconASPNETAuthorization.asp of http://msdn.microsoft.com/library/en-us/cpref/html/frlrfSystemWebSecurityUrlAuthorizationModuleClassTopic.asp)

8.2.4 WindowsPrincipal Object

De System.Security.Principal namespace biedt een WindowsPrincipal klasse die de beveiliging representeert waar de code onder wordt uitgevoerd. Dit object wordt automatisch aangemaakt als er gebruik wordt gemaakt van Windows Authentication in IIS. Het geeft de mogelijkheid om een Windows groeplid te bekijken van een Windows gebruikersaccount en op basis hiervan de toegang te bepalen. (Voor meer informatie: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconCreatingWindowsIdentityWindowsPrincipalObjects.asp)

8.2.5 GenericPrincipal Object

Er kan een object worden gemaakt vanuit de GenericPrincipal klasse gebaseerd op zelf geconfigureerde rollen. Dit kan worden gebruikt as er gebruikt wordt gemaakt van een database voor de opslag van gebruikers/rollen. Het principel object kan worden aangesproken in het OnAuthenticate event. Er kan een gespecificeerde tabel worden gerelateerd aan een Windows account die wordt gebruikt bij dit event. Door die informatie te gebruiken, kan er een nieuw principel object worden aangemaakt voor de geauthentificeerde gebruiker. Als er wordt verwacht dat een gebruiker terugkomt, kan er gebruik worden gemaakt van een cookie om het principal opject opnieuw aan te maken. (Voor meer informatie:
http://msdn.microsoft.com/library/en-us/cpguide/html/cpconCreatingGenericPrincipalGenericIdentityObjects.asp)

8.2.6 Roles and Method-Level Security

Het kan zo zijn dat het nodig is om de beveiliging op methode niveau toe te passen. Zo kan per gebruiker (principal) worden aangegeven welke methodes mogen worden aangeroepen. Binnen dit niveau van beveiliging zijn er verschillende benaderingen mogleijk:

Als er gebruik wordt gemaakt van Windows accounts kunnen er rollen worden aangemaakt door een Windows gebruikersgroep aan te maken. Omdat er gebruik wordt gemaakt van impersonation is het WindowsPrincipal object beschikbaar.

- Creëer NTFS ALC's voor de beveiligde bronnen die worden opgevraagd door ASP.NET.
- Roep de WindowsPrincipalIsInRole methode op telkens als er moet worden gevalideerd of de gebruiker de nodige permissies bezit. Dit kan worden geïmplementeerd in een logisch statement binnen de code die een bepaalde subroutine aanroept gebaseerd op de rol (groep) van de gebruiker.
  http://msdn.microsoft.com/library/en-us/cpref/html/frlrfSystemSecurityPrincipalWindowsPrincipalClassIsInRoleTopic.asp

Als er gebruik wordt gemaakt van het GenericPrincipal object gemaakt van gebruikers en rollen uit een database:

- In de programmatuur kan er worden gecontroleerd welke rol de gebruiker bezit door het aanroepen van de GenericPrincipal.IsInRole methode.
  http://msdn.microsoft.com/library/en-us/cpref/html/frlrfSystemSecurityPrincipalGenericPrincipalClassIsInRoleTopic.asp

Als er geen gebruik wordt gemaakt van het principal object zijn er ook andere opties:

- Het op rol gebaseerde beveiligingsmodel ondersteunt een permissie object die ongeveer gelijk is aan de permissie objecten die zich bevinden in het code acces beveiligingsmodel. Dit object, PrincipalPermission, representeert de identiteit en rol die een normale klasse moet uitvoeren;
  http://msdn.microsoft.com/library/en-us/cpguide/html/cpconPrincipalPermissionObjects.asp
- Gebruikers credentials worden geaccepteerd als parameters om een methode aan te roepen en een zoekopdracht uit te voeren binnen deze methode;
- Controleren of een cookie bestaat als eerste operatie van de methode aanroep;
- Het bouwen van een login methode die een sleutelwaarde teruggeeft. Dit kan op dezelfde manier worden gebruikt als cookies. Maar dit kan ook worden gebruikt als de browsers geen cookies ondersteunen.

(Voor meer informatie:
http://msdn.microsoft.com/library/en-us/cpguide/html/cpconRole-BasedSecurity.asp)

Door de gebruikersnaam en rol door te geven aan de constructor, kan de identiteit van het principal object worden achterhaald die gekoppeld is aan de huidige sessie die gelijk is aan een gebruikersnaam en rol. Nadat de identiteit van het pricipal object is vastgesteld kunnen er beveiligingscontroles op hun worden uitgevoerd op één van de volgende manieren:

- **Imperatieve beveiligingscontroles**
  Dit achterhaald tijdens de runtime of het principal object van het huidige proces gelijk is aan het gespecificeerde PrincipalPermission object.
  Voor meer informatie:
  http://msdn.microsoft.com/library/en-us/cpguide/html/cpconUsingImperativeSecurity.asp

- **Gedeclareerde beveiligingscontroles**
  Dit maakt gebruik van het PrincipalPermissionAttribute om vast te stellen welke principal er benodigd is om het proces uit te mogen voeren.
  Voor meer informatie:
  http://msdn.microsoft.com/library/en-us/cpguide/html/cpconUsingDeclarativeSecurity.asp

- **Directe toegang tot het principal object**
  Door het Pricipal object direct op te vragen, kan er in de programmatuur een beslissing worden gemaakt gebaseerd op de eigenschappen van het principal object en het proces.
  Voor meer informatie:
  http://msdn.microsoft.com/library/en-us/cpguide/html/cpconDirectlyAccessingPrincipalObject.asp

Voor meer informatie over Roles and Method-Level Security:
http://msdn.microsoft.com/library/en-us/cpguide/html/cpconRole-basedSecurityChecks.asp

9. Configuratie van de beveiliging binnen ASP.NET

9.1 Inleiding

Nu de beschikbare vormen van authentificatie en autorisatie bekend zijn kan er gekeken worden naar de configuratie hiervan. Met behulp van het boek: “ASP.NET Unleashed” is dit onderzocht en is gericht op de configuratie voor de verschillende vormen van beveiliging.

9.2 SSL

Secure Sockets Layer (SSL) is de meest gebruikte manier voor het aangaan van beveiligde electronische transacties op het Internet. Deze verbinding verzekert dat alle gegevens die een webserver en browser uitwisselen, geheim blijven voor buitenstaanders. SSL is een wereldwijde standaard en wordt gebruikt door miljoenen websites voor het beveiligen van online transacties. Om een SSL-verbinding tot stand te brengen heeft de webserver een SSL-certificaat nodig.

9.2.1 Versleuteling

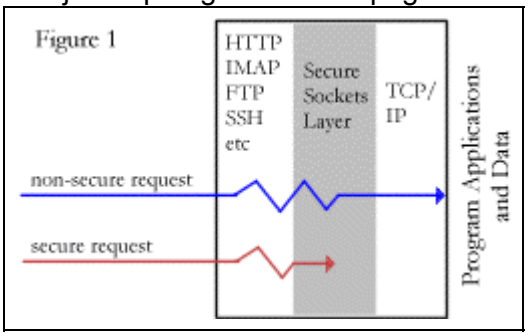
De methode gebruikt publieke sleutel cryptografie om een sessiesleutel te genereren en uit te wisselen. In het Publieke Sleutel concept worden twee sleutels gebruikt. Één sleutel is de Publieke Sleutel die op allerlei manieren verspreid wordt en die iedereen in zijn bezit mag hebben. De andere sleutel is de Privé Sleutel. Deze sleutel is geheim en mag niet verspreid worden. Alleen de eigenaar is in het bezit van de Privé Sleutel. Data die gecodeerd is door de publieke sleutel kan alleen worden gedecodeerd door de privé sleutel.

De sessiesleutel wordt gebruikt voor symmetrische versleuteling. Symmetrisch versleutelen kenmerkt zich door gebruik van één sleutel zowel bij het coderen (= versleutelen van de gegevens) als bij het decoderen (= reproduceren van de originele gegevens).

Omdat dezelfde sleutel in beide gevallen gebruikt wordt is het essentieel dat de sleutel geheim gehouden wordt. De versleutelde gegevens hoeven niet vertrouwelijk behandeld te worden, maar iemand die zowel de versleutelde gegevens als de sleutel in handen heeft, heeft ook toegang tot de originele gegevens.

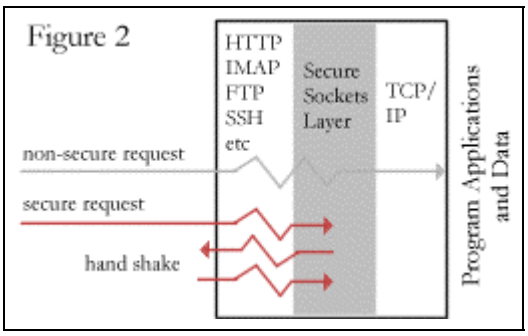
9.2.2 De werking van SSL

De communicatie over het internet verloopt meestal door verschillende applicatie lagen voordat het request aankomt bij het opvragen van een pagina of script.



Het eerste wat wordt aangesproken zijn protocols zoals http, IMAP of FTP. Welke er wordt aangesproken hangt af van het type request van de cliënt. Dit protocol geeft de request door aan de Secure Sockets Layer (SSL). Als het gaat om een request voor een niet beveiligde verbinding wordt hij doorgestuurd naar via TCP/IP laag naar de server-applicatie of de data.

Als het gaat om een request waarbij een beveiligde verbinding nodig is wordt er SSL toegepast. SSL zal de “handshake” procedure starten voor het begin van het opzetten van een beveiligde communicatie verbinding.



De “handshake” synchroniseert de server en de cliënt met de encrypty methoden en sleutels die zullen worden gebruikt voor de rest van de communicatie. Dit is ook het gedeelte waar de authentificatie plaatsvindt.
Hieronder wordt in een opsomming gegeven wat er precies gebeurd bij het handshake proces.

1. De gebruikers webbrowser stuurt de webserver zijn methoden voor het coderen van data. Dit bevat het type encryptie, willekeurige data dat de encryptie aan beide kanten kan gebruiken voor eht coderen van routines en andere SSL gerelateerde data.
2. De server controleerd zijn eigen willekeurige data om te gebruiken voor de encryptie en andere SSL informatie. De SSL informatie bevat onder andere het SSL certificaat samen met een string karakters (de publieke sleutel).
3. De browser van de client controleerd de informatie die hij heft ontvangen en vergelijkt het met het domein waarmee hij een verbinding wilde maken. Als dit niet overeenkomt zal de browser een foutmelding geven. Op dit punt wordt ook de geldigheidsdatum en de autorisatie van het certificaat gecontroleerd.
4. De browser zal op dit moment een “premaster secret” aanmaken die wordt gebruikt om de rest dan de sessie te coderen. Het premaster secret is een gecodeerd geheim bericht die wordt gecodeerd met behulp van de overeengekomen encryptie methode en de server zijn publieke sleutel (string).
5. Met behulp van de nieuwe premaster sectret kan zowel de browser als de webserver een “master secret” genereren en deze gebruiken om sessie sleutels te genereren. Deze sleutels worden gebruikt voor het coderen en decoderen van de gegevens tijdens de rest van de sessie. Met de “master secret” sleutel kunnen de browser en de webserver een verifieren dat de gegevens tijdens het verzenden niet zijn gewijzigd.
6. De browser heeft nu de informatie die hij nodig heeft om een beveiligde verbindin op te zetten en geeft aan de server door dat hij zal starten met het gebruik van de sessie sleutel.
7. De browser verifieerd dat de webserver gereed is met het opzetten van de beveiligde sessie.
8. Hierna zal ook de webserver een bericht sturen aan de browser dat hij gebruik zal maken van de sessie sleutel.
9. Ook de webserver controleerd of de browser gereed is met het opzetten van de beveiligde sessie.

Het overige deel van de SSL sessie wordt uitgevoerd met het “master secret” als de sleutel.

9.2.3 Encryptie methoden

Er zijn verschillende vormen van encryptie methoden die kunnen worden gebruikt om gegeven te beveiligen. Hieronder zijn een paar bekende voorbeelden gegeven:

- DES - Digital Encryption Standard
- DSA - Digital Signature Algorithm
- KEA - Key Exchange Algorithm
- MD5 - Message Digest algorithm
- RC2 - Rivest encryption cipher
- RC4 - Rivest encryption cipher
- SHA-1 - Secure Hash Algorithm
- Triple DES - DES used 3 times
- Others

Zowel RC2 als RC4 ondersteunen 128 bit encryptie. Deze zijn in vergelijking tot 40bit encryptie zeer moeilijk te achterhalen.

9.2.4 Certificaat

Een certificaat is een speciaal tekstbestand dat 2 delen bevat: een deel leesbare tekst dat informatie bevat over de eigenaar, de uitgever van het certificaat, enz. en een tweede versleuteld deel (niet leesbaar voor mensen) dat de digitale handtekening bevat en een publieke sleutel van de certificatie authoriteit.
Het tekstbestand krijgt een .cer-extentie zodat het operating systeem het bestand kan open met een (ingebouwde) applicatie die certificaten kan lezen. Als je een certificaat zou openen in een teksteditor dan zou het ongeveer er uit zien als dit :

-----BEGIN CERTIFICATE-----
CBHcm91cCBDQS5jcmwwR
qBEoEKGQGZpbGU6Ly9cX
ENFUlRTUlZcQ2VydFNyd
lxDZXJ0RW5yb2xsXE1TI
ENlcnRTcnYgVGVzdCBHc
m91cCBDQS5jcmwwCQYDV
R0TBAIwADBiBggrBgEFB
QcBAQRWMFQwUgYIKwYBB
QUHMAKGRmh0dHA6Ly9DR
VJUU1JWL0NlcnRTcnYvQ
2VydEVucm9sbC9DRVJUU
1JWX01TIENlcnRTcnYgV
GVzdCBHcm91cCBDQS5jc
nQwDQYJKoZIhvcNAQEEB
QADQQAhq70nRlse0ulPs
tU+lWdjeNj5p
-----END CERTIFICATE-----

9.2 Global.asax

Een global.asax bestand is bekend als het ASP.NET applicatie bestand. Dit is een optioneel bestand met code voor events op applicatie niveau die worden gestart door ASP.NET of een HttpModules. Het global.asax bestand wordt in de root van de ASP.NET gebaseerde applicatie geplaatst. Tijdens de runtime wordt de global.asax gecompileerd in een dynamisch gegenereerde framework klasse uit de HttpApplication basis klasse.
Het bestand is zo geconfigureerd dat als hijzelf wordt opgevraagd, dit verzoek wordt afgewezen. Externe gebruikers kunnen dit bestand niet downloaden of de code ervan bekijken.

Een global.asax bestand kan zowel in notepad worden ontwikkeld als in een voor gecompileerde klasse die beschikbaar is binnen de applicatie (\bin). In het laatste geval heb je nog steeds het global.asax betand nodig om naar te verwijzen naar de geassembleerde versie.

Als het global.asax bestand is aangepast wordt dit gedetecteerd door de ASP.NET page framework. Alle huidige request voor de applicatie ontvangen de Application_OnEnd event en het bestand wordt opnieuw geladen. Dit resulteerd in het effect dat alle browser sessies worden beëindigd en dat alle “state” informatie wordt gewist. Zodra de volgende aanvraag van een browser arriveert zal het ASP.NET page framework het global.asax betand opnieuw compileren en zal het Applciation_OnStart event worden geladen.

9.3 ASP.NET's web.config

Binnen ASP.Net kan er gebruik worden gemaakt van Windows, Forms of Passport authenticatie. Het heeft de mogelijkheid om cookies te gebruiken en gebruikers door te sturen, maar het bevat ook rol gebaseerde beveiliging.

De web.config file wordt geplaatst in website directory en bevat de configuratie van de web applicatie. Het is bestand opgebouwd uit XML dat controle bied over de ASP.NET applicaties.

De configuratie filtert de directory en zijn subdirectories en biedt de controle over de autorisatie, authenticatie, browserver informatie, foutafhandeling etc.

Hieronder is een voobeeld gegeven van de inhoud van een web.config bestand.

```
<configuration>
  <system.web>
    <authentication mode="mode">
    </authentication>
  </system.web>
</configuration>
```

Binnen de <authentication> tags wordt gegeven welke manier van authenticatie ASP.NET moet gebruiken. Dit wordt meegegeven aan de eigenschap “mode”. De mogelijke waarden zijn: Windows, Forms en Passport.

De <authentication> tags werken alleen in de root van de map van de virtuele map, niet in de onderliggende mappen.

9.4 View State, Application State, Session State

Standaard worden vrijwel alle waarden van een ASP.NET control bewaard. Als een pagina wordt verstuurd, blijven de waarde behouden. Een viewstate is onafhankelijk van type browser, cookies, sessie variabelen of applicatie variabele. View state wordt geïmplementeerd in een verborgen veld en wordt automatisch aangemaakt in elke webpagina. Dit kan goed worden gebruikt bij het weergeven van data die net uit een database is gelezen. Zo hoeft de database niet elke keer te worden uitgelezen als de waarden worden opgevraagd.

<%@ Page EnableViewState=”False” %>

In de application state kunnen variabele direct worden opgeslagen. Een variabele die is toegevoegd aan de application state is beschikbaar vanuit elke pagina die wordt uitgevoerd binnen de applicatie. Variabele die zijn opgeslagen in de application state functioneren als globale variabele voor een applicatie. De application state wordt gegeven door de HttpApplicationState klasse. In dit type state kunnen verschillende typen objecten worden opgeslagen zoals bijvoorbeeld datasets.

ASP.NET bevat ook de mogelijkheid om gebruikers te traceren en de bijbehorende gegevens. Dit wordt gedaan met de Session State. Als een gebruiker een pagina opvraagd van een ASP.Net Web site kent de site automatisch een cookie toe aan de browser van de gebruiker. Alle gegevens die aan de Session state worden toegekend blijven beschikbaar zolang een gebruiker gebruikt maakt van de website. De Session state wordt pas verwijderd als een gebruiker 20 minuten niet actief is geweest (geen pagina meer heeft opgevraagd). De gegevens binnen de Session sate zijn specifiek voor een gebruiker. Voor elke gebruiker wordt er een aparte Session state bijgehouden.
Ook in de Sessie state kunnen verschillende typen objecten worden opgeslagen.

10. HttpHandlers en HttpModules.

Omdat er steeds meer ontwikkelaars vraag hebben naar functionaliteiten van Web servers is er ISASPI ontwikkeld. ISAPI is ontwikkeld door Microsoft en geeft als het ware een plug in mogelijkheid voor IIS om webserver functies toe te voegen.

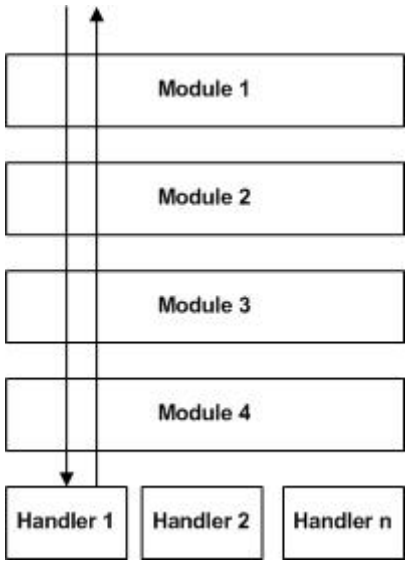
ISAPI bestaat uit 2 belangrijke componenten: ISAPI extensies en ISAPI filters. ISAPI extensies worden geïmplementeerd door Win32 DDL's. Deze extensies worden geladen zodra een http request deze aanvraagt. Een ISAPI filter is een filter welke zich bevindt tussen de web server en de cliënt. Als een cliënt een request doet naar de server, wordt deze request door het filter gefilterd. Filters zijn in staat het request aan te passen, operaties uit te voeren om gegevens te loggen, etc.

Deze ISAPI componenten zijn ingewikkeld om te implementeren omdat deze vaak zeer complex zijn. ASP.NET biedt HttpHandlers en HttpModules.

10.1 ASP.NET request processing

ASP.NET request processing is gebaseerd op een pipeline model waarin ASP.NET de http request doorgeeft aan de modules in de pipeline. Elke module die de http request ontvangt heeft volledige controle over de request. Zodra de request alle modulen heeft gepasseerd wordt het opgevangen door een http handler. De http handler verwerkt de request en geeft het resultaat terug in de pipeline.

Het onderstaande figuur beschrijft het pipeline model.



Er wordt uiteindelijk maar 1 http handler gebruikt en meerdere http modules.

10.2 Http Handlers

Http handlers bevinden zich in de System.Web.IHttpHandler interface. Http handlers zijn te vergelijken met ISAPI extensies. Elke klasse die de IHttpHandler interface implementeert kan zich gedragen als een doel voor een inkomend Http request.

Http handlers beschikken over de Processrequest en de IsReusable methoden. De ProcessRequest methode wordt aangeroepen om de Http request uit te voeren. De IsReusable methode wordt gebruikt om te herkennen of deze instantie van de Http handler kan worden hergebruikt voor andere http requests van hetzelfde type. Http handlers kunnen de waarde true of false teruggeven.

De klassen kunnen worden gerelateerd naar http requests in het bestand web.config of machine.config. Zodra dit is opgegeven wordt een request voor deze handler automatisch doorgestuurd.

10.2.1 Registreren van HTTP Handlers in configuratie bestanden

ASP.NET beheert de configuratie informatie in 2 bestanden. Het web.config bestand en het machine.config bestand. Het machine.config bestand bevat de configuratie settings voor de gehele machine. De web.config is voor een specifieke applicatie of zelfs een submap van een applicatie. Door gebruik te maken van <httpHandlers> en <add> nodes kunnen er Http handlers worden toegekend aan Web applicaties. Hieronder is een voorbeeldgegeven.

```
<httpHandlers>
        <add verb="ondersteunde http verbs" path="pad" type="namespace.klassennaam,
assembleernaam" />
<httpHandlers>
```

- Het attribuut “verb” specificeert de verbs die worden ondersteund door de handler (post, get).
- Het attribuut “path” specificeert het pad naar de bestanden wanneer de handler wordt gebruikt. Met het path attribuut kan dit ook voor een specifiek bestand worden aangegeven (specifiek.abc) of voor een specifieke extensie (*.abc).
- Het attribuut “type” specificeert het type handler door het opgeven van de namespaces, de klassennaam en de assembleernaam.

10.2.2 Session State in http Handlers

Het beheren van de Session State is één van de meest gebruikte taken binnen web applicaties. http handlers hebben ook toegang krijgen tot de Session State. Dit is standaard niet het geval. In verband met de behoefte om de Session state data te lezen of schrijven kan één van het volgende worden geïmplementeerd.

- IRequiresSessionState
- IReadOnlySessionState.

De naam geeft al aan welke er gebruikt wordt bij welke behoefte. De toegang tot de Session state kan als volgt worden gedefinieerd.

```
public class NewHandler : IHttpHandler, IRequiresSessionState
```

10.3 HTTP Modules

http modules zijn de .NET componenten die de System.Web.IHttpModule interface implementeren. Deze componenten voegen zich in de ASP.NET processing pipeline en registreren zich voor bepaalde events. Zodra zo'n event zich voordoet haalt ASP.NET de bijbehorende module erbij voor het uitvoeren van de request.

Het is de bedoeling van een http module dat deze de methoden Init en Dispose implementeerd van de IHttpModule.

- De methode Init geeft een http module de mogelijkheid zich te registreren in het HttpApplication object.
- De methode Dispose geeft een http module de mogelijkheid om objecten op te schonen (verwijderen) zonder dat dit gedetecteerd hoeft te worden.

Een http module kan zich registreren voor de volgende events die beschikbaar zij gesteld door het System.Web.HttpApplication object.

Eventnaam	**Omschrijving**
AcquireRequestState	This event is raised when ASP.NET runtime is ready to acquire the Session state of the current HTTP request.
AuthenticateRequest	This event is raised when ASP.NET runtime is ready to authenticate the identity of the user.
AuthorizeRequest	This event is raised when ASP.NET runtime is ready to authorize the user for the resources user is trying to access.
BeginRequest	This event is raised when ASP.NET runtime receives a new HTTP request.
Disposed	This event is raised when ASP.NET completes the processing of HTTP request.
EndRequest	This event is raised just before sending the response content to the client.
Error	This event is raised when an unhandled exception occurs during the processing of HTTP request.
PostRequestHandlerExecute	This event is raised just after HTTP handler finishes execution.
PreRequestHandlerExecute	This event is raised just before ASP.NET begins executing a handler for the HTTP request. After this event, ASP.NET will forward the request to the appropriate HTTP handler.
PreSendRequestContent	This event is raised just before ASP.NET sends the response contents to the client. This event allows us to change the contents before it gets delivered to the client. We can use this event to add the contents, which are common in all pages, to the page output. For example, a common menu, header or footer.
PreSendRequestHeaders	This event is raised just before ASP.NET sends the HTTP response headers to the client. This event allows us to change the headers before they get delivered to the client. We can use this event to add cookies and custom data into headers.
ReleaseRequestState	This event is raised after ASP.NET finishes executing all request handlers.
ResolveRequestCache	This event is raised to determine whether the request can be fulfilled by returning the contents from the Output Cache. This depends on how the Output Caching has been setup for your web application.
UpdateRequestCache	This event is raised when ASP.NET has completed processing the current HTTP request and the output contents are ready to be added to the Output Cache. This depends on how the Output Caching has been setup for your Web application.

Lostaand van deze events zijn er nog 4 andere events die gebruikt kunnen worden. Deze events kunnen worden geimplementeerd door de methoden te specificeren in het global.asax bestand van de webapplicatie. Dit zijn de volgende events:

- Application_OnStart
  Dit event wordt gestart bij de eerste keer dat een request wordt gemaakt wij de web applicatie.
- Application_OnEnd
  Dit event wordt gestart vlak voor het beeindigen van de applicatie.
- Session_OnStart
  Dit even wordt gestart bij het eerste request voor de gebruikers sessie.
- Session_OnEnd
  Dit event wordt gestart al seen sessie verloopt.

10.3.1 Registreren van HTTP Modules in configuratie bestanden

Als een http module is gebouwd en geplaatst in de bin map van de webapplicatie, moet het deze worden geregistreerd in het web.config of machine.config bestand. De <httpModules> en <add> nodes kunnen worden gebruikt voor het toevoegen van http modules bij de webapplicatie.

Omdat de configuratie gebruik maakt van overerving, kan het voorkomen dat er in de mappen van de “childs” een http module wordt gebruikt die niet gewenst is. Hiervoor kan de <remove> node worden gebruikt. Als alle overgeërfde http modulen verwijderd moeten worden kan er gebruik worden gemaakt van de node <clear>. De volgende code geeft een voorbeeld van het toevoegen van een standaard http module.

```
<httpModules>
        <add type="classname, assemblyname" name="modulename"  />
<httpModules>
```

De volgende code geeft een voorbeeld van het verwijderen van een standaard http module.

```
<httpModules>
        <remove name="modulename"  />
<httpModules>
```

- Het attribuut “type” specificeerd het type van de http module in de vorm van een klasse en assembleernaam.
- Het attribuut “naam” specificeerd de naam voor de module. Deze naam wordt gebruikt ter identificatie voor andere applicaties.

11. Overzicht van de beveiliging

11.1 Inleiding

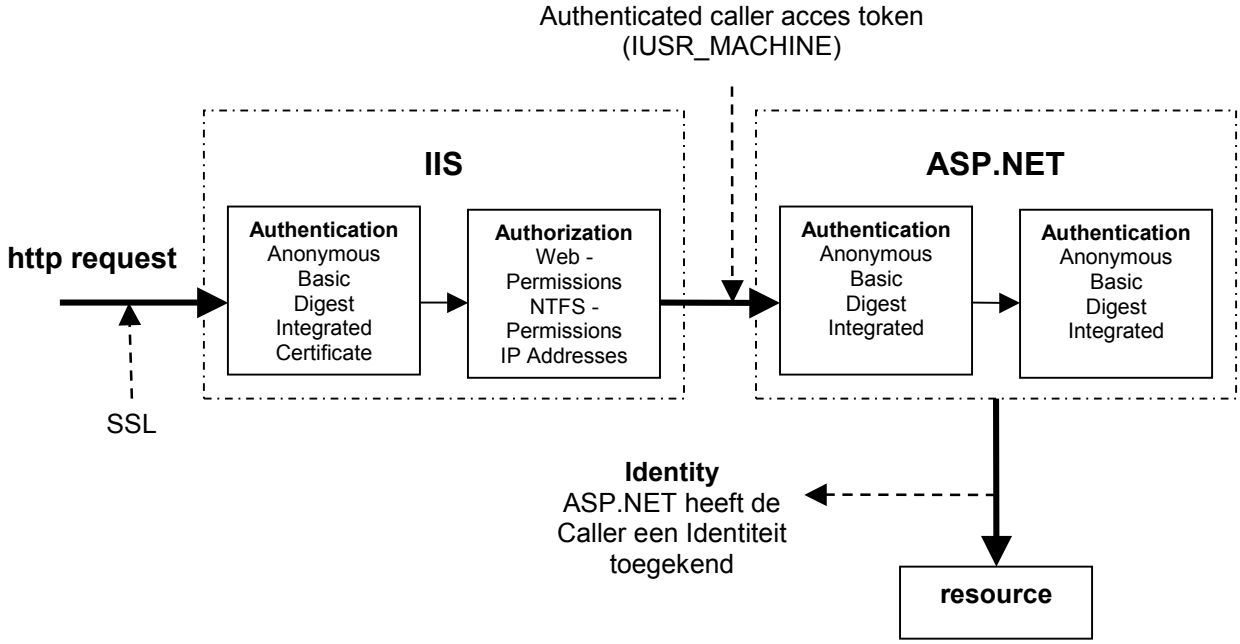
In dit onderdeel van het document worden de onderdelen die zijn onderzocht naast elkaar geplaatst en in schema's verwerkt zodat er kan worden waargenomen hoe de processen verlopen. Ook worden de onderzochte onderdelen gerelateerd aan het te ontwikkelen systeem bij de SMRA. Als eerste zal er worden vastgesteld wat er benodigd is om .NET te kunnen. Daarna zal er worden gekeken naar de vormen van beveiliging die ASP.NET biedt en welke hiervan het meest geschikt zijn voor toepassing binnen de SMRA.

11.2 Wat is er benodigd om .NET te kunnen gebruiken?

Om .NET te kunnen ontwikkelen is er een PC benodigd met Windows XP professional of 2000 benodigd. Windows XP Home is ook mogelijk maar dan moet de ASP.NET WebMatrix worden geïnstalleerd.
Ook moet het .NET framework op de computer worden geïnstalleerd waarop de IIS webservice draait.

11.3 ASP.NET Web Application Security

Als een gebruiker een resource opvraagt waarvoor hij geautoriseerd moet zijn, moet de gebruiker worden geauthentificeerd. Een gebruiker dient met zijn gebruikersnaam en wachtwoord zich te valideren om toegang te verkrijgen tot de aangevraagde resource. Hieronder is de Authentication/Authorization request flow weergegeven.



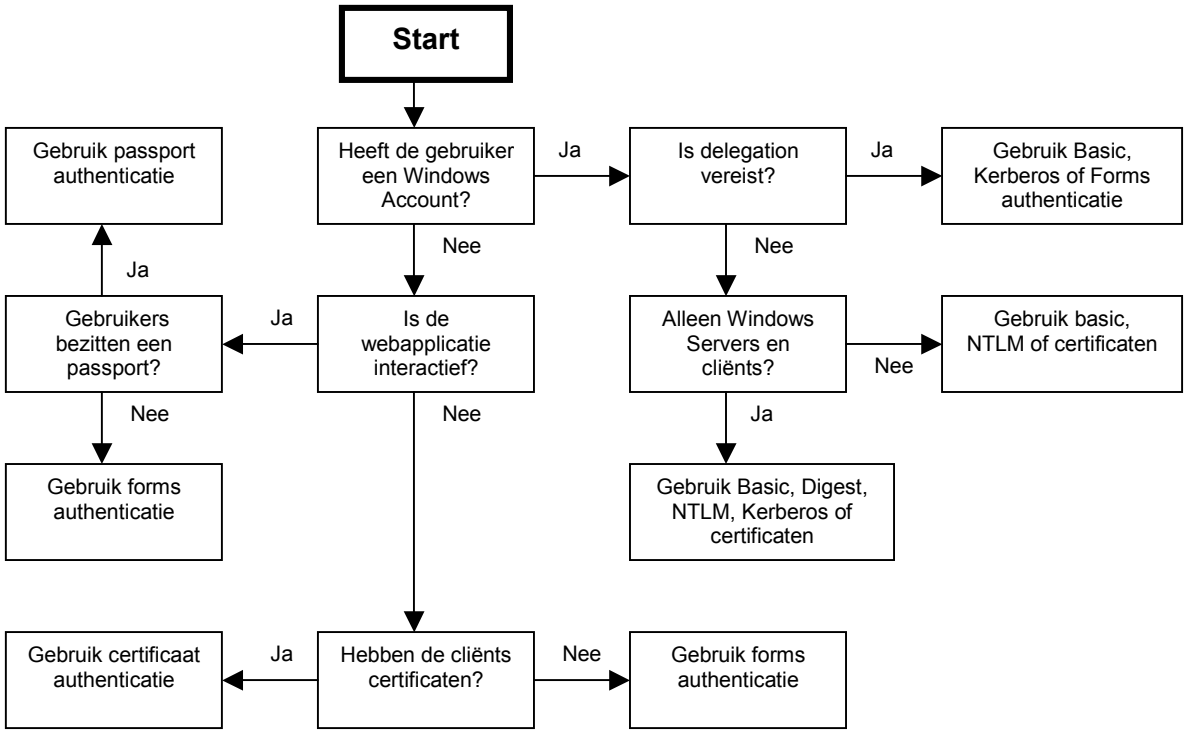
In het figuur is zichtbaar date en gebruiker een http request doet naar een resource. Omdat de gebruiker een beveiligde pagina opvraagt wordt er SSL gebruikt. SSL codeert de gegevens die over het Internet worden verstuurd. Bij het opvragen van een beveiligde pagina krijgt een cliënt een certificaat van de webserver. De cliënt krijgt zo een unieke Identiteit en de gecodeerde gegevens kunnen worden gedecodeerd bij de cliënt. De http request wordt opgevangen door IIS. IIS zal de gebruiker bekend maken als onbekende gebruiker, of zal een authenticatie mechanisme gebruiken zoals Basic, Digest, Integrated of certificate. Het autorisatiedeel van IIS controleert of de gebruiker permissie heeft toe het opvragen van de

pagina, toegang tot een bepaald deel op de harde/schijf (Controle op NTFS permissies) en of een gebruiker een IP adres bezit die geautoriseerd is tot het opvragen van de resource. De autorisatie en authenticatie kan ook plaatsvinden binnen ASP.NET. Binnen ASP.NET zijn er verschillende manieren om dit te doen. Er kan gebruik worden gemaakt van Windows, Forms of Passport authenticatie. Zodra de gebruiker is geauthenticeerd zal het autorisatiedeel van ASP.NET het overnemen en controleren op niveau van bestanden, URL of een .NET gebruikersrol.

ASP.NET kent uiteindelijk aan de Caller (aanvrager) een identiteit toe en bepaald op grond hiervan of de gebruiker de bevoegdheid heeft voor het opvragen van de pagina.

11.3.1 De Authenticatie methode

Er zijn dus verschillende manieren om gebruikers te authenticeren. Om de voordelen, de nadelen mogelijkheden en de beperkingen van de verschillende authenticatie methoden te vergelijken is hieronder een overzicht gegeven. Dit overzicht geeft een “walkthrough” voor het kiezen van een authenticatie methode die toepasbaar is binnen het te ontwikkelen beveiligingssysteem.



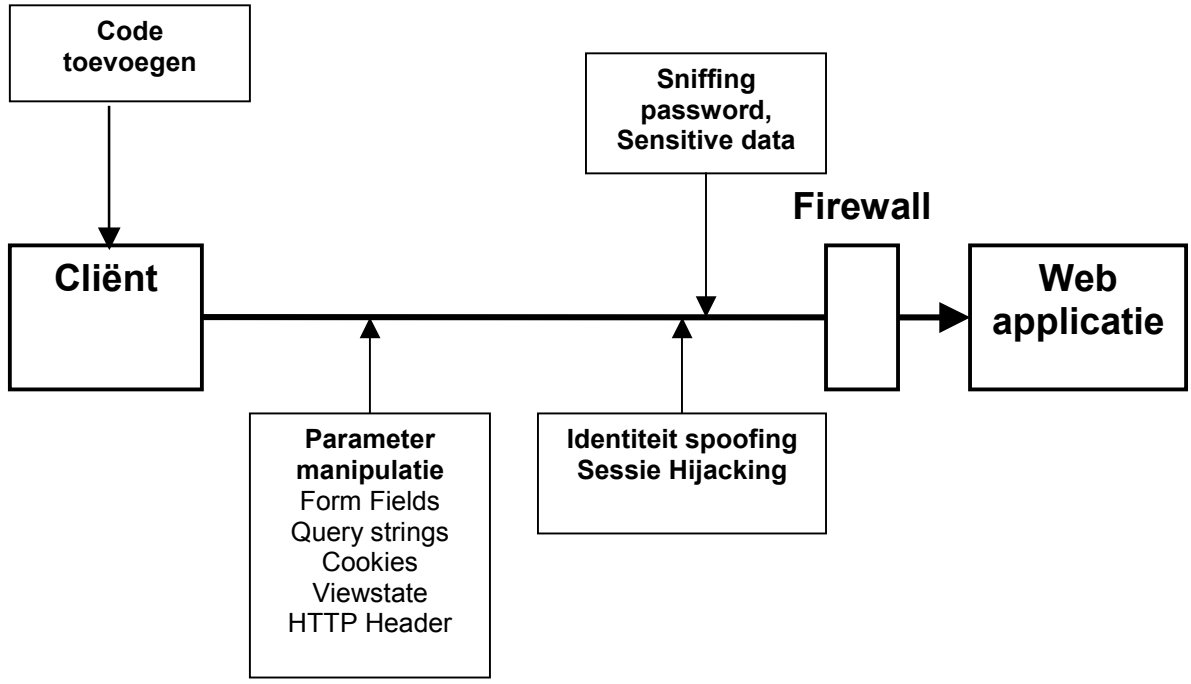
De gebruikers van het systeem zullen niet allemaal een Windows account gebruiken. De voornaamste reden hiervoor is omdat er veel gebruikers gebruik zullen maken van het systeem. Ook moeten er van iedere gebruiker gegevens worden bijgehouden.

De webapplicatie zal interactief met de gebruikers omgaan. Dit houdt in dat de gebruiker zich bekend maakt bij het systeem en dat de pagina’s dynamisch worden gegenereerd op basis van een gebruiker zijn persoonlijk eigenschappen.

De volgende stap is het overwegen van pasporten. Elke gebruiker zou een passport toegewezen krijgen en door een externe instantie worden geauthenticeerd. Dit neemt de nodige kosten en tijd met zich mee. Om deze redenen is er gekozen om gebruik te maken van Forms authentication.

11.4 Standaard aanvallen op een webapplicatie

Wat kunnen zogenoemde hackers nu voor aanvallen plegen op een webapplicatie? Op welke delen in de beveiliging moet aandacht worden gelegd. Hieronder is een schematische weergave gegeven van verschillende manieren waarop een aanval op een website kan worden gepleegd.



In het schema is duidelijk af te leiden dat het gevoeligste onderdeel het de verbinding tussen de cliënt en de firewall is. Parameter manipulatie wordt voorkomen door in de programmeerfase rekening te houden met deze vorm van “hacken”.
Om data sniffing te voorkomen (het uitlezen van data over een netwerkverbinding) dient er een vorm van encryptie te worden gebruikt. Door een goede manier van encrypty te gebruiken wordt het de hacker vrijwel onmogelijk gemaakt om data of wachtwoorden te sniffen (SSL v3).
Een andere manier voor de hacker om aanvallen te doen op de webapplicatie is door zich voor te doen als een andere persoon. Ofwel Identiteit spoofing of Sessie Hijacking. Ook hiervoor kan de maatregel voor het gebruiken van encryptie worden gebruikt.

11.5 Configuratie instellingen

De beveiliging van de website dient te worden geconfigureerd. Dit wordt gedaan in IIS en binnen ASP.NET in de web.config.

11.5.1 IIS to Secure Communication

IIS wordt ingesteld op “anonymous authentication”. Zodra een gebruiker door IIS wordt doorgestuurd naar ASP.NET wordt hem de identiteit gegeven die wordt toegekend aan een anonieme gebruiker (IUSR_MACHINE).

Er moet een web server worden geïnstalleerd en er moet een certificaat worden aangevraagd. De aangevraagde certificaat moet worden ingesteld voor de verbinding met de beveiligde website.

Hiernaast moeten ook de nodige NTFS persmissies van de IUSR_MACHINE gebruiker worden ingesteld.

11.5.2 ASP.NET Settings in Web.config

Binnen ASP.NET wordt de Web.Config gebruikt voor de configuratie van de webapplicatie. Binnen dit bestand moet worden aangegeven van welke authenticatie mode er gebruik wordt gemaakt.

```
<authentication mode="Forms" />
```

Ook dient impersonate te worden aangezet in verband met de “trusted connectie” met de SQL webserver.

```
<identity impersonate="true" />
```

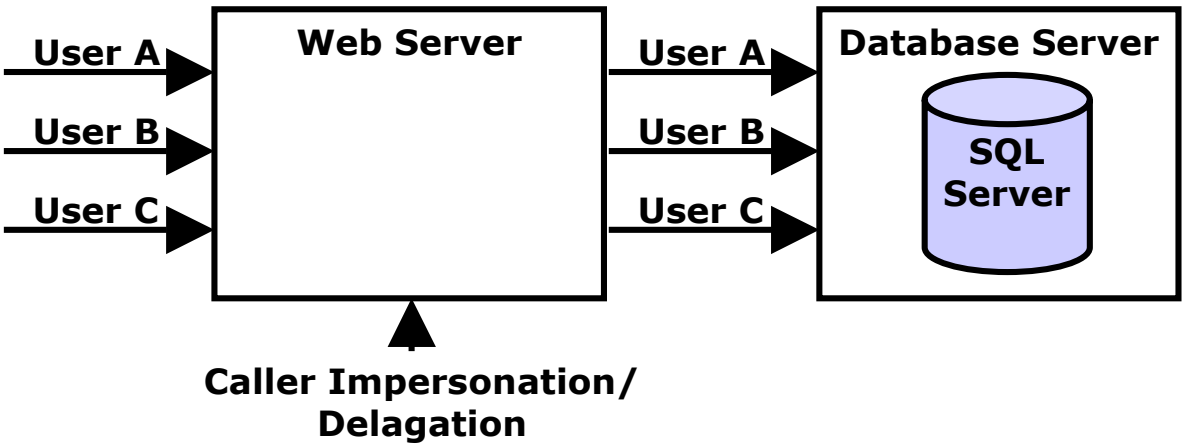
Als laatste wordt het autorisation deel gerealiseerd.

```
<authorization>
    <deny users="*" />
</authorization>
```

Door deze gegevens te plaatsen in het web.config wordt de configuratie ingesteld op Forms authentication, waarbij alle gebruikers geen autorisaties hebben indien ze niet geauthenticeerd en geautoriseerd zijn .
Door impersonate=”true” in de web.config op te geven wordt de identiteit van IIS doorgegeven aan ASP.NET.

11.6 Impersonation

Als impersonation wordt gebruikt met een anonieme gebruiker zal ASP.NET de persoon koppelen aan het account dat is gespecificeerd door IIS. Standaard is dit de "IUSR_MACHINE" account welke niet kan worden gebruikt met delagation. Dit komt omdat het wachtwoord van de IUSR_MACHINE account wordt beheerd door IIS.
Om delegation te kunnen gebruiken moet IIS worden geconfigureerd met een lokaal account dat identiek is aan een account (zelfs het wachtwoord) op een remote systeem. Als IIS geconfigureerd is om een account van een domein te gebruiken, is delegation mogelijk.
Hieronder is een figuur weergegeven met daarin een schematische weergave van Impersonation met delagation.



Omdat er binnen de beveiligingsapplicatie bij de SMRA een verbinding is vereist met een SQL database zal er gebruik worden gemaakt van Impersonation en delegation (trusted connectie).

11.7 Authorization

Mechanisme nog niet bekend.

12. Conclusie

Het .NET framework bestaat uit nieuwe technologieën. Deze technologieën zijn specifiek gericht op het verbinden van mensen, apparaten en informatie. Dit is de kernactiviteit van de SMRA. Ook is het een gestandaardiseerde omgeving die voor het ontwikkelen van zeer verschillende toepassingen kan worden gebruikt (van een website tot een volledige Windows applicaties). Geïntegreerd in .NET zitten vele objecten waarvan gebruikt kan worden gemaakt tijdens het ontwikkelen. Dit maakt het mogelijk, zodra de programmeurs bekend zijn in de omgeving, om op een snel tempo applicaties te ontwikkelen.

Een ander belangrijk onderdeel is er binnen .NET een CLR wordt gebruikt. Deze CLR maakt het mogelijk om binnen de .NET omgeving te programmeren in verschillende talen. De CLR wordt standaard geleverd met een compiler voor Visual Basic, welke binnen de SMRA wordt gezien als standaard programmeertaal.

De beveiligingsmogelijkheden binnen ASP.NET zijn zeer uitgebreid en zoals is gebleken uit hoofdstuk 10 (de beveiliging binnen het SMRA systeem) bieden deze voldoende mogelijkheden voor de beveiliging. Mochten de geïntegreerde klassen uit de library niet voldoen wordt de functionaliteit geboden om zelf klassen te ontwikkelen die gebruikt kunnen worden binnen ASP.NET.

Geraadpleegde literatuur

Website: Anything goes ASP v3.0
URL: http://authors.aspalliance.com/wisemonk/
Omschrijving: Website met diverse informatie over ASP.NET.

Website: Microsoft
URL: http://www.microsoft.com/netherlands/msdn/products/adonet.asp
Omschrijving: Microsoft zelf over het ADO.NET onderdeel uit het .NET framework.

Website: Microsoft
URL: http://www.microsoft.com/net/business/bizbene.asp
Omschrijving: Microsoft over de betekenis van .NET voor bedrijven.

Website: Microsoft GotDotNet Web site
URL: www.gotdotnet.com
Omschrijving: Leren werken met ASP.NET en C#. Site wordt aangeboden door Microsoft.

Website: Lizatec
URL: http://www.lizatec.com/LIZATEC/TECHTRENDS/TOEKOMSTNET
Omschrijving: Een programmeur geeft een verslag met zijn mening over de .NET omgeving.

Website: Microsoft
URL: http://msdn.microsoft.com/netframework/technologyinfo/overview/default.aspx
Omschrijving: Microsoft over .NET framework

Website: Microsoft TechNet Security
URL: www.microsoft.com/technet/security
Omschrijving: Microsoft over de beveiliging van .NET

Website: Microsoft Security Model
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconaspnetdelegation.asp
Omschrijving: Microsoft over beveiliging binnen .NET

Website: What?is
URL: http://whatis.techtarget.com/definition/0,289893,sid9_gci343029,00.html
Omschrijving: Hier wordt de definitie van SSL gegeven samen met links naar andere informatiebronnen voor specifieke informatie over SSL.

Boek: ASP.NET Unleashed (Stephen Walther)
ISBN: 0 672 32542 x
Omschrijving: ASP.NET 1.1, Code in C# en Visual Basic, bouwen van veilige, dynamische webpagina's.

Boek: Microsoft ASP.NET applicatieontwerp (Douglas J. Reilly)
ISBN: 90 395 1970 6
Omschrijving: Boek van Academic service Microsoft (Reilly).

Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Document**
  Definitiestudie

- **Auteur**
  Mark van Houten (20015081)

- **Datum**
  26 juli 2004

- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag

- **Opleidingsvariant**
  MBIT

- **Periode**
  November 2003 – april 2004

- **Opdrachtgever**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft

- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Voorwoord

Voor u ligt het rapport Definitiestudie ten behoeve van een Web based beveiligingssysteem met functie en klantbeheer voor de SMRA. Dit rapport is tot stand gekomen in overleg met dhr R. Mehlkopf en dhr C. Schaaf.

De definitiestudie dient als basis voor de pilotontwikkeling. Elke keer als er een pilot wordt opgeleverd wordt de definitiestudie opnieuw doorlopen. Tijdens het doorlopen wordt de definitiestudie, waarnodig, uitgebreid of verfijnd.

Het uiteindelijke doel van de definitiestudie is het bevestigen van de haalbaarheid van het project, het opstellen van een functioneel systeemconcept en het bereiken van een gezamenlijke overeenstemming over het te volgen ontwikkelscenario.

Delft, SMRA, 26 juli 2004

Inhoudsopgave

1 Inleiding

In de fase definitiestudie zal een onderzoek worden gedaan naar de mogelijkheden van een Web based beveiligingssysteem en zal worden vastgesteld welke functionaliteiten het systeem zal gaan bevatten.

Het eerste onderdeel is het realiseren van een gedetailleerder plan van aanpak. Het plan van aanpak voor de fase definitiestudie fungeert als een detaillering van het algemene plan van aanpak over de definitiefase.

De systeemeisen en wensen worden vastgelegd. Deze eisen en wensen vormen een weergave van de behoefte van de organisatie. Deze eisen en wensen krijgen een prioriteit toegewezen en dienen als basis voor de ontwerpactiviteiten.

Met behulp van de systeemeisen en wensen kan er een systeemconcept worden ontwikkeld. Het systeemconcept wordt gemodelleerd op functioneel niveau vanuit het gezichtspunt van de gebruiker.

Ook wordt er gekeken of de technische structuur van het bedrijf voldoet aan de eisen van het systeem.

Het laatste onderdeel is het opstellen van een pilotplan. In het pilotplan wordt het te ontwikkelen systeem onderverdeeld in verschillende pilots die betrekking hebben op een bepaald deel van het systeemconcept.

2 Plan van aanpak

2.1 Inleiding

Het eerder opgestelde plan van aanpak wordt in deze activiteit, gericht op de definitiefase, gedetailleerd. Het doel van dit plan van aanpak is het vastleggen van afspraken over de inhoud en omvang van de op te leveren producten en de uit te voeren werkzaamheden binnen de definitiestudie.

2.2 Resultaten definitiestudie

Op basis van de resultaten van de definitiestudie kan worden bepaald welke producten (voor projectuitvoering en management) moeten worden opgeleverd en welke activiteiten daarvoor nodig zijn. Hieronder zijn de producten die tijdens de definitiefase worden opgeleverd gegeven:

- Planning definitiestudie
- Pilot evaluatie per pilot
- Systeemeisen
- Systeemconcept
- Technische structuur
- Pilotontwikkelplan

2.3 fasen definitiestudie

De definitiestudie bestaat uit een reeks van fases. Hieronder zijn de fases weergegeven met de daarbij behorende belangrijkste activiteiten:

1. **Pilot evaluatie (per pilotdeel);**
   - Pilot evaluatierapport met feedback (indien er al een pilot is ontwikkeld);
   - Vergadering.
2. **Systeemeisen;**
   - Vergadering;
   - Mindmap;
   - MoSCoW.
3. **Systeemconcept;**
   - Specificeren globaal actoren;
   - Specificeren globaal events;
   - Specificeren globaal gegevensmodel;
   - Systeeminterfaces vaststellen;
   - Vergadering.
4. **Technische structuur;**
   - Controleren of het systeemconcept toepasbaar is in de technische omgeving;
   - Vastleggen technische veranderingen;
   - Opstellen aanschaflijst.
5. **Pilotontwikkelplan.**
   - Vastleggen pilotstructuur;
   - Structureren pilotontwikkeling;
   - Opstellen globaal invoerplan;
   - Opstellen globaal pilot ontwikkelplan.

2.4 Planning definitiestudie

In de globale planning is er 110 uur gepland voor de definitiefase. In deze periode dienen de op te leveren producten uit hoofdstuk 2.2 gerealiseerd te worden. Aan het opstellen van de planning is 8 uur besteed en er blijft nog 102 uur over voor de overige activiteiten.

Fase	Minimaal	Maximaal	Gemiddeld	Planning
Pilot evaluatie (per pilot)	8	14	9	10
Systeemeisen	14	22	19	18
Systeemconcept	25	40	30	32
Technische structuur	14	23	17	18
Pilotontwikkelplan	20	28	25	24

(minimaal + maximaal + 2 * gemiddeld) / 4 = planning)

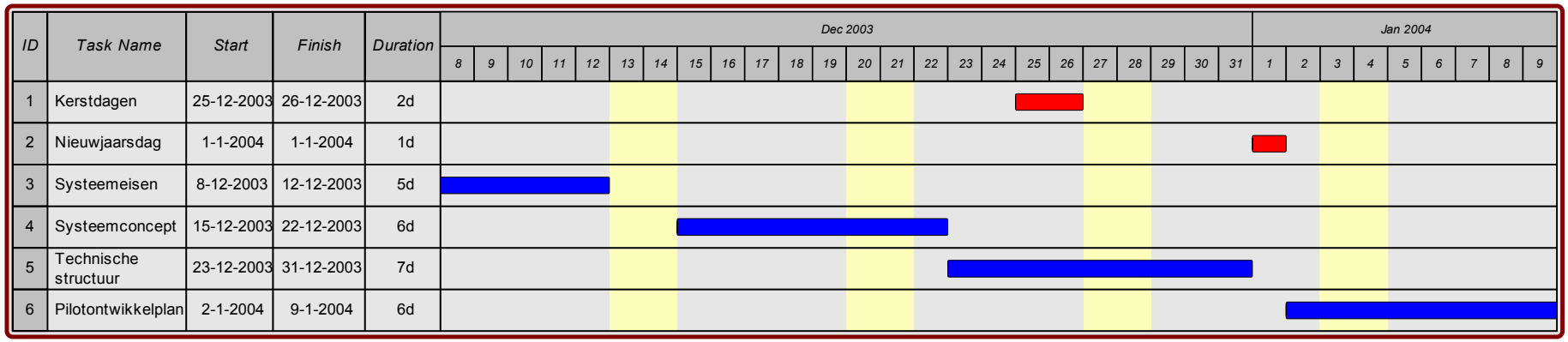
Tijdens deze fase zullen alle onderdelen 1 maal volledig worden gerealiseerd en wegens het gebruik van IAD zal deze fase meerdere malen worden doorlopen en verfijnd of uitgebreid. De tijd die voor de onderdelen in het overzicht is gegeven is dan ook de totaal verwachte tijd die aan elk onderdeel besteed zal worden.

De Pilot evaluatie vindt plaats na de ontwikkeling van elke pilot.

2.5 Overzicht planning definitiestudie

Hieronder is een schema weergegeven met hierop aangegeven wanneer er aan welke fasen wordt gewerkt. Ook wordt aangegeven op welke datum het onderdeel verwacht wordt, gereed te zijn.

ID	Task Name	Start	Finish	Duration
1	Kerstdagen	25-12-2003	26-12-2003	2d
2	Nieuwjaarsdag	1-1-2004	1-1-2004	1d
3	Systeemeisen	8-12-2003	12-12-2003	5d
4	Systeemconcept	15-12-2003	22-12-2003	6d
5	Technische structuur	23-12-2003	31-12-2003	7d
6	Pilotontwikkelplan	2-1-2004	9-1-2004	6d



De Fase Pilot evaluatie is niet in dit onderdeel opgenomen omdat deze fase pas zal worden uitgevoerd nadat er een pilot is gerealiseerd.

In dit overzicht is de activiteit eindverslag niet meer opgenomen aangezien er eerder is opgemerkt dat er tijdens het gehele traject aan dit verslag gewerkt wordt.

3 Pilot evaluatie

3.1 Inleiding

In dit onderdeel van het rapport worden de ervaringen met eerder pilots getoetst in relatie tot de actuele systeemeisen en het oorspronkelijke pilotplan. De resultaten worden tijdens de vergaderingen besproken en eventuele wijzigingen worden doorgevoerd in de systeemeisen, systeemconcept of het pilotplan. De resultaten van de vergaderingen met betrekking tot de pilots worden in dit onderdeel vastgelegd.

3.2 Resultaten pilot 1

In het eerste pilotdeel is de beveiligingsapplicatie ontworpen en ontwikkeld. Er is gestart met het functionele ontwerp. In het functionele ontwerp is de functionele inhoud van de pilot bepaald. Het systeemconcept dat is gegeven in de voorgaande definitiestudie is hier uitgebreid.
Op basis van het functionele ontwerp is het technische ontwerp gerealiseerd. In het technische ontwerp zijn de technische aspecten rond de pilotontwikkeling gegeven. Hierbij zijn besproken: de fysieke allocatie, het technischopslagmodel en de externe interfaces. Hierna is er een pilotontwikkelplan gemaakt waarin is omschreven hoe de pilot is opgebouwd. De pilot is hier opgedeeld in pilotdelen en de bouweenheden zijn hier bepaald. Als volgt zijn er ontwerpen gemaakt van de softwarebouweenheden en zijn er testspecificaties opgesteld. Na het bouwen van de softwarebouweenheden zijn deze getest met behulp van de unit testsets. Het uitvoeren van de unittest verliep vrijwel vlekkeloos. Uiteindelijke zijn de software-bouweenheden samengevoegd en heeft er een regressiontest plaats gevonden. De in de regressiontest aangetroffen “fouten” zijn verholpen maar bij sommige testen zijn er opmerkingen geplaatst.
Ook heeft er een functionaliteitentest plaatsgevonden om te controleren of alle genoemde onderdelen volgens de definitiestudie zijn verwerkt in de eerste pilot. Alle functionaliteiten zijn verwerkt behalve de koppeling met de main database en de functie voor het synchroniseren van de kerkelijke gemeenten.

Na het afronden van de eerste pilot is er een pilotworkshop gehouden waarin het product van de eerste pilot is gepresenteerd. De voornaamste belangen van deze workshop was het verkrijgen van feedback op het systeem en de gebruikers een beeld te geven of de mogelijkheden van het systeem.

Het beveiligingssysteem is afgerond en de basis is gelegd voor het bouwen van het functie en gebruikersbeheer in pilot 2.

3.3 Resultaten pilot 2

Nog niet volledig afgerond.

3.4 Resultaten pilot 3

Nog niet afgerond.

4 Analyse eisen/wensen

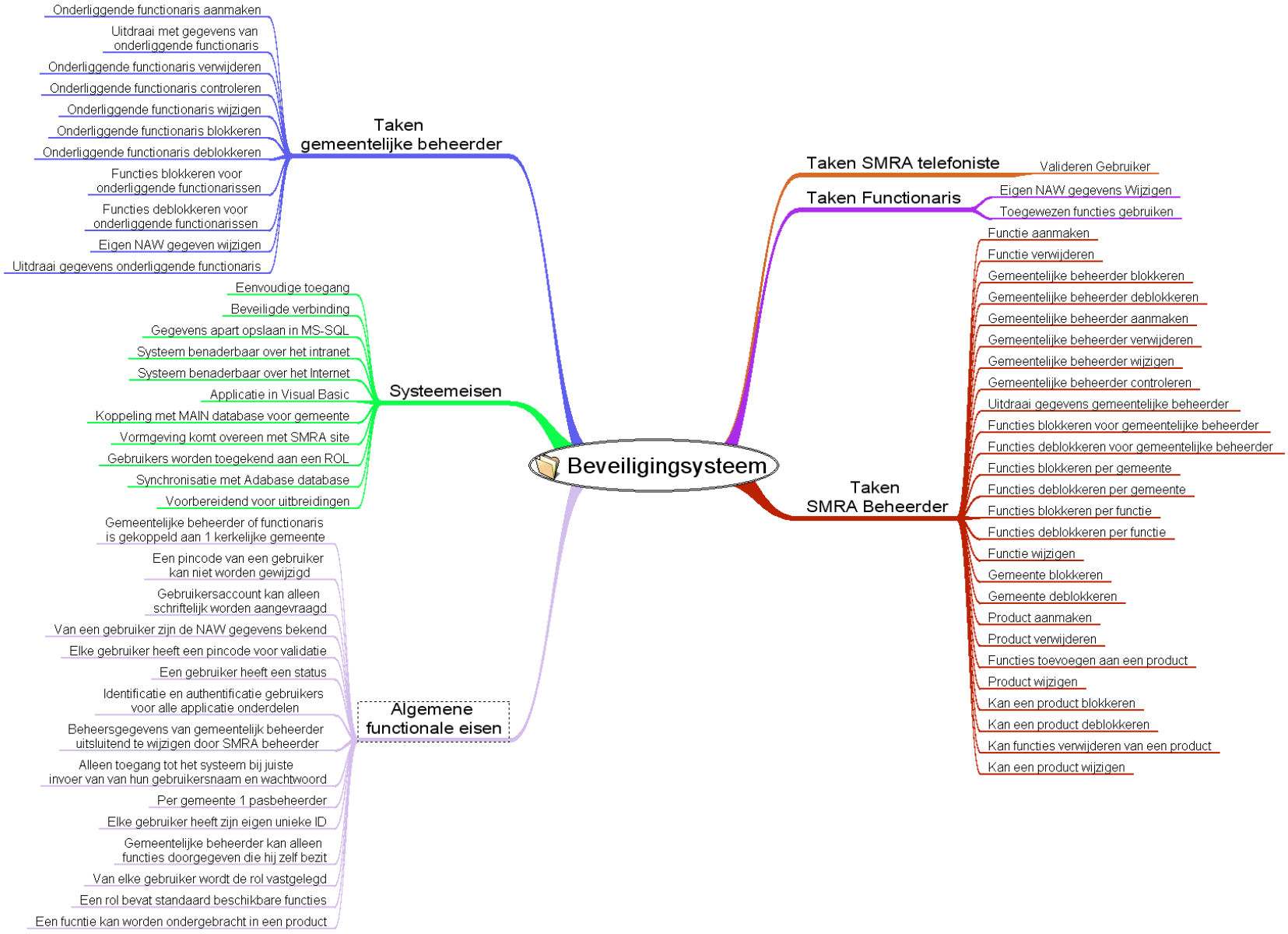
4.1 Inleiding

In dit onderdeel van het document is er een lijst opgesteld met eisen en wensen van de opdrachtgever. Er heeft een brainstormsessie plaats gevonden en de resultaten hiervan zijn verwerkt in een Mindmap. De wensen die in de mindmap zijn gegeven zijn daarna volgens MoSCoW geprioriteerd en de systeemeisen zijn vastgelegd. De eisen en wensen geven een de definitie van de eisen en wensen, en levert een juiste weergave van de behoefte van de organisatie. Hieruit komen lijsten voort die de basis vormen voor de ontwerpactiviteiten.

4.2 Analyse eisen/wensen

De systeemeisen/wensen zijn vastgelegd met behulp van de mindmap techniek. De mindmap is in eerste instantie gebaseerd op de opdrachtomschrijving. De hieruit voortgekomen mindmap is besproken bij een vergadering en is hierbij gewijzigd.

De mindmap geeft weer wat de opdrachtgever van het systeem verwacht. Per gebruiker is bedacht welke handelingen hij zou willen uitvoeren met behulp van het systeem. Deze zijn in het overzicht vastgelegd als taken. Ook zijn de algemene functionele eisen en systeemeisen die gesteld zijn aan het systeem hierin vastgelegd.
De uiteindelijke mindmap is op de volgende pagina weergegeven.


Beveiligingsysteem
Taken gemeentelijke beheerder
Onderliggende functionaris aanmaken
Uitdraai met gegevens van onderliggende functionaris
Onderliggende functionaris verwijderen
Onderliggende functionaris controleren
Onderliggende functionaris wijzigen
Onderliggende functionaris blokkeren
Onderliggende functionaris deblokkeren
Functies blokkeren voor onderliggende functionarissen
Functies deblokkeren voor onderliggende functionarissen
Eigen NAW gegeven wijzigen
Uitdraai gegevens onderliggende functionaris
Systeemeisen
Eenvoudige toegang
Beveiligde verbinding
Gegevens apart opslaan in MS-SQL
Systeem benaderbaar over het intranet
Systeem benaderbaar over het Internet
Applicatie in Visual Basic
Koppeling met MAIN database voor gemeente
Vormgeving komt overeen met SMRA site
Gebruikers worden toegekend aan een ROL
Synchronisatie met Adabase database
Voorbereidend voor uitbreidingen
Algemene functionale eisen
Gemeentelijke beheerder of functionaris is gekoppeld aan 1 kerkelijke gemeente
Een pincode van een gebruiker kan niet worden gewijzigd
Gebruikersaccount kan alleen schriftelijk worden aangevraagd
Van een gebruiker zijn de NAW gegevens bekend
Elke gebruiker heeft een pincode voor validatie
Een gebruiker heeft een status
Identificatie en authentificatie gebruikers voor alle applicatie onderdelen
Beheersgegevens van gemeentelijk beheerder uitsluitend te wijzigen door SMRA beheerder
Alleen toegang tot het systeem bij juiste invoer van van hun gebruikersnaam en wachtwoord
Per gemeente 1 pasbeheerder
Elke gebruiker heeft zijn eigen unieke ID
Gemeentelijke beheerder kan alleen functies doorgegeven die hij zelf bezit
Van elke gebruiker wordt de rol vastgelegd
Een rol bevat standaard beschikbare functies
Een fucntie kan worden ondergebracht in een product
Taken SMRA telefoniste
Valideren Gebruiker
Taken Functionaris
Eigen NAW gegevens Wijzigen
Toegewezen functies gebruiken
Taken SMRA Beheerder
Functie aanmaken
Functie verwijderen
Gemeentelijke beheerder blokkeren
Gemeentelijke beheerder deblokkeren
Gemeentelijke beheerder aanmaken
Gemeentelijke beheerder verwijderen
Gemeentelijke beheerder wijzigen
Gemeentelijke beheerder controleren
Uitdraai gegevens gemeentelijke beheerder
Functies blokkeren voor gemeentelijke beheerder
Functies deblokkeren voor gemeentelijke beheerder
Functies blokkeren per gemeente
Functies deblokkeren per gemeente
Functies blokkeren per functie
Functies deblokkeren per functie
Functie wijzigen
Gemeente blokkeren
Gemeente deblokkeren
Product aanmaken
Product verwijderen
Functies toevoegen aan een product
Product wijzigen
Kan een product blokkeren
Kan een product deblokkeren
Kan functies verwijderen van een product
Kan een product wijzigen

4.3 Functionele eisen/wensen

Hieronder functionele eisen en wensen van het te ontwikkelen systeem gegeven. Door middel van de MoSCoW regels wordt aangegeven of de gewenste functies uiteindelijk in het beveiligingssysteem verwerkt zullen worden.

Hierbij staat een: **M** voor **Must Have**
S voor **Should Have**
C voor **Could Have**
W voor **Would be nice to Have**

De functionele eisen die als Must Have zijn aangegeven zullen dan uiteindelijk ook in de applicatie verwerkt moeten zijn. De overige functionele eisen zullen aan de hand van hun prioriteit verwerkt worden in de applicatie totdat de applicatie aan alle functionele eisen voldoet of totdat de beschikbare periode ten einde is gekomen.

4.3.1 Functionele eisen/wensen telefoniste

NR	Functionele eis	MoSCoW
F1	Telefoniste dient een gebruiker te valideren door zijn pincode te verifiëren	**M**

4.3.2 Functionele eisen/wensen functionaris

NR	Functionele eis	MoSCoW
F2	Mag zelf zijn NAW gegevens wijzigen	**M**
F3	Is in staat zijn toegewezen functies te gebruiken	**M**

4.3.3 Functionele eisen/wensen SMRA Beheerder

NR	Functionele eis	MoSCoW
F4	Is in staat een functie aan te maken	**M**
F5	Is in staat een functie te verwijderen	**M**
F6	Is in staat om een gemeentelijke beheerder te blokkeren	**M**
F7	Is in staat om een gemeentelijke beheerder te deblokkeren	**M**
F8	Is in staat om een gemeentelijke beheerder aan te maken	**M**
F9	Is in staat om een uitdraai met gegevens maken van een gemeentelijke beheerder	**M**
F10	Is in staat om een gemeentelijke beheerder te verwijderen	**M**
F11	Is in staat om een gemeentelijke beheerder te wijzigen	**M**
F12	Is in staat om een gemeentelijke beheerder te controleren	**M**
F13	Is in staat functies voor een gemeentelijke beheerder te blokkeren en deblokkeren	**M**
F14	Is in staat functies te blokkeren per functie	**S**
F15	Is in staat functies te deblokkeren per functie	**S**
F16	Is in staat een functie te wijzigen	**C**
F17	Is in staat om een gehele gemeente (klant) te blokkeren (alle functies)	**C**
F18	Is in staat om een gehele gemeente (klant) te deblokkeren (alle functies)	**C**
F45	*Is in staat om een product aan te maken*	***M***
F46	*Is in staat om een product te wijzigen*	***M***
F47	*Is in staat om een product te verwijderen*	***M***
F48	*Is in staat een product te blokkeren*	***M***
F49	*Is in staat een product te deblokkeren*	***M***
F50	*Is in staat functies toe te wijzen aan een product*	***M***
F51	*Is in staat functies te verwijderen van een product.*	***M***

4.3.4 Functionele eisen/wensen gemeentelijke beheerder

NR	Functionele eis	MoSCoW
F19	Kan een onderliggende functionaris aanmaken	**M**
F20	Kan een uitdraai met gegevens maken van een onderliggende functionaris	**M**
F21	Kan een onderliggende functionaris verwijderen	**M**
F22	Kan een onderliggende functionaris wijzigen/controleren	**M**
F23	Kan functionarissen uit de betreffende gemeente blokkeren	**S**
F24	Kan functionarissen uit de betreffende gemeente deblokkeren	**S**
F25	Een gemeentelijke beheerder kan functies blokkeren en deblokkeren voor onderliggende functionarissen	**C**
F26	Kan zijn eigen adresgegevens wijzigen	**C**

4.3.5 Algemene functionele eisen

NR	Functionele eis	MoSCoW
F27	Een gemeentelijke beheerder of functionaris is altijd gekoppeld aan 1 kerkelijke gemeente (werkingsgebied)	**M**
F28	Een gemeentelijke beheerder, functionaris of een SMRA beheerder kan een pincode niet wijzigen	**M**
F29	Een gebruikersaccount kan alleen schriftelijk worden aangevraagd bij een gemeentelijke pasbeheerder of een SMRA beheerder	**M**
F30	Een pincode kan alleen gewijzigd worden door een gebruiker te verwijderen en opnieuw aan te maken (schriftelijk aanvragen met opgave van oude toegangspas en pincode)	**M**
F31	Van een gebruiker worden de bijbehorende persoonlijke gegevens opgeslagen	**M**
F32	Elke gebruiker heeft een Pincode waarmee hij gevalideerd kan worden	**M**
F33	Elke gebruiker heft een status (geblokkeerd of niet geblokkeerd)	**M**
F34	De gemeente die bij een gebruiker hoort, mag niet worden gewijzigd	**M**
F35	Van een gebruiker moet een print kunnen worden gemaakt die kan worden verstuurd als brief naar een gemeentelijke beheerder of functionaris. Hierop is zijn de mogelijke functies, de persoonlijke gegevens, de toegangsgegevens en de pincode gegeven.	**M**
F36	Identificatie gebruikers voor alle applicatie onderdelen	**M**
F37	Authentificatie gebruikers voor alle applicatie onderdelen	**M**
F38	De beheergegevens van de gemeentelijke pasbeheerders zijn uitsluitend te wijzigen door de SMRA.	**M**
F39	De gemeentelijke beheerders en de lokale beheerders krijgen uitsluitend toegang tot de website door het correct invoeren van hun toegangspascode en hun pincode	**M**
F40	Per gemeente maximaal 1 pasbeheerder	**S**
F41	Elke gebruiker heeft zijn eigen unieke ID	**M**
F42	Elke functie wordt ondergracht onder een product	**M**
F43	Een gemeentelijke beheerder kan alleen de functies die hij zelf bezit doorgeven aan onderliggende functionarissen, exclusief het aanmaken van lokale beheerders	**C**
F44	Per gemeentelijke beheerder een maximaal aantal aan te maken gebruikers.	**C**

4.4 Technische eisen

NR	Eis
E1	Eenvoudige toegang voor alle applicatie onderdelen
E2	Gebruik maken van een beveiligde verbinding (SSL of dergelijke)
E3	Alle gegevens worden opgeslagen in een aparte MS-SQL database
E4	Het systeem dient op te vragen te zijn over het internet
E5	De applicatie dient geschreven te worden gebruikt maken van ASP.net (VB)
E6	Het systeem dient op te vragen te zijn over het intranet
E7	Koppeling met de main-database voor het opvragen van gemeente
E8	De vormgeving van de webapplicatie moet overeenkomen met de huidige website van SMRA
E9	Een product kan bestaan uit 1 of meerdere functies. Een product kan binnen het systeem worden gezien als een groepering van functies.
E10	Synchronisatie mogelijkheid met de main-database
E11	Het systeem moet voorbereidend zijn voor uitbreidingen

4.5 Eisen/wensen tijdens de uitvoering van pilot 1

NR	Eis/wens
E1	Grafische hyperlinks in het menu?
E2	Geblokkeerder producten en functies toch zichtbaar, maar niet uitvoerbaar. Als deze worden gekozen door de gebruiker dient te worden weergegeven door wie hij is geblokkeerd.
E3	Login naam is beperkt. Zijn vele personen met zelfde voorletters of achternaam. Wellicht de mogelijkheid van het kiezen van een kerkelijke gemeente.
E4	Gebruikerstatistieken bijhouden?
E5	Sessie beëindigen van een specifieke gebruiker?
E6	Automatische e-mail aan de gebruiker bij het aanmaken van een nieuwe gebruiker.

4.6 Eisen/wensen tijdens de uitvoering van pilot 2

NR	Eis/wens

4.7 Eisen/wensen tijdens de uitvoering van pilot 2

NR	Eis/wens

5 Systeemconcept

5.1 Inleiding

In dit onderdeel van het document wordt het systeemconcept gegeven. Het systeemconcept is gebaseerd op de eerder opgestelde systeemeisen. Het systeemconcept is een beschrijving op functioneel niveau van de oplossing. Hierin licht een functionele architectuur besloten die de functionele componenten en hun afhankelijkheden aangeeft. Deze beschrijving wordt gemodelleerd uit het gezichtspunt van de gebruikers.

Als eerste is er een tekstuele beschrijving gegeven van het concept achter het systeem. Daarna wordt het systeemconcept gedetailleerder benaderd met behulp van UML technieken. In dit systeemconcept is er gebruik gemaakt van UML. De actoren zijn bepaald en per actor zijn de functies bepaald. Deze gegevens zijn verwerkt in Use-cases. De Use-cases geven de functionele eisen weer. Een Use-case geeft aan hoe een gebruiker het systeem zou willen bedienen.

5.2 Tekstuele beschrijving

Het primaire doel van het systeem is het beschikbaar stellen van diensten over het Internet. Omdat veel van deze diensten persoonlijke gegevens bevatten dient het systeem te beschikken over een beveiligingssysteem. De gegevens die met het beveiligingssysteem beschikbaar worden gesteld moeten veilig over het Internet worden verstuurd. Hiervoor is een beveiligde verbinding met een encryptie mechanisme noodzakelijk.

Het beveiligingssysteem dient universeel toepasbaar te zijn binnen de SMRA. Dit wil zeggen dat het systeem de mogelijkheid moet geven om de huidige diensten aan te bieden over het Internet. Hiervoor dienen uiteraard wel de Windows applicaties te worden omgezet naar webapplicaties. De functionaliteiten van de Windows applicaties kunnen per functie worden verplaatst naar het beveiligingssysteem. Dit creëert voor de SMRA zelfs de mogelijkheid om alleen bepaalde functionaliteiten van diensten beschikbaar te stellen.

De kernfunctionaliteiten van het beveiligingssysteem bestaan uit: authentificatie, autorisatie, het dynamisch genereren van menu’s en het geven van blokkades mogelijkheden.
De authentificatie zorgt ervoor dat de identiteit van een gebruiker van het systeem achterhaald kan worden.
Op basis hiervan kan er autorisatie plaatsvinden. De autorisatie zorgt ervoor dat gebruikers van het systeem alleen de functionaliteiten kunnen gebruiken waartoe zij de bevoegdheid hebben verkregen.

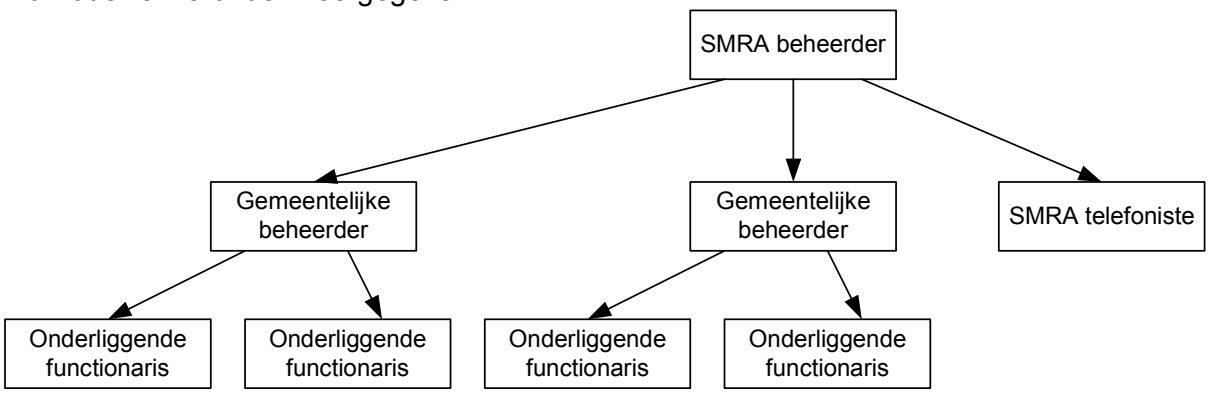
Omdat de te gebruiken functies gebruikerafhankelijk zijn worden de menu’s dynamisch opgebouwd. Zo krijgt elke gebruiker een overzicht van de functies waarvoor hij is geautoriseerd.
Als laatste spelen blokkades een belangrijke rol in het systeem. Het systeem dient in staat te zijn een functie, gebruiker, een gehele kerkelijke gemeente of zelfs alle gebruikers van het systeem te kunnen blokkeren. Dit in verband met onderhoud en contracten of afspraken tussen gebruikers.

5.2.1 Gebruikersbeheer

Het systeem dient ook de mogelijkheid te bieden voor het beheren/onderhouden van de gebruikers van het systeem. Voor het systeem zijn er een aantal verschillende gebruikerstypen herkenbaar:

- SMRA telefoniste
- SMRA beheerder
- Gemeentelijke beheerder
- Functionarissen

Binnen het gebruikersbeheer wordt er gebruik gemaakt van een gelaagd beveiligingsmodel. Dit model is hieronder weergegeven.



De SMRA beheerder beheert de gemeentelijke beheerders en SMRA telefonisten. De daaronder liggende gemeentelijke beheerders beheren de onderliggende functionarissen uit hun kerkelijke gemeente. Op deze manier kan een deel van het beheer worden uitbesteed aan gemeentelijke beheerder.

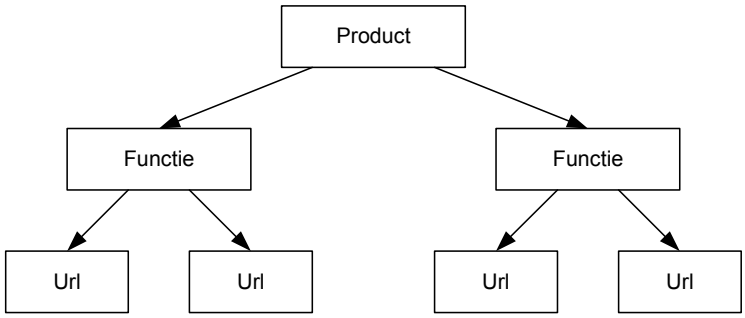
Onder gebruikersbeheer valt in dit geval:

- Toevoegen gebruiker
- Wijzigen gebruiker
- Verwijderen gebruiker
- Wijzigen gegevens gebruiker

5.2.2 Functiebeheer

Voor het beheren van functies wordt ook het gelaagde beveiligingsmodel aangehouden zoals die wordt gebruikt bij gebruikersbeheer. De door de SMRA ontwikkelde functies kunnen beschikbaar worden gesteld aan gemeentelijke beheerders. Deze beheerders kunnen gebruik maken van deze functionaliteiten maar deze ook doorgeven aan onderliggende functionarissen. Een functionaris mag deze functies dan alleen uitvoeren.

Omdat er verwacht wordt dat er vele functies zullen worden aangeboden zijn functies onder te verdelen in producten. Hieronder is een schematisch overzicht gegeven.



In het schema is zichtbaar dat ook een functie is onderverdeeld in urls. Dit geeft de mogelijkheid om zeer webfuncties te ontwikkelen. Het systeem toont een productenmenu aan de gebruiker en als de gebruiker een product kiest zullen de onderliggende functies worden getoond. Bij het kiezen van een functie zal de gebruiker de mogelijkheid krijgen om de functie uit te voeren.

Onder functiebeheer valt in dit geval:
- Toevoegen functie/product/webpagina
- Wijzigen functie/product/webpagina
- Verwijderen gebruiker functie/product/webpagina

5.2.3 Blokkades

Er zijn verschillende type blokkaden die op verschillende niveau's kunnen worden geplaatst. Zo zijn er blokkades van het type: product, functie en gebruiker. Het type geeft aan wat er geblokkeerd is. Maar de blokkades onderscheiden ook niveau's. Deze niveau's zijn: gebruiker, klant en systeem. Zo kan een blokkade van het type product worden geplaatst voor een enkele gebruiker, een geheel kerkelijke gemeente of voor alle gebruikers. Dit geld ook voor de blokkadetypen functie en gebruiker. Ook voor de blokkades is een apart model ontwikkeld. Deze is hieronder weergegeven.

Type →

Niveau ↓

	Product	Functie	Gebruiker
Gebruiker	X	X	X
Kerkelijke gemeente	X	X	X
Systeem	X	X	X

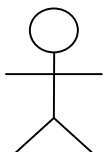
Voor de blokkades geldt dat deze alleen geplaatst kunnen worden door een SMRA beheerder of een gemeentelijke beheerder voor hun direct onderliggende gebruikers.

5.3 Actoren

De globale actoren binnen dit bedrijf zijn:
- SMRA beheerder;
- SMRA telefoniste;
- Gemeentelijke beheerder;
- Functionaris.

In de volgende onderdelen zijn de actoren volgens UML weergegeven met de bijbehorende functionaliteiten.

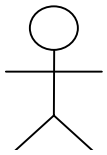
5.3.1 SMRA telefoniste



SMRA telefoniste

<<actor>> SMRA telefoniste
LogIn() VraagGebruikerGegevensOp()

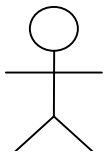
5.3.2 Functionaris



Functionaris

<<actor>> Functionaris
LogIn() EigenNawWijzigen()

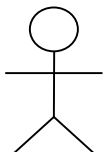
5.3.3 SMRA beheerder



SMRA beheerder

<<actor>> SMRA Beheerder
LogIn() GemeentelijkeBeheerderAanmaken() GemeentelijkeBeheerderPrinten() GemeentelijkeBeheerderVerwijderen() GemeentelijkeBeheerderWijzigen() GemeentelijkeBeheerderOpvragen() AanmakenFunctie() WijzigenFunctie() VerwijderenFunctie() FunctiesGemeentelijkeBeheerderWijzigen() GebruikersBlokkeren() GebruikersDeblokkeren() FunctieBlokkeren() FunctieDeBlokkeren() ProductBlokkeren() ProductDeblokkeren() AanmakenProduct() WijzigenProduct() VerwijderenProduct() FunctieToewijzenProduct() FunctieVerwijderenProduct()

5.3.4 Gemeentelijke beheerder



Gemeentelijke beheerder

<<actor>> Gemeentelijke beheerder
LogIn() EigenNawWijzigen() OnderliggendeFunctionarisAanmaken() OnderliggendeFunctionarisPrinten() OnderliggendeFunctionarisVerwijderen() OnderliggendeFunctionarisWijzigen() OnderliggendeFunctionarisOpvragen() OnderliggendeFunctionarisBlokkeren() OnderliggendeFunctionarisDeblokkeren() FunctiesWijzigenOnderliggendeFunctionaris()

5.4 Use-cases

Verklarende Woordenlijst:

SMRA beheerder:	De beheerder op het hoogste niveau. Deze beheerder beheerd de gemeentelijke beheerders.
SMRA telefoniste:	De SMRA telefoniste is werkzaam binnen de SMRA en gebruikt het systeem om gebruikers te valideren.
Gemeentelijke beheerder:	Een gemeentelijke beheerder beheerd de functionarissen uit de gemeente waarin hij werkzaam is.
Functionaris:	Een functionaris is een gebruiker van het systeem op het laagste niveau. Functionarissen zijn gebruikers die onder een gemeentelijke beheerder vallen. Een functionaris kan de functies gebruiken die door zijn gemeentelijke beheerder beschikbaar zijn gesteld.
Gebruiker:	SMRA beheerder, SMRA telefoniste, Gemeentelijke beheerder of Functionaris.
PinCode:	Een pincode is een getal van vier cijfers en wordt gebruikt in combinatie met de gebruikersnaam om een gebruiker te valideren.
Gebruikersnaam:	Een gebruiker krijgt een unieke gebruikersnaam toegewezen waarmee hij zich identificeert op het systeem.
Wachtwoord:	Het wachtwoord in combinatie met de gebruikersnaam geeft de gebruiker de mogelijkheid zich aan te melden op het systeem en de voor hem beschikbare toepassingen uit te voeren.
Functie:	Een functie wordt beschikbaar gesteld voor gebruikers. Een functie bestaat uit een of meerdere webpagina's.
Product:	Een product is een groepering van functies. Onder een product vallen een aantal functies.

5.4.1 SMRA telefoniste

Naam	LogIn()
Aannamen	Systeem is zojuist gestart.
Beschrijving	1. De gebruiker maakt zijn/haar gebruikernaam bekent met het daarbij behorende wachtwoord. 2. Het systeem controleert of de gebruiker bekend is bij het systeem. Zo niet, dan treed er een uitzondering op.
Uitzonderingen	Gebruikersnaam of wachtwoord is niet juist. Er wordt een melding gegeven dat de gebruikersnaam of het wachtwoord niet klopt. Verdere toegang wordt uitgesloten.
Resultaat	De SMRA telefoniste is ingelogd en is in staat gebruikersgegevens op te vragen.

Naam	VraagGebruikerGegevensOp()
Aannamen	De SMRA telefoniste is zojuist ingelogd en het scherm voor het opvragen van gebruikersgegevens wordt getoond.
Beschrijving	1. De SMRA telefoniste voert de gemeentelijke beheerder of functionaris zijn gebruikersnaam en pincode in. 2. Het systeem controleert of de gebruiker bekend is bij het systeem. Zo niet dan treed er een uitzondering op. 3. De gebruikersgegevens van de persoon die hoort bij de ingevoerde gebruikersnaam worden getoond.
Uitzonderingen	Gebruikersnaam of pincode niet juist. Er wordt een melding gegeven dat het gebruikersnaam of pincode niet klopt.
Resultaat	De SMRA telefoniste kan de gebruiker valideren.

5.4.2 SMRA beheerder

Naam	LogIn()
Aannamen	Systeem is zojuist gestart.
Beschrijving	1. De gebruiker maakt zijn/haar gebruikernaam bekend met het daarbij behorende wachtwoord. 2. Het systeem controleert of de gebruiker bekend is bij het systeem. Zo niet, dan treedt er een uitzondering op.
Uitzonderingen	Gebruikersnaam of wachtwoord is niet juist. Er wordt een melding gegeven dat de gebruikersnaam of het wachtwoord niet klopt. Verdere toegang wordt uitgesloten.
Resultaat	De gebruiker krijgt de mogelijkheid de uit te voeren functies te kiezen.

Naam	GemeentelijkeBeheerderAanmaken()
Aannamen	Het scherm “Gemeentelijke beheerder aanmaken” wordt getoond.
Beschrijving	1. De SMRA beheerder voert een nieuwe gebruikersnaam, de bijbehorende NAW gegevens en de bijbehorende gemeente in.. 2. Het systeem genereert een bijbehorend wachtwoord. 3. De SMRA beheerder kan nu opslaan kiezen. Indien hij dit niet kiest treedt er een uitzondering op.[1] 4. De wijzigingen worden opgeslagen en er wordt getoond dat de wijzigingen zijn doorgevoerd. Als de gebruikersnaam al bestaat treedt er een uitzondering op.[2] 5. Use-case gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Word na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen. 2. Er wordt gecontroleerd of de gebruikersnaam niet reeds bestaat. Indien de gebruiker al bestaat worden de gegevens niet opgeslagen en wordt er een melding gegeven dat de gebruikersnaam al in gebruik is. Het scherm keert terug naar de ingevoerde gebruikersgegevens.
Resultaat	Er is een gemeentelijke beheerder aangemaakt.

Naam	Gebruikeroverzicht (sub-case)
Aannamen	Deze functie wordt opgeroepen door een andere functie.
Beschrijving	1. Het systeem zoekt de gebruikergegevens op met behulp van de meegekregen gebruikersnaam. 2. Het systeem toont de gegevens van de gebruiker in een overzicht.
Uitzonderingen	Geen.
Resultaat	Het systeem toont de gegevens van de gebruiker in een overzicht aan de gebruiker.

Naam	GemeentelijkeBeheerderPrinten()
Aannamen	Het scherm “Gemeentelijke beheerder opvragen” wordt getoond
Beschrijving	1. De SMRA beheerder voert de gebruikersnaam in van de op te vragen gemeentelijke beheerder. 2. Het systeem controleert de gebruikersnaam en het wachtwoord. Indien deze niet correct zijn, treed er een uitzondering op.[1] 3. Het systeem controleert of de gebruiker van het type gemeentelijke beheerder is. Zo niet, dan treedt er een uitzondering op.[2] 4. Use-case gebruikeroverzicht wordt uitgevoerd waar de SMRA beheerder de mogelijkheid heeft om de gemeentelijke beheerder te printen.
Uitzonderingen	1. Gebruikersnaam en wachtwoord incorrect, er wordt hiervan een melding gegeven. 2. De gebruiker is geen gemeentelijke beheerder, er wordt hiervan melding gegeven.
Resultaat	Een geprint overzicht met gebruikersgegevens die kan worden verstuurd naar een (nieuwe) gemeentelijke beheerder.

Naam	GemeentelijkeBeheerderVerwijderen()
Aannamen	Het scherm “Gemeentelijke beheerder verwijderen” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de gebruikersnaam van de te verwijderen gemeentelijke beheerder in. 2. De SMRA beheerder kiest nu verwijderen. Zo niet dan treedt er een uitzondering op.[1] 3. Het systeem controleert of de gebruikersnaam bestaat en of de gebruiker een gemeentelijke beheerder is. Zo niet dan treedt er een uitzondering op.[2]
Uitzonderingen	1. Wordt na invoeren van de gegevens, verwijderen niet gekozen en wordt de pagina verlaten, dan worden er geen gegevens gewijzigd. 2. Als de gebruikersnaam niet bestaat of de gebruiker geen gemeentelijke beheerder is, zal dit worden weergegeven en wordt er niets verwijderd.
Resultaat	De gemeentelijke beheerder is verwijderd.

Naam	GemeentelijkeBeheerderOpvragen()
Aannamen	Het scherm “gemeentelijke beheerder opvragen” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de gebruikersnaam in van de op te vragen gemeentelijke beheerder. 2. Het systeem controleert of de gebruikersnaam bestaat en of de gebruiker een gemeentelijke beheerder is. Zo niet dan treedt er een uitzondering op.[1] 3. Usecase gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat of de gebruiker geen gemeentelijke beheerder is, zal dit worden weergegeven.
Resultaat	Het systeem toont de gegevens van de gemeentelijke beheerder in een overzicht aan de SMRA beheerder.

Naam	GemeentelijkeBeheerderWijzigen()
Aannamen	Het scherm “Gemeentelijke beheerder wijzigen” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de gebruikersnaam in van de te wijzigen gemeentelijke beheerder. 2. Het systeem controleert of de gebruikersnaam bestaat en of de gebruiker een gemeentelijke beheerder is. Zo niet dan treedt er een uitzondering op.[1] 3. Het systeem haalt de gegevens van de gebruiker op en toont deze in het scherm. 4. De SMRA beheerder voert de wijzigingen in. 5. De SMRA beheerder kiest opslaan. Zo niet, dan treed er een uitzondering op.[2] 6. De wijzigingen worden opgeslagen en er wordt getoond dat de wijzigingen zijn doorgevoerd. 7. Use-case gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat of de gebruiker geen gemeentelijke beheerder is, zal dit worden weergegeven. 2. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	De gewijzigde gegevens zijn opgeslagen in het systeem.

Naam	VerwijderenFunctie()
Aannamen	Het scherm “Functie verwijderen” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de naam van de functie en kiest verwijderen. 2. Het systeem zoekt controleert of de opgegeven functienaam bestaat. Indien de functienaam niet bestaat treed er een uitzondering op.[1] 3. De functie wordt uit het systeem verwijderd.
Uitzonderingen	1. De Functienaam bestaat niet. Hiervan wordt melding gegeven.
Resultaat	De functie is verwijderd.

Naam	AanmakenFunctie()
Aannamen	Het scherm “Functie aanmaken” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de naam van de functie, kiest het bijbehorende product en kiest de bijbehorende webpagina waarmee de functie start. 2. De SMRA beheerder kiest toevoegen. 3. Het systeem controleert de Functienaam. Indien de functienaam al bestaat, treedt er een uitzondering op.[1] 4. Use-case Functieoverzicht wordt uitgevoerd.
Uitzonderingen	1. Als de functienaam al bestaat wordt dit weergegeven en wordt de functie niet toegevoegd.
Resultaat	Er is een nieuwe functie aangemaakt.

Naam	Functieoverzicht (sub-case)
Aannamen	Deze functie wordt opgeroepen door een andere functie.
Beschrijving	1. Het systeem zoekt de functiegegevens op met behulp van de meegekregen functienaam. 2. Het systeem toont de gegevens van de functie in een overzicht.
Uitzonderingen	Geen.
Resultaat	Het systeem toont de gegevens van de functie in een overzicht aan de gebruiker.

Naam	WijzigenFunctie()
Aannamen	Het scherm “Functie wijzigen” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de naam van de functie. 2. Het systeem zoekt de gegevens van de functie met de opgegeven functienaam op. Indien de functienaam niet bestaat treed er een uitzondering op.[1] 3. De SMRA beheerder voert de wijzigingen door en kiest wijzigen. 4. Het systeem slaat de wijzigingen op. 5. Use-case Functieoverzicht wordt uitgevoerd.
Uitzonderingen	1. De Functienaam bestaat niet. Hiervan wordt melding gegeven.
Resultaat	De functie is gewijzigd.

Naam	FunctiesGemeentelijkeBeheerderWijzigen()
Aannamen	Het scherm “Functies gemeentelijke beheerder” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de gebruikersnaam in van de op te vragen gemeentelijke beheerder. 2. Het systeem controleert of de gebruikersnaam bestaat en of de gebruiker een gemeentelijke beheerder is. Zo niet dan treedt er een uitzondering op.[1] 3. Het systeem toont een lijst met de reeds toegewezen en beschikbare functies voor de gemeentelijke beheerder. 4. De SMRA beheerder wijzigt de toegestane functies en kiest opslaan. Zo niet, dan treed er een uitzondering op.[2] 5. Er wordt een overzicht gegeven met de toegestane en niet toegestane functies van de gemeentelijke beheerder.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat of de gebruiker geen gemeentelijke beheerder is, zal dit worden weergegeven. 2. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	De toegestane functies van een gemeentelijke beheerder zijn gewijzigd.

Naam	FunctieBlokkeren() of FunctieDeBlokkeren()
Aannamen	Het scherm “functie blokkeren” wordt getoond.
Beschrijving	1. Het systeem genereert een lijst met beschikbare functies. 2. De SMRA beheerder kiest de functie. 3. Het systeem toont de mogelijke niveau’s waarop de functie geblokkeerd kan worden. (Gebruikers-, kant- of systeemniveau) 4. De SMRA beheerder kiest, indien nodig, de gebruikersnaam of klantnaam. 5. De SMRA beheerder kiest blokkeren of deblokkeren. Zo niet, dan treed er een uitzondering op.[1]
Uitzonderingen	1. Wordt na het kiezen van de functie, blokkeren of beschikbaar stellen niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	Een functie is geblokkeerd of gedeblokkeerd voor een specifieke gebruiker, gebruikers van een specifieke bepaalde klant of voor alle gebruikers.

Naam	ProductBlokkeren() of ProductDeBlokkeren()
Aannamen	Het scherm “product blokkeren” wordt getoond.
Beschrijving	1. Het systeem genereert een lijst met beschikbare producten. 2. De SMRA beheerder kiest de product. 3. Het systeem toont de mogelijke niveau’s waarop het product geblokkeerd kan worden. (Gebruikers-, kant- of systeemniveau) 4. De SMRA beheerder kiest, indien nodig, de gebruikersnaam of klantnaam. 5. De SMRA beheerder kiest blokkeren of deblokkeren. Zo niet, dan treed er een uitzondering op.[1]
Uitzonderingen	1. Wordt na het kiezen van het product, blokkeren of beschikbaar stellen niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	Een product is geblokkeerd of gedeblokkeerd voor een specifieke gebruiker, gebruikers van een specifieke bepaalde klant of voor alle gebruikers.

Naam	GebruikersBlokkeren() of GebruikersDeblokkeren()
Aannamen	Het scherm “Gemeente blokkeren/deblokkeren” wordt getoond.
Beschrijving	1. De SMRA beheerder kiest het gebruikersniveau en kiest blokkeren of deblokkeren. Het niveau kan hier zijn: gebruiker, klant of systeem. 2. Afhankelijk van het gekozen niveau kiest de SMRA beheerder de te blokkeren of deblokkeren gebruikersnaam of klantnaam. 3. De SMRA beheerder kiest blokkeren of deblokkeren. Zo niet, dan treed er een uitzondering op.[1]
Uitzonderingen	1. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	Er is eens specifieke gebruiker of klant geblokkeerd of alle gebruikers zijn geblokkeerd.

Naam	AanmakenProduct()
Aannamen	Het scherm “Nieuw product” wordt getoond.
Beschrijving	1. De SMRA beheerder voert de naam in van het nieuwe product. 2. De SMRA beheerder kiest toevoegen. Indien de productnaam al bestaat dan treed er een uitzondering op.[1]
Uitzonderingen	1. Er wordt een melding gegeven dat de productnaam al bestaat.
Resultaat	Er is een nieuw product aangemaakt.

Naam	WijzigenProduct()
Aannamen	Het scherm “Wijzigen Product” wordt getoond.
Beschrijving	1. De SMRA beheerder kiest de naam van het te wijzigen product. 2. De SMRA beheerder wijzigt de naam van het product en kiest opslaan. Zo niet, dan treed er een uitzondering op.[1]
Uitzonderingen	1. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zal de gewijzigde naam niet worden opgeslagen.
Resultaat	De naam van het product is gewijzigd.

Naam	VerwijderenProduct()
Aannamen	Het scherm “Verwijderen Product” wordt getoond.
Beschrijving	1. De SMRA beheerder kiest de naam van het te verwijderen product. 2. De SMRA beheerder kiest verwijderen.
Resultaat	Het product is verwijderd.

Naam	FunctieToewijzenProduct()
Aannamen	Het scherm “Functie toewijzen aan Product” wordt getoond.
Beschrijving	1. De SMRA beheerder kiest de functie. 2. De SMRA beheerder kiest het product en kiest opslaan.
Resultaat	Er is een functie toegevoegd aan het product.

Naam	FunctieVerwijderenProduct()
Aannamen	Het scherm “Functie verwijderen uit Product” wordt getoond.
Beschrijving	1. De SMRA beheerder kiest het product. 2. De SMRA beheerder kiest de functie en kiest verwijderen.
Resultaat	Er is een functie toegevoegd aan het product.

5.4.3 Functionaris

Naam	LogIn()
Aannamen	Systeem is zojuist gestart.
Beschrijving	1. De gebruiker maakt zijn/haar gebruikernaam bekend met het daarbij behorende wachtwoord. 2. Het systeem controleert of de gebruiker bekend is bij het systeem. Zo niet, dan treedt er een uitzondering op.
Uitzonderingen	Gebruikersnaam of wachtwoord is niet juist. Er wordt een melding gegeven dat de gebruikersnaam of het wachtwoord niet klopt. Verdere toegang wordt uitgesloten.
Resultaat	De gebruiker krijgt de mogelijkheid de uit te voeren functies te kiezen.

Naam	EigenNawWijzigen()
Aannamen	Het scherm NAW gegevens wordt getoond.
Beschrijving	1. De functionaris kan de NAW gegevens van zichzelf wijzigen. 2. De functionaris kiest opslaan. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan treedt er een uitzondering op. 3. De gewijzigde gegevens worden opgeslagen. 4. Het scherm NAW gegevens wordt opnieuw getoond met de gewijzigde gegevens.
Uitzonderingen	1. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	De NAW gegevens van de functionaris zijn opgeslagen.

5.4.4 Gemeentelijke beheerder

Naam	LogIn()
Aannamen	Systeem is zojuist gestart.
Beschrijving	1. De gebruiker maakt zijn/haar gebruikernaam bekend met het daarbij behorende wachtwoord. 2. Het systeem controleert of de gebruiker bekend is bij het systeem. Zo niet, dan treedt er een uitzondering op.
Uitzonderingen	Gebruikersnaam of wachtwoord is niet juist. Er wordt een melding gegeven dat de gebruikersnaam of het wachtwoord niet klopt. Verdere toegang wordt uitgesloten.
Resultaat	De gebruiker krijgt de mogelijkheid de uit te voeren functies te kiezen.

Naam	EigenNawWijzigen()
Aannamen	Het scherm NAW gegevens wordt getoond.
Beschrijving	1. De gemeentelijke beheerder kan de NAW gegevens van zichzelf wijzigen. 2. De gemeentelijke beheerder kiest opslaan. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan treedt er een uitzondering op. 3. De gewijzigde gegevens worden opgeslagen. 4. Het scherm NAW gegevens wordt opnieuw getoond met de gewijzigde gegevens.
Uitzonderingen	1. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	De NAW gegevens van de gemeentelijke beheerder zijn opgeslagen.

Naam	FunctionarisAanmaken()
Aannamen	Het scherm “Functionaris aanmaken” wordt getoond.
Beschrijving	1. De Gemeentelijke beheerder voert een nieuwe gebruikersnaam en de bijbehorende NAW gegevens in. 2. Het systeem genereert een bijbehorend wachtwoord. 3. De Gemeentelijke beheerder kan nu opslaan kiezen. Indien hij dit niet kiest treedt er een uitzondering op.[1] 4. De wijzigingen worden opgeslagen en er wordt getoond dat de wijzigingen zijn doorgevoerd. Als de gebruikersnaam al bestaat treedt er een uitzondering op.[2] 5. Use-case gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Word na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen. 2. Er wordt gecontroleerd of de gebruikersnaam niet reeds bestaat. Indien de gebruiker al bestaat worden de gegevens niet opgeslagen en wordt er een melding gegeven dat de gebruikersnaam al in gebruik is. Het scherm keert terug naar de ingevoerde gebruikersgegevens.
Resultaat	Er is een functionaris aangemaakt.

Naam	Gebruikeroverzicht (sub-case)
Aannamen	Deze functie wordt opgeroepen door een andere functie.
Beschrijving	1. Het systeem zoekt de gebruikergegevens op met behulp van de meegekregen gebruikersnaam. 2. Het systeem toont de gegevens van de gebruiker in een overzicht.
Uitzonderingen	Geen.
Resultaat	Het systeem toont de gegevens van de gebruiker in een overzicht aan de gebruiker.

Naam	FunctionarisPrinten()
Aannamen	Het scherm “Functionaris opvragen” wordt getoond
Beschrijving	1. De Gemeentelijke beheerder voert de gebruikersnaam in van de op te vragen functionaris. 2. Het systeem controleert de gebruikersnaam, het wachtwoord en de gemeente. Indien deze niet correct zijn, treed er een uitzondering op.[1] 3. Het systeem controleert of de gebruiker van het type functionaris is. Zo niet, dan treedt er een uitzondering op.[2] 4. Use-case gebruikeroverzicht wordt uitgevoerd waar de Gemeentelijke beheerder de mogelijkheid heeft om de functionaris te printen.
Uitzonderingen	1. Gebruikersnaam, wachtwoord of gemeente incorrect, er wordt hiervan een melding gegeven. 2. De gebruiker is geen functionaris, er wordt hiervan melding gegeven.
Resultaat	Een geprint overzicht met gebruikersgegevens die kan worden verstuurd naar een (nieuwe) functionaris.

Naam	FunctionarisVerwijderen()
Aannamen	Het scherm “Functionaris verwijderen” wordt getoond.
Beschrijving	1. De Gemeentelijke beheerder voert de gebruikersnaam van de te verwijderen functionaris in. 2. De Gemeentelijke beheerder kiest nu verwijderen. Zo niet dan treedt er een uitzondering op.[1] 3. Het systeem controleert of de gebruikersnaam bestaat, of de gebruiker een functionaris is en of de gebruiker zich in de desbetreffende gemeente bevindt. Zo niet dan treedt er een uitzondering op.[2]
Uitzonderingen	1. Wordt na invoeren van de gegevens, verwijderen niet gekozen en wordt de pagina verlaten, dan worden er geen gegevens gewijzigd. 2. Als de gebruikersnaam niet bestaat, de gebruiker geen functionaris is of dat de gebruiker zich in een andere gemeente bevindt, dan zal dit worden weergegeven en wordt er niets verwijderd.
Resultaat	De functionaris is verwijderd.

Naam	FunctionarisWijzigen()
Aannamen	Het scherm “Functionaris wijzigen” wordt getoond.
Beschrijving	1. De Gemeentelijke beheerder voert de gebruikersnaam in van de te wijzigen functionaris. 2. Het systeem controleert of de gebruikersnaam bestaat, of de gebruiker een functionaris is en of de gebruiker zich in de desbetreffende gemeente bevindt. Zo niet dan treedt er een uitzondering op.[1] 3. Het systeem haalt de gegevens van de gebruiker op en toont deze in het scherm. 4. De Gemeentelijke beheerder voert de wijzigingen in. 5. De Gemeentelijke beheerder kiest opslaan. Zo niet, dan treed er een uitzondering op.[2] 6. De wijzigingen worden opgeslagen en er wordt getoond dat de wijzigingen zijn doorgevoerd. 7. Use-case gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat, de gebruiker geen functionaris is of dat de gebruiker zich in een andere gemeente bevindt, dan zal dit worden weergegeven. 2. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	De gewijzigde gegevens zijn opgeslagen in het systeem.

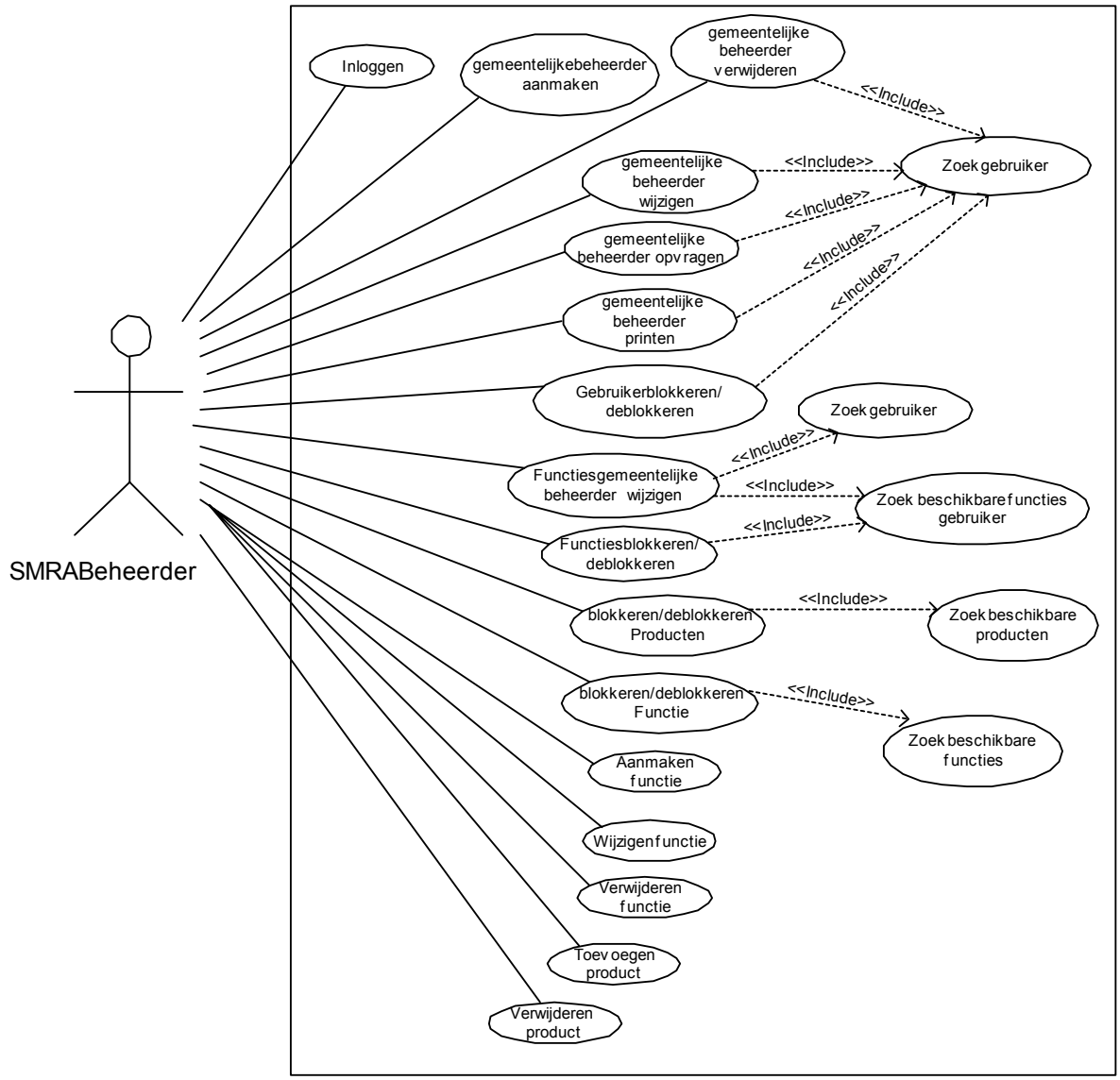
Naam	FunctionarisOpvragen()
Aannamen	Het scherm “functionaris opvragen” wordt getoond.
Beschrijving	1. De Gemeentelijke beheerder voert de gebruikersnaam in van de op te vragen functionaris. 2. Het systeem controleert of de gebruikersnaam bestaat, of de gebruiker een functionaris is en of de gebruiker zich in de desbetreffende gemeente bevindt. Zo niet dan treedt er een uitzondering op.[1] 3. Usecase gebruikeroverzicht wordt uitgevoerd.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat, de gebruiker geen functionaris is of dat de gebruiker zich in een andere gemeente bevindt, dan zal dit worden weergegeven.
Resultaat	Het systeem toont de gegevens van de functionaris in een overzicht aan de Gemeentelijke beheerder.

Naam	FunctionarisBlokkeren() of FunctionarisDeblokkeren()
Aannamen	Het scherm functionaris blokkeren/deblokkeren wordt getoond.
Beschrijving	1. De Gemeentelijke beheerder voert de gebruikersnaam in van de te blokkeren functionaris. 2. De Gemeentelijke beheerder kiest blokkeren of deblokkeren. 3. Het systeem controleert of de gebruikersnaam bestaat, of de gebruiker een functionaris is en of de gebruiker zich in de desbetreffende gemeente bevindt. Zo niet dan treedt er een uitzondering op.[1] 4. Het systeem blokkeert de functionaris. Er wordt een melding gegeven dat de functionaris is geblokkeerd of gedeblokkeerd.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat, de gebruiker geen functionaris is of dat de gebruiker zich in een andere gemeente bevindt, dan zal dit worden weergegeven.
Resultaat	Er is een functionaris geblokkeerd of gedeblokkeerd.

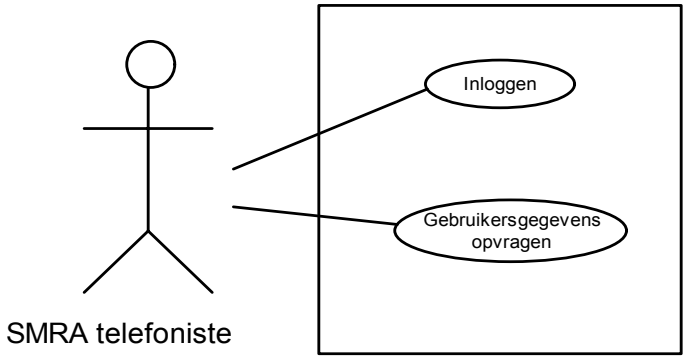
Naam	FunctiesFunctionarisWijzigen()
Aannamen	Het scherm “Functies functionaris” wordt getoond.
Beschrijving	1. De gemeentelijke beheerder voert de gebruikersnaam in van de op te vragen functionaris. 2. Het systeem controleert of de gebruikersnaam bestaat, of de gebruiker een functionaris is en of de gebruiker zich in de desbetreffende gemeente bevindt. Zo niet dan treedt er een uitzondering op.[1] 3. Het systeem toont een lijst met de reeds toegewezen en beschikbare functies van de functionaris. 4. Gemeentelijke beheerder wijzigt de toegestane functies en kiest opslaan. Zo niet, dan treed er een uitzondering op.[2] 5. Er wordt een overzicht gegeven met de toegestane en niet toegestane functies van de gemeentelijke beheerder.
Uitzonderingen	1. Als de gebruikersnaam niet bestaat, de gebruiker geen functionaris is of dat de gebruiker zich in een andere gemeente bevindt, dan zal dit worden weergegeven. 2. Wordt na het wijzigen van de gegevens, opslaan niet gekozen en wordt de pagina verlaten, dan zullen de gewijzigde gegevens niet worden opgeslagen.
Resultaat	De toegestane functies van een functionaris zijn gewijzigd.

5.5 Use-case diagram

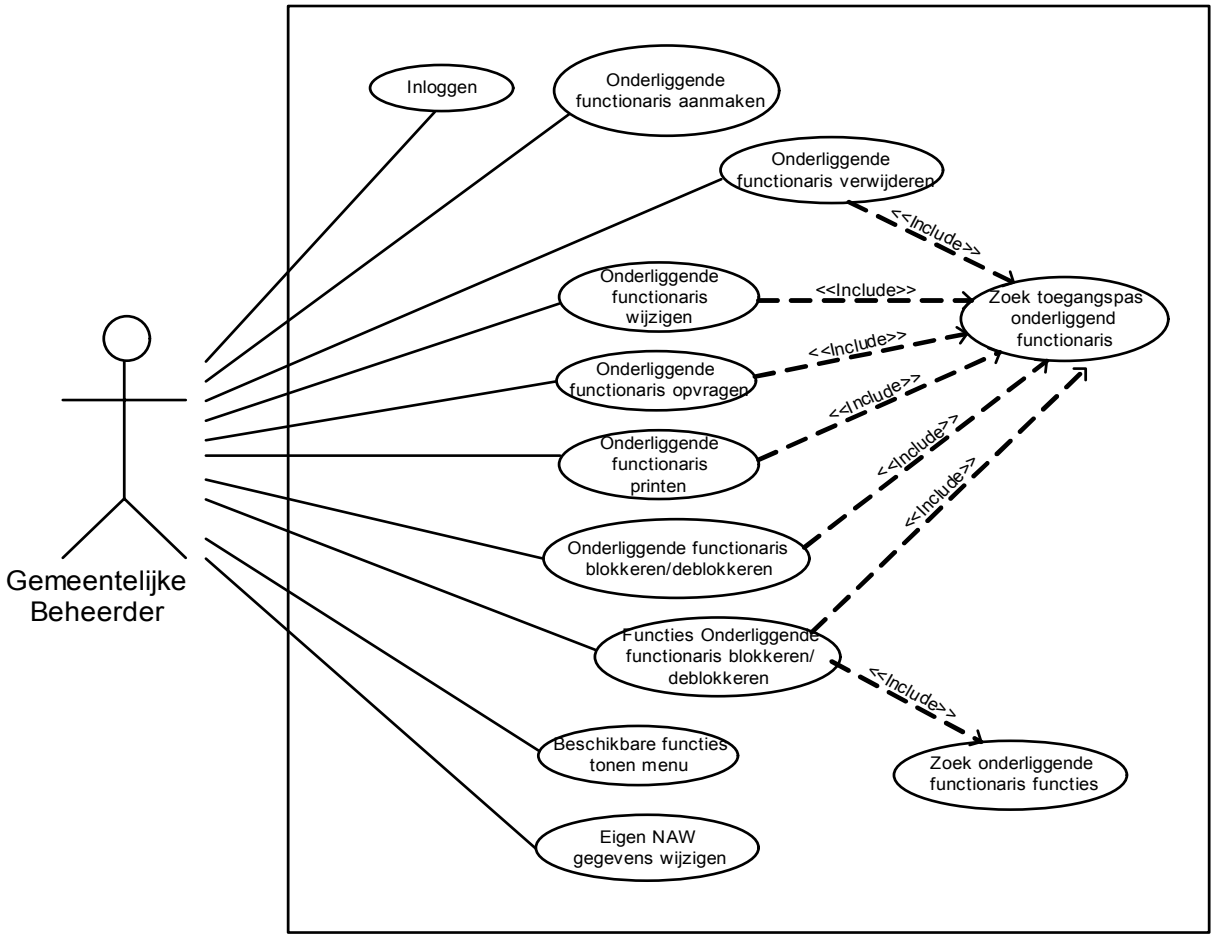
5.5.1 SMRA Beheerder



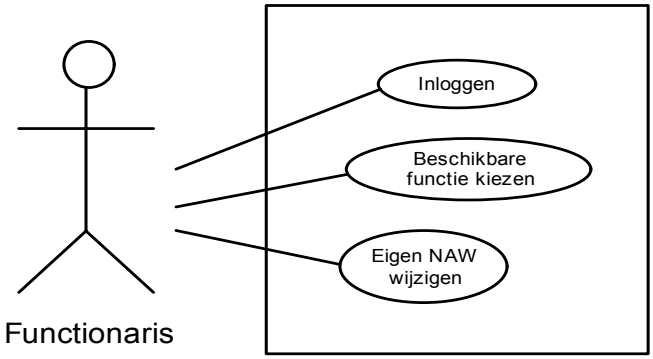
5.5.2 SMRA telefoniste



5.5.3 Gemeentelijke beheerder



5.5.4 Functionaris



5.6 Aggregatie functies

In dit onderdeel wordt er bekeken welke functionaliteiten van het systeem bij meerdere gebruikers voorkomen. Van deze functionaliteiten wordt gecontroleerd of het hierbij om dezelfde functies gaat of om functies die op elkaar lijken.

Functie	**SMRA telefoniste**	**SMRA Beheerder**	**Gemeentelijke beheerder**	**Functionaris**
LogIn()	**X**	**X**	**X**	**X**
VraagGebruikerGegevensOp()	X			
EigenNawWijzigen()			**X**	**X**
GemeentelijkeBeheerderAanmaken()		X		
GemeentelijkeBeheerderPrinten()		X		
GemeentelijkeBeheerderVerwijderen()		X		
GemeentelijkeBeheerderWijzigen()		X		
GemeentelijkeBeheerderOpvragen()		X		
GemeentelijkeBeheerderBlokkeren()		X		
GemeentelijkeBeheerderDeblokkeren()		X		
AanmakenFunctie()		X		
WijzigenFunctie()		X		
VerwijderenFunctie()		X		
AanmakenProduct()		X		
WijzigenProduct()		X		
VerwijderenProduct()		X		
FunctiesGemeentelijkeBeheerderWijzigen()		X		
FunctieBlokkeren()		**X**	**X**	
FunctieDeBlokkeren()		**X**	**X**	
GebruikersBlokkeren()		**X**	**X**	
GebruikersDeblokkeren()		**X**	**X**	
ProductBlokkeren()		**X**	**X**	
ProductDeBlokkeren()		**X**	**X**	
FunctieToewijzenProduct()		X		
FunctieVerwijderenProduct()		X		
OnderliggendeFunctionarisAanmaken()			X	
OnderliggendeFunctionarisPrinten()			X	
OnderliggendeFunctionarisVerwijderen()			X	
OnderliggendeFunctionarisWijzigen()			X	
OnderliggendeFunctionarisOpvragen()			X	
OnderliggendeFunctionarisBlokkeren()			X	
OnderliggendeFunctionarisDeblokkeren()			X	
FunctiesWijzigenOnderliggendeFunctionaris()			X	

In het overzicht valt gelijk op dat de functie LogIn() bij alle gebruikerstypen voorkomt en dat de functie EigenNawGegevensWijzigen() voorkomt bij de rol gemeentelijke beheerder en de functionaris.

Toch zijn er nog andere functies die waarschijnlijk in praktijk dezelfde functies zullen zijn maar met beperkingen. Deze functies zijn hieronder weergegeven.

GemeentelijkeBeheerderAanmaken()	OnderliggendeFunctionarisAanmaken()
GemeentelijkeBeheerderPrinten()	OnderliggendeFunctionarisPrinten()
GemeentelijkeBeheerderVerwijderen()	OnderliggendeFunctionarisVerwijderen()
GemeentelijkeBeheerderWijzigen()	OnderliggendeFunctionarisWijzigen()
GemeentelijkeBeheerderOpvragen()	OnderliggendeFunctionarisOpvragen()
FunctiesGemeentelijkeBeheerderWijzigen()	FunctiesWijzigenOnderliggendeFunctionaris()

Tijdens de pilotontwikkeling zal blijken of deze functies daadwerkelijk in het systeem worden verwerkt als losse functies die met verschillende argumenten worden aangeroepen. Indien

de functies gecompliceerd worden door het aantal argumenten en de structuur van de functies ingewikkeld blijkt te zijn kan ervoor gekozen worden om de functies apart op te nemen.

5.7 Klasse diagram

5.7.1 Klasse selectie

Kandidaat-klasse	Beslissing		Kandidaat-klasse	Beslissing
Gebruiker	Is klasse		Melding	implementatie
SMRA telefoniste	Redundant met gebruiker		Gebruikersgegevens	Te vaag (attributen)
Systeem	Implementatie		Gemeentelijke beheerder	Redundant met gebruiker
Gebruikersnaam	Attribuut		Functionaris	Redundant met gebruiker
Wachtwoord	Attribuut		Functies	Is klasse
Uitzondering	Implementatie		Beveiligde webpagina	Associatie
Gebruikersmenu	Implementatie		Applicatie	Is Klasse
Producten	Is klasse		SMRA beheerder	Redundant met gebruiker
NAW-gegevens	Te vaag (attributen)		Gebruikerstype	Is Klasse
Gemeente	Is klasse		Leden	Te vaag
Beveiligde webpagina's	Is klasse		Print gebruikersgegevens	associatie
Menu	Implementatie		Code	Redundant met wachtwoord

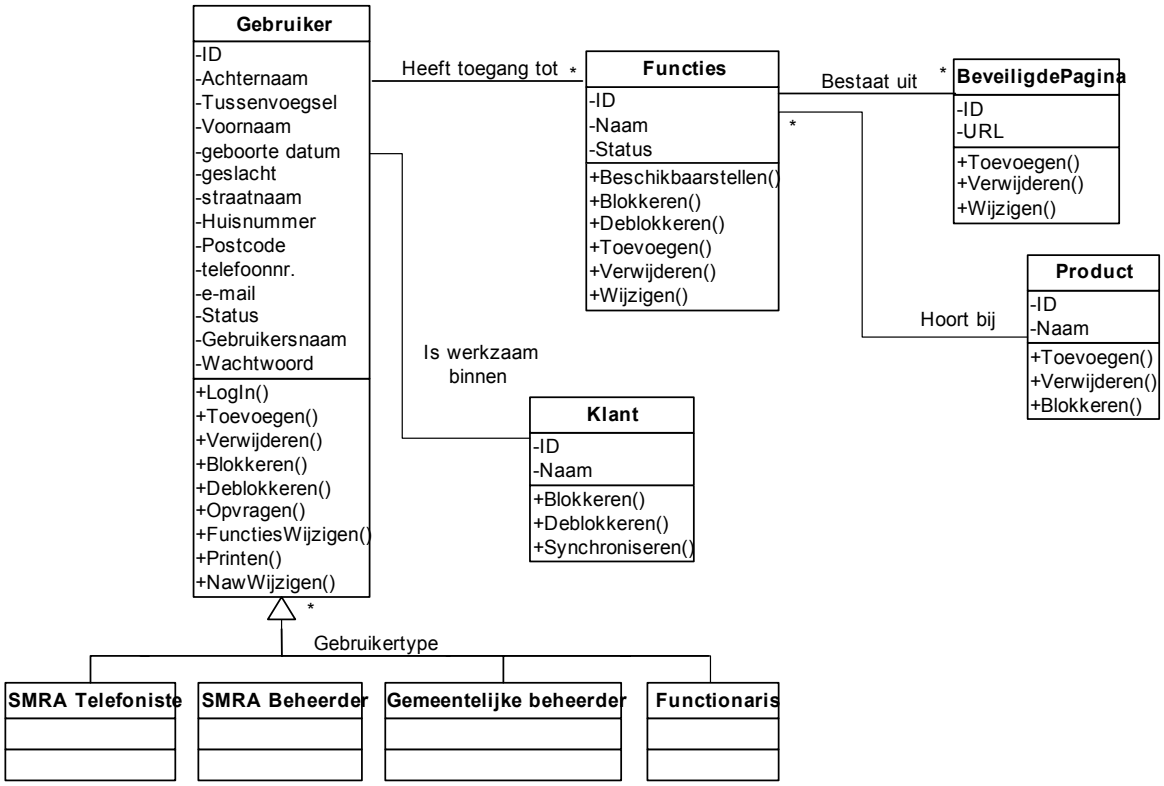
5.7.2 Modeldictionary

klasse	Omschrijving
Gebruiker	Representatie binnen het systeem van een reëel persoon.
Functies	Een functionaliteit die aan een gebruiker kan worden toegewezen.
Gemeente	Een lokale kerk waarvan leden (functionarissen) toegang krijgen tot SMRA fucnties.
Beveiligde webpagina's (URL)	Een Functie bestaat uit 1 of meerdere beveiligde webpagina's.
Gebruikerstype	Het gaat om een gelaagd beveiligingsmodel. Ofwel een gebruiker kan een SMRA beheerder, SMRA telefoniste, Gemeentelijke beheerder of een Functionaris zijn.
Product	Een product is een groepering van een aantal functies.

5.7.3 Identificatie attributen en operaties

klasse	Attributen	Operaties
Gebruiker	ID Voornaam Tussenvoegsel Achternaam Geboorte datum Geslacht Straatnaam Huisnummer Postcode Gebruikerstype Gebruikersrol Gemeente_ID	LogIn() Toevoegen() Verwijderen() Blokkeren() Deblokkeren() Opvragen() FunctiesWijzigen() Printen() NawWijzigen()
Functies	ID Naam Applicatie_ID URL_ID Status	Blokkeren() DeBlokkeren() Aanmaken() Verwijderen() Wijzigen() Beschikbaarstellen()
Gemeente	ID Naam Status	Blokkeren() DeBlokkeren() Synchroniseren()
Beveiligde webpagina's (URL)	ID URL	Toevoegen() Verwijderen() Wijzigen()
Product	ID Naam	Toevoegen() Verwijderen()

5.7.4 Klassendiagram



6 Technische structuur

6.1 Inleiding

In dit deel van het document wordt de technische structuur van het bedrijf bekeken en wordt er gecontroleerd of het te realiseren systeem in de huidige technische structuur kan worden toegepast. Dit geldt zowel voor de hardware als voor de software.

6.2 Technische architectuur

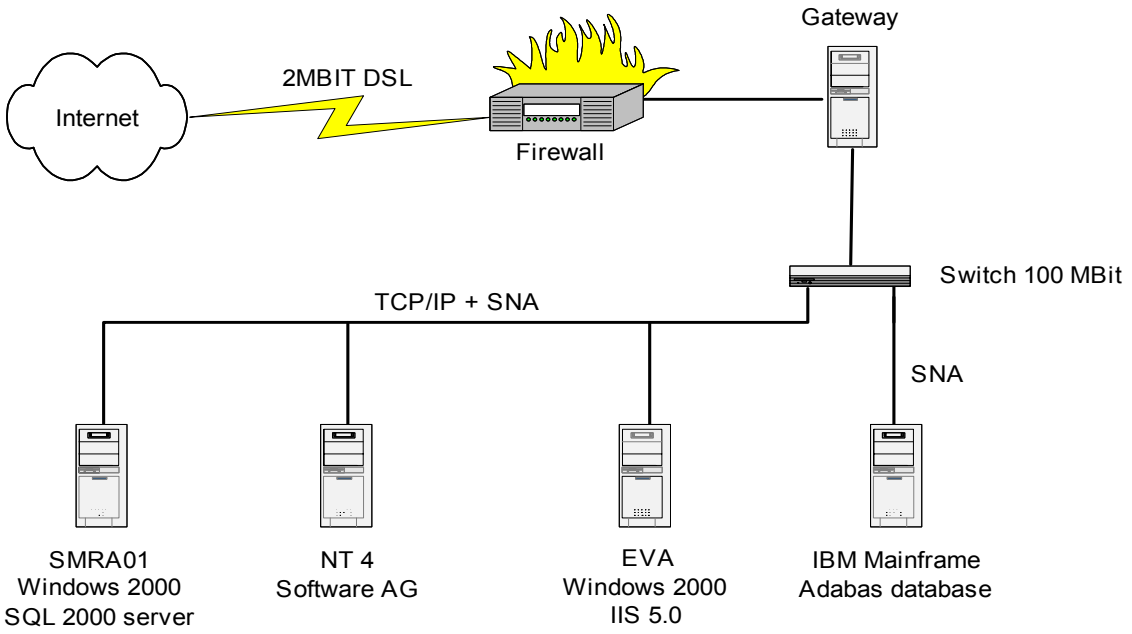
De SMRA host zijn eigen website. Het is de bedoeling dat de beveiligingsapplicatie op een webserver binnen de SMRA wordt geconfigureerd. In dit onderdeel zijn alleen de onderdelen die relevant zijn voor het systeem opgenomen.

De SMRA Heeft een DSL Internet verbinding met een verbindingssnelheid van 2 Mbit up en down. Deze verbinding wordt geregeld via de Firewall. De Firewall filtert inkomend verkeer op virussen en blokkeert de toegang op bepaalde poorten. (Deze zijn in overleg met de systeembeheerder in te stellen). De gateway achter de Firewall dient als proxy server zodat het personeel binnen de SMRA gebruik kan maken van Internet en e-mail. Ook zorgt de gateway ervoor dat de webservice (www.smra.nl) op de server EVA kan worden opgevraagd vanaf het Internet.

Binnen het netwerk van de SMRA wordt er gewerkt met verschillende protocollen. De voor dit project relevante protocollen zijn TCP/IP en SNA. Hiernaast wordt ook nog IPX-SPX gebruikt voor de Novell FileSharing server.

Het beveiligingssysteem zal gebaseerd zijn op ASP.NET en maakt gebruikt van een IIS server. De website van de SMRA wordt reeds gehost op een IIS service en het beveiligingssysteem zou hiernaast kunnen functioneren.

Schematisch overzicht huidige technische structuur:



De computers die betrekking hebben tot de invoering van het beveiligingssysteem werken alle met TCP/IP behalve de IBM Mainframe server. De IBM Mainframeserver werkt met SNA. Op de IBM Mainframeserver staat de Adabas database en deze gegevens zijn nodig binnen het beveiligingssysteem.
Op de NT4 server is een service van Software AG beschikbaar die een TCP/IP request vertaalt naar SNA en de request doorstuurt naar de IBM server.

6.3 Technische veranderingen

Veranderingen binnen het fysieke deel van netwerk zijn niet nodig. In overleg met de systeembeheerder kunnen de instellingen van de firewall en de proxy worden aangepast voor het gebruik van de beveiligingsapplicatie. De exacte poorten en protocollen zijn nog niet bekend, maar de mogelijkheid om deze te gebruiken is aanwezig.

De IIS 5.0 versie op de webserver zal moeten worden ge-upgrade naar versie 5.1. Dit is de minimale versie die wordt vereist door ASP.NET. Ook zal het .NET Framework moeten worden geïnstalleerd op de webserver.

Om de gegevens van de gemeente uit de Adabas database te kunnen synchroniseren met die van de database van het beveiligingssysteem kan er gebruik worden gemaakt van de service van Software AG. De request wordt gestuurd naar de NT4 server en deze vertaalt de request naar SNA. De request komt binnen op de IBM server en zal moeten worden verwerkt. Voor de verwerking van de request kan er een service worden geschreven in Natural. Natural is een programmeertaal die wordt geleverd voor communicatie met de Adabase database. De Natural service voert de request uit en geeft de response terug aan de NT4 server. Deze vertaalt hem van SNA naar TCP/IP en stuurt hem naar de beveiligingsapplicatie.

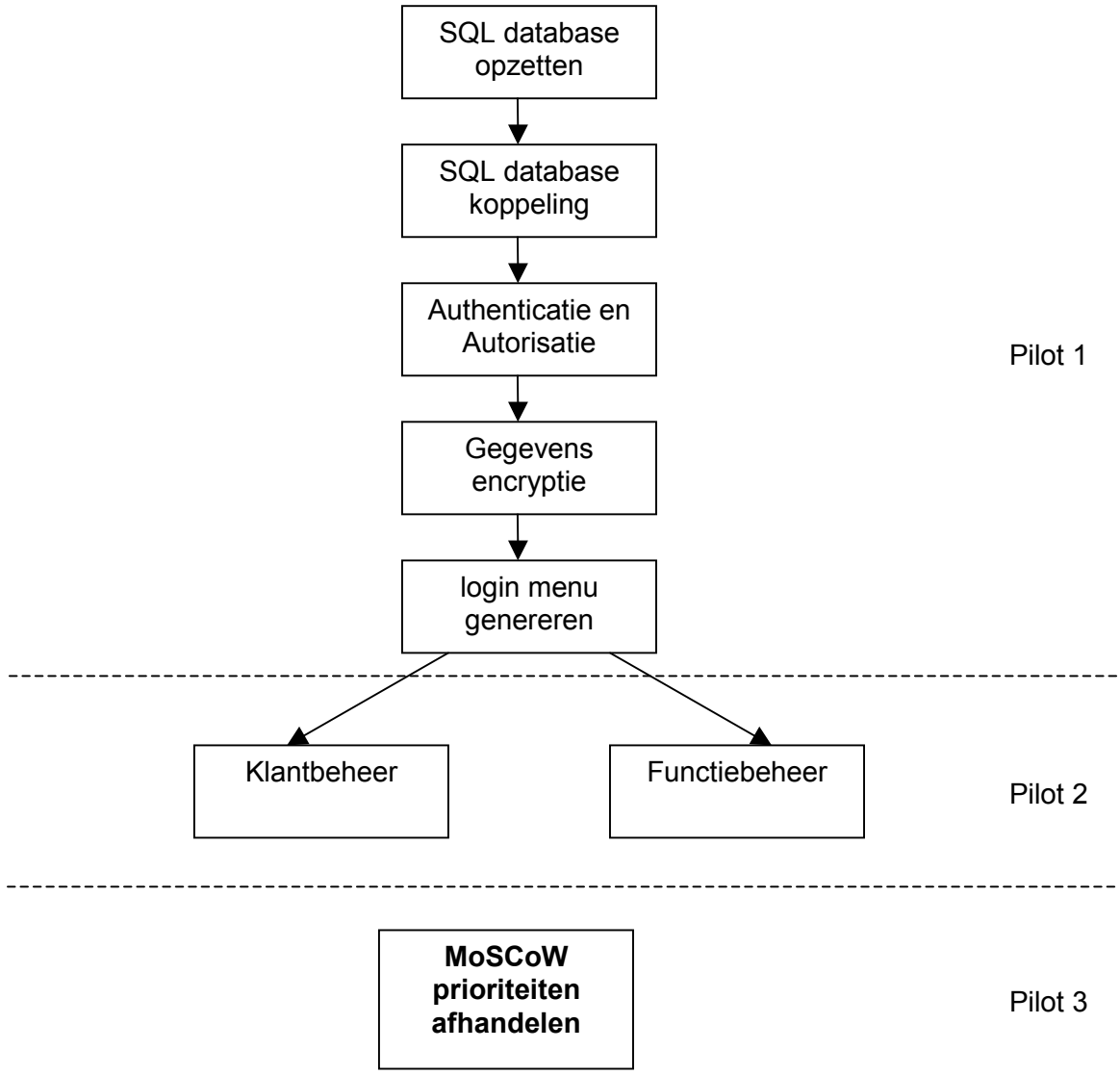
7 Pilot ontwikkelplan

7.1 Inleiding

Het pilotplan bestaat uit een aantal geprioriteerde pilots, die sequentieel ontwikkeld zullen worden. Elke pilot heeft betrekking op een bepaald deel uit het systeemconcept. Voor elke pilot is er een inschatting gemaakt van de kosten, benodigde middelen en tijd.

7.2 Pilotstructuur

In het onderstaande figuur zijn delen van het systeem gecombineerd tot compacte coherente clusters die ontwikkeld worden als subsets van het systeem.



De eerste pilot zal bestaan uit het opzetten van de SQL database, de SQL databasekoppeling, de authenticatie en verificatie van de gebruikers en de manier waarop de gegevens beveiligd over het Internet worden verstuurd. Ook zal er een login-menu worden gegenereerd aan de hand van de ingelogde gebruiker.

Tijdens de tweede pilotfase zal er een klant- en functiebeheer deel worden ontwikkeld voor de applicatie.

De laatste fase bestuut uit het toevoegen van de eisen en wensen met de hoogste prioriteit (MoSCow). Uitgaande dat alle Must Have's al zijn verwerkt tijdens de voorgaande fasen zullen nu de "Should haves", "Could haves" en "Would be nice to have" worden verwerkt.

7.3 Faseplanning Pilotontwikkeling

Tijdens de pilotontwikkelfase worden er een aantal "subfasen" uitgevoerd. Hieronder zijn deze subfasen weergegeven met de daarbij behorende belangrijkste activiteiten:

1. **Pilot 1: database en beveiliging**
   - SQL database implementatie model;
   - SQL database koppeling;
   - Authenticatie en Autorisatie;
   - Login menu genereren;
   - Gegevens encryptie.
2. **Definitiestudie**
   - Pilot evaluatie;
   - Indien nodig, verfijnen of uitbreiden eisen en wensen;
   - Indien nodig, systeemconcept verfijnen of uitbreiden;

   Indien nodig, technische structuur verfijnen of uitbreiden;
   - Indien nodig, pilotontwikkelplan verfijnen of uitbreiden.
3. **Pilot 2: klant- en functiebeheer**
   - Klantbeheer;
   - Functiebeheer.
4. **Definitiestudie**
   - Indien nodig, verfijnen of uitbreiden eisen en wensen;
   - Indien nodig, systeemconcept verfijnen of uitbreiden;
   - Indien nodig, technische structuur verfijnen of uitbreiden;
   - Indien nodig, pilotontwikkelplan verfijnen of uitbreiden.
5. **Pilot 3: MoSCoW prioriteiten**
   - MoSCoW Prioriteiten afhandelen.
   - Definitiestudie en betreffende pilotrapporten aanpassen op de wijzigingen.

Omdat er wordt gewerkt volgens de IAD ontwikkelmethode zal na elk opgeleverd pilotontwikkelrapport, de fase definitiestudie opnieuw worden doorlopen. De definitiestudie hoeft hierbij niet opnieuw te worden gerealiseerd maar wordt tijdens het doorlopen verfijnd of uitgebreid.

Omdat er gebruik wordt gemaakt van de iteratiestrategie Big-bang-invoeren worden de definitiestudie en de pilotontwikkeling iteratief uitgevoerd totdat pilot 3 is ontwikkeld. De pilotontwikkeling fase levert drie pilotrapporten op.

Bij het afhandelen van de MoSCoW prioriteiten worden de wensen die nog niet zijn gerealiseerd (alle anders dan must have's) in het systeem verwerkt. Dit gebeurt aan de hand van de prioriteiten zoals die zijn toegekend met de MoSCoW regels.

Elke pilotfase bevat de volgende activiteiten:

- Globale functiestructuur
- Globale technische structuur
- Pilotontwikkelplan
- Ontwerp software bouweenheden
- Bouw software bouweenheden
- Aanpassen externe componenten (indien van toepassing)
- Wijzigen andere informatiesystemen (indien van toepassing)
- Integreer bouweenheden
- Beoordelen en testen

Hieronder is in procenten aangegeven hoeveel tijd naar schatting nodig is voor het uitvoeren van een fase.

Fase	Geschat percentage
Pilot 1: database en beveiliging	35%
Definitiestudie	3%
Pilot 2: klant- en functiebeheer	35%
Definitiestudie	3%
MoSCoW prioriteiten afhandelen (incl. aanpassen definitiestudie)	25%

7.4 Urenplanning

De fase pilotontwikkeling zal worden doorlopen in de periode van 9 januari 2004 tot en met 17 maart. Dit resulteert in 49 beschikbare werkdagen ofwel 392 beschikbare uren. Ook wordt er in deze periode aan het eindverslag gewerkt. Dit neemt ongeveer 60 uur in beslag. Zo blijft er 332 uur over voor de pilootontwikkelingsfase.

Fase	Minimaal	Maximaal	Gemiddeld	Planning
Pilot 1	100	140	110	120
Definitiestudie	12	22	16	16
Pilot 2	100	140	110	120
Definitiestudie	6	12	8	8
MoSCoW prioriteiten afhandelen	30	95	74	68
Eindverslag	40	80	65	64
			Totaal:	392

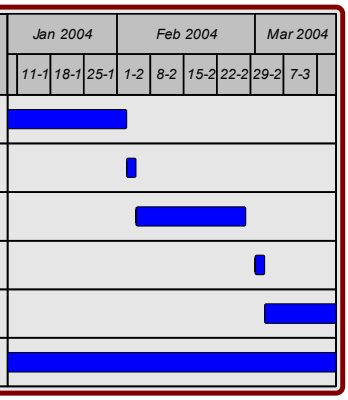
(minimaal + maximaal + 2 * gemiddeld) / 4 = planning)

De fase waarin de MoSCoW prioriteiten worden afgehandeld hangt af van de overgebleven tijd. Deze fase zal beëindigen zodra de beschikbare tijd over is of als alle wensen in het systeem verwerkt zijn. De 68 uren die voor deze fase is gepland is de verwachte tijd die over en nodig is voor deze het verwerken van de overige eisen.

7.5 Overzicht planning

Hieronder is een schema weergegeven met hierop aangegeven wanneer er aan welke fase wordt gewerkt.

ID	Task Name	Start	Finish	Duration
1	Pilot 1	9-1-2004	2-2-2004	17d
2	Definitiestudie	3-2-2004	4-2-2004	2d
3	Pilot 2	5-2-2004	27-2-2004	17d
4	Definitiestudie	1-3-2004	2-3-2004	2d
5	Pilot 3	3-3-2004	17-3-2004	11d
6	Eindverslag	9-1-2004	17-3-2004	49d



8. Conclusie

Met behulp van dit document is er een akkoord bereikt met de opdrachtgever over het te ontwikkelen systeem. Tevens is er uit het document gebleken dat het te realiseren systeem technisch haalbaar is binnen de huidige organisatie.

De volgende fase is de pilotontwikkeling waarin de pilots ontworpen en gebouwd zullen worden.

Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Document**
  Pilotontwikkeling: Pilot 1/3
- **Auteur**
  Mark van Houten (20015081)
- **Datum**
  26 juli 2004
- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag
- **Opleidingsvariant**
  MBIT
- **Periode**
  November 2003 – april 2004
- **Opdrachtgever**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft
- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Voorwoord

Voor u ligt het rapport “pilotontwikkeling: pilot 1” ten behoeve van een Web based beveiligingssysteem met functie en klantbeheer voor de SMRA. Dit rapport is tot stand gekomen in overleg met dhr R. Mehlkopf en dhr C. Schaaf.

De basis van dit rapport is de eerder opgeleverde definitiestudie. Het deel van het daarin ontworpen systeemconcept zal in dit document tot in detail worden uitgewerkt. Dit rapport vormt de basis voor het bouwen van het systeem.

Delft, SMRA, 26 juli 2004

Inhoudsopgave

1. Inleiding

Het doel van de pilotontwikkeling is het ontwerpen van een pilot. De pilot zal worden ontwikkeld aan de hand van de eisen die eerder zijn opgesteld in de definitiestudie. Een pilot biedt een operationele subset van de functionaliteit van het uiteindelijk beoogde systeem.

Als eerste zal er een globale functiestructuur van de pilot worden gemaakt. Het systeemconcept zal hier worden verfijnd.

Hierna zal de technische functiestructuur globaal rond de pilotontwikkeling worden gerealiseerd. De fysieke allocatie, het technische opslagmodel, het netwerkcommunicatiemiddel en de externe interfaces worden hier beschreven.

Er zal een pilotontwikkelplan moeten worden opgesteld op basis van de gemaakte globale functionele en technische specificaties. Het pilotontwikkelplan beschrijft hoe de pilot gaat worden gebouwd.

Op basis van bovenstaande onderdelen is het mogelijk om een ontwerp te maken van de software bouweenheden. De componenten zullen worden afgebeeld op de software architectuur. De detailgegevenspecificaties, detailprocesspecificaties en de testspecificaties worden besproken.

Nadat het ontwerp is gemaakt van de software bouweenheden is het mogelijk de bouweenheden daadwerkelijk te gaan bouwen. De fysieke database zal worden gerealiseerd, de gebruikersinterfaces zullen worden gebouwd en de gebouwde bouweenheden zullen worden getest.

Mochten er externe componenten of informatiesystemen (herbruikbare componenten of informatiesystemen) worden gebruikt, dan zullen deze wellicht moeten worden aangepast. Voor elk te gebruiken externe component of informatiesysteem zal dan ook een ontwerp met aanpassingen worden gemaakt, die zullen worden geïmplementeerd en getest.

Nadat alle bouweenheden zijn gebouwd en getest, kunnen deze worden geïntegreerd en kan worden gecontroleerd of de bouweenheden op de juiste manier met elkaar communiceren.

Uiteindelijk vindt het beoordelen van de pilotontwikkeling plaats. Er wordt een gezamenlijke test gedaan en hierover wordt een oordeel gegeven.

2. Globale functiestructuur

2.1 Inleiding

In dit deel van het document wordt de functionele inhoud van de pilot bepaald. In de definitiestudie is de functionele architectuur besproken (in het systeemconcept) die hier zal worden verfijnd. In dit onderdeel wordt gegeven: het functionele procesmodel, prototype gebruikers interface, conceptueel gegevensmodel en eventueel functionele aanpassingen aan externe componenten.

De sequencediagrammen worden gegeven, met daarin de informatie over het interne gedrag van het systeem. De basis wordt gevormd door de use-cases uit de definitiestudie, tenzij het gaat over een procedure die binnen het systeem door een andere procedure wordt aangeroepen. In elk sequencediagram worden de boodschappen tussen de objecten aangegeven met de nadruk op het tijdsaspect.

Er worden prototypen gegeven van de gebruikersinterface zodat de opdrachtgever een indruk krijgt over de gebruikersinterface van het te ontwikkelen systeem. Dit geeft de opdrachtgever de mogelijkheid om zijn argumenten te laten verwerken in de werkelijke applicatie. Ook neemt het vaak twijfels bij de opdrachtgever weg over de functionaliteiten en de structuur van de applicatie.

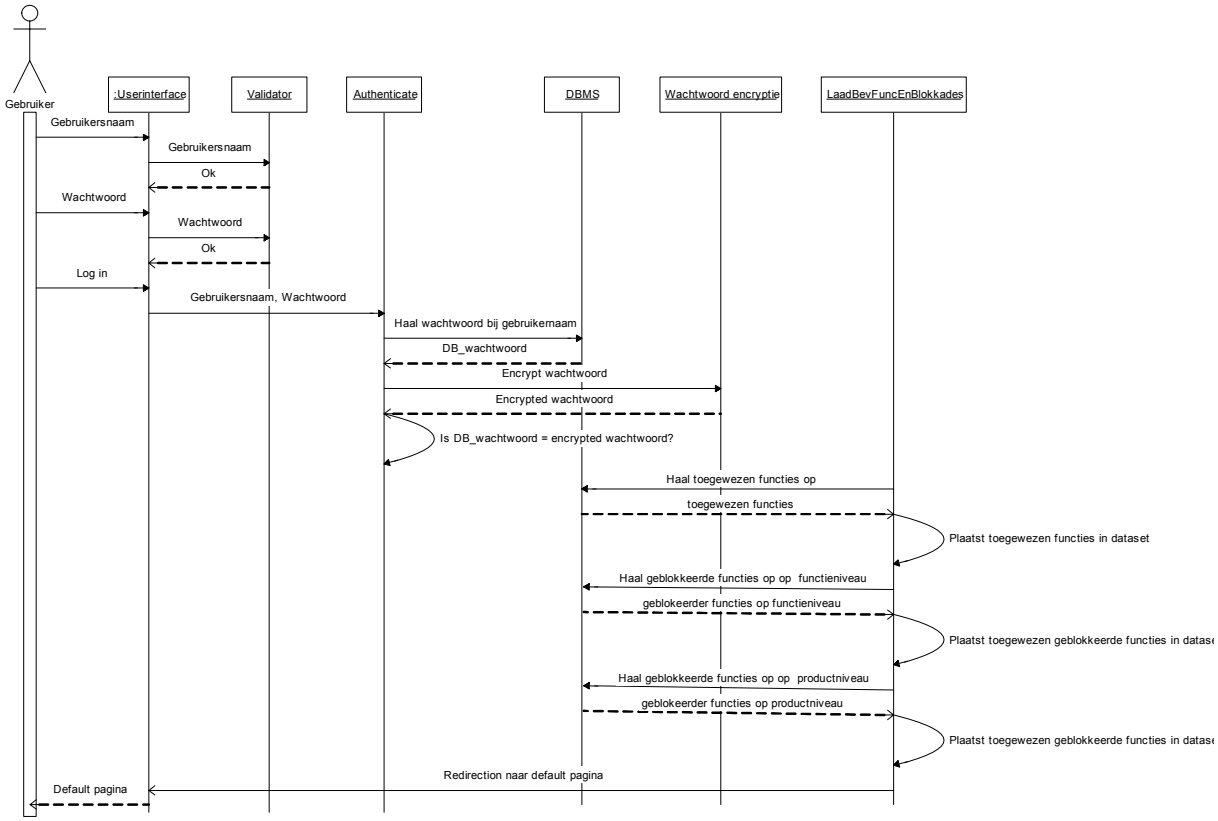
In het conceptueel gegevensmodel worden de entiteiten en hun relaties gedetailleerder beschreven, inclusief de gegevenselementen die ze bevatten (attributen) en de toegestane toestanden van de entiteiten.

Als laatste deel van dit onderdeel wordt er gekeken naar eventuele functionele aanpassingen van externe systemen voor de integratie met het te ontwikkelen systeem.

2.2 Sequencediagrammen

2.2.1 Authentificatie

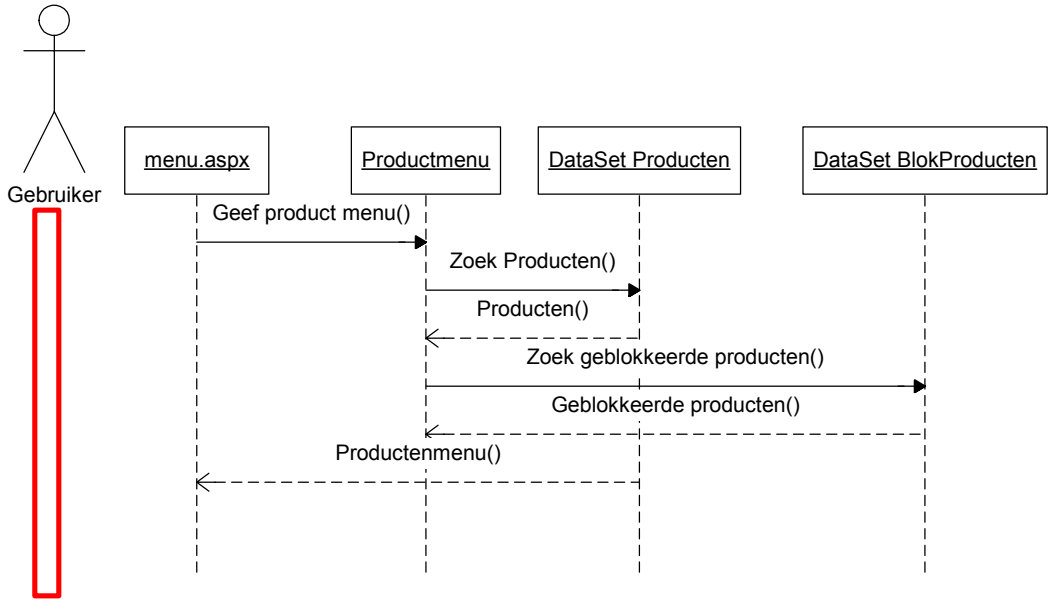
Voordat een gebruiker gebruik mag maken van het systeem moet deze zich authentificeren. Dit wordt gedaan door middel van de authentificatie procedure. Een gebruiker maakt zijn naam en bijbehorend wachtwoord bekend en de procedure controleert of de gebruiker bekend is bij het systeem. In werkelijkheid zullen er tijdens deze procedure meer handelingen worden gedaan. Deze zijn in het volgende overzicht gegeven.



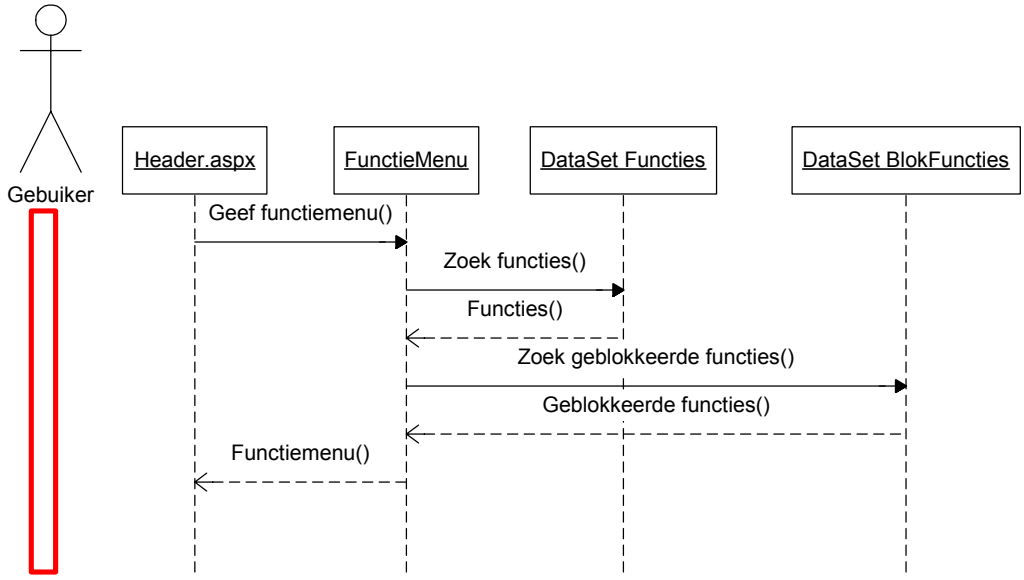
2.2.2 Automatisch genereren menu's

Als een gebruiker is geauthentificeerd, wordt hij doorgestuurd naar de standaard pagina van de webapplicatie (default.aspx), welke is gegenereerd uit meerdere pagina´s (frames). Afhankelijk van de gebruiker, de beschikbare gestelde functies, de blokkades en de beschikbaar gestelde functiegroepen wordt de hoofdpagina gegenereerd. Deze pagina geeft de gebruiker de mogelijkheid om de functies te gebruiken waartoe hij geautoriseerd is. Ook in deze procedure zullen er in de werkelijkheid meerdere handelingen worden verricht. Hieronder is een schematisch overzicht gegeven van de procedure voor het genereren van het productenmenu met daarin de handelingen die worden uitgevoerd tijden het uitvoeren van deze procedure.

Producten menu:

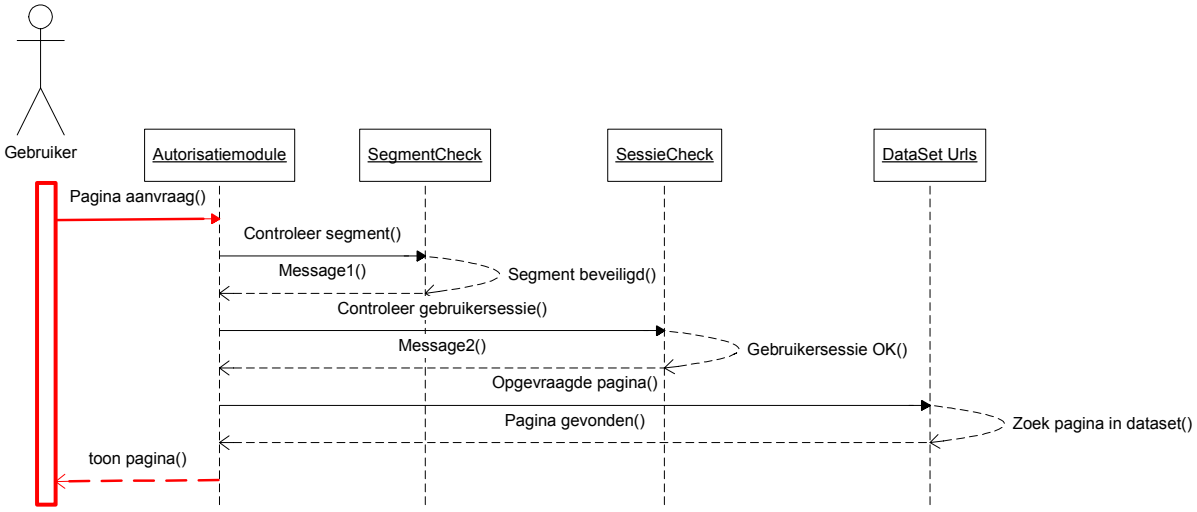


Functiemenu:

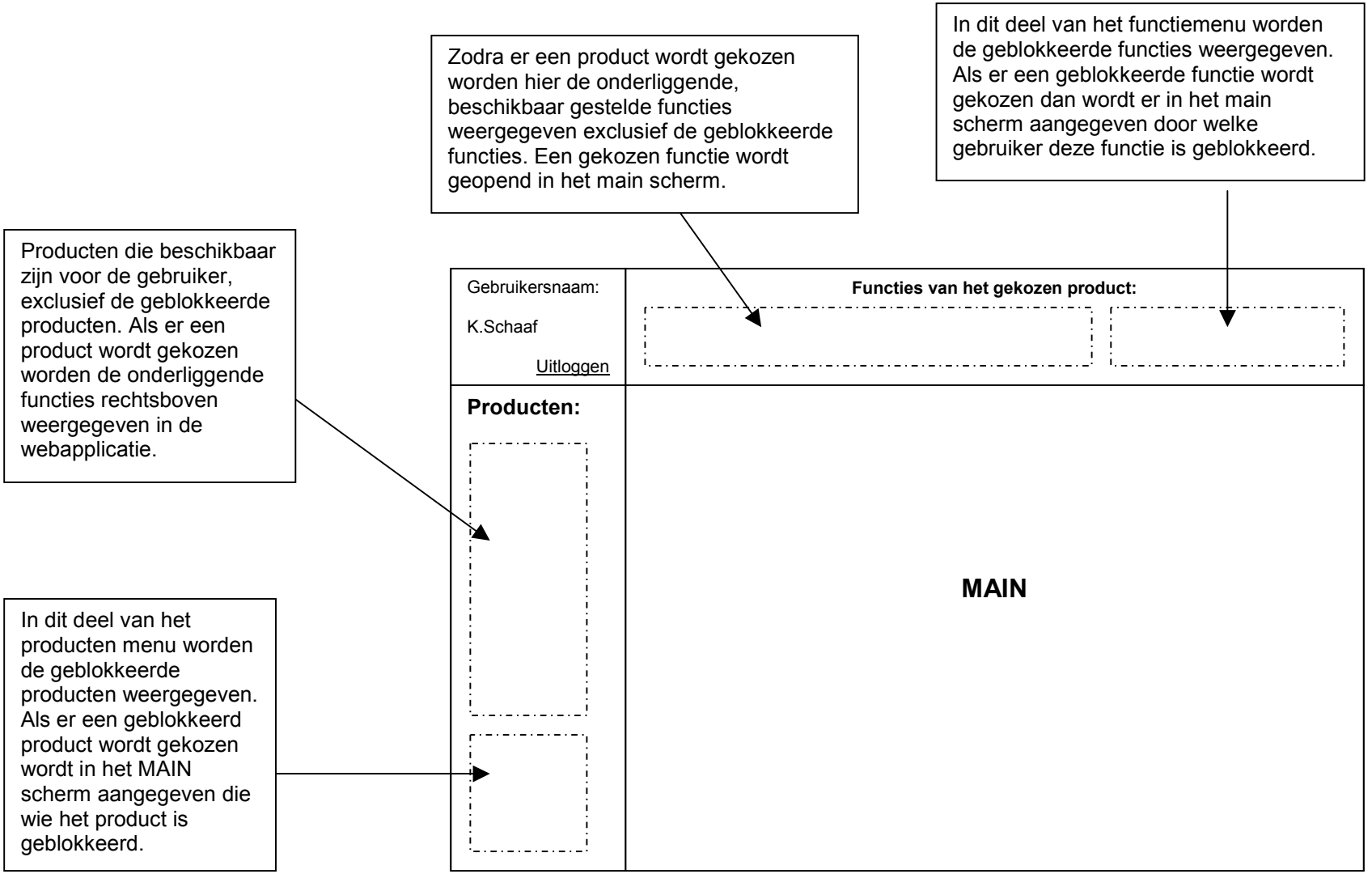


2.2.3 Autorisatie

Als een gebruiker een pagina opvraagt moet er voor een tweede maal gecontroleerd worden of de gebruiker geautoriseerd is tot het opvragen van de pagina. Dit voorkomt dat na het inloggen een gebruiker een willekeurige pagina in kan typen (om zo een functie op te vragen) en deze kan uitvoeren.
Om deze reden worden de pagina´s die functies bevatten ondergebracht in een aparte virtuele map met een eigen web.config. Aan deze web.config kan een http module worden gekoppeld die in werking treedt zodra een gebruiker een pagina wil opvragen uit de betreffende virtuele map. Bij elke aanvraag van een ASP.NET pagina zal deze module worden toegepast. Deze module controleert de of de gebruiker toegang heeft tot de opgevraagde URL en of er geen blokkades op deze URL zijn geplaatst die voor deze gebruiker van toepassing zijn.

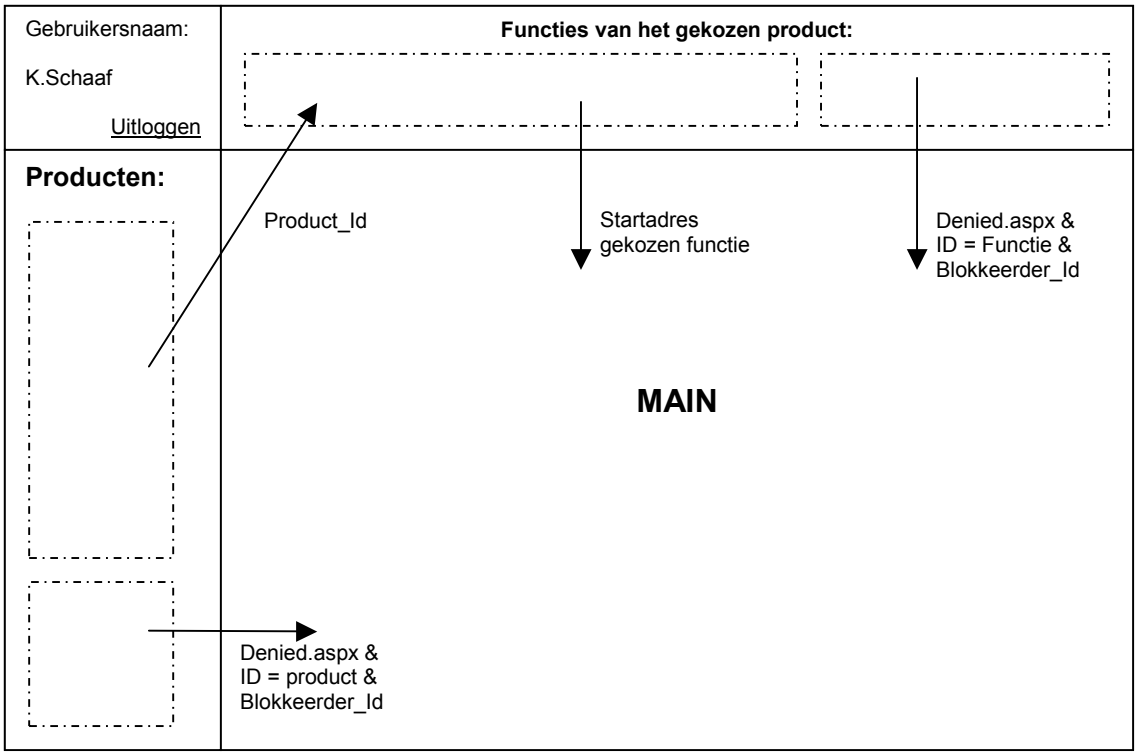


2.3 Grafische structuur webapplicatie



2.4 Menu navigatie

Omdat er wordt gewerkt met persoonlijke menu's en submenu's dient er gebruik te worden gemaakt van navigatie mechanisme die gebruikerafhankelijk de menu's bepaald. Hieronder is een schematisch overzicht gegeven van de manier waarop er binnen de menu's genavigeerd wordt.



Als de gebruiker is ingelogd wordt hij automatisch doorgestuurd naar de standaard pagina (default.aspx). Vanuit deze pagina kan hij één van de beschikbare producten kiezen en de onderliggende functie worden daarna weergegeven in het functiemenu.
Als er een geblokkeerd product wordt gekozen wordt de denied.aspx pagina aangesproken en wordt er door deze pagina weergegeven door wie de blokkade is geplaatst.

Het functiemenu wordt gegenereerd aan de hand van een gekozen product. Zodra er een product wordt gekozen wordt het product_Id doorgegeven aan het functiemenu (header.aspx) en worden de toegewezen en geblokkeerde functies in het functie menu weergegeven. Indien er een functie wordt gekozen wordt de functie geopend in het main gedeelte van het scherm. Indien er een geblokkeerde functie wordt gekozen wordt net als bij het kiezen van een geblokkeerd product, de denied.aspx opgevraagd. Ook hier zal de denied.aspx bepalen door de blokkade is geplaatst en deze informatie weergegeven aan de gebruiker.

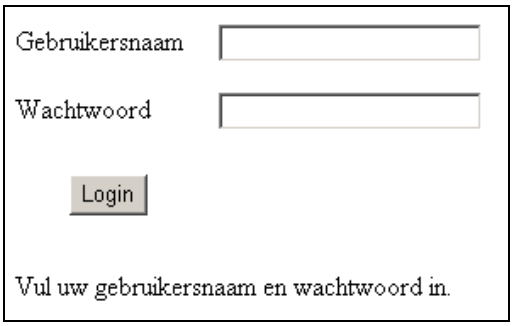
2.5 Prototype gebruikersinterface

In dit onderdeel zullen de prototypen van de gebruikersinterfaces aan bod komen.
De webinterface zal bestaan uit een login-pagina waar de gebruikers kunnen inloggen. Dit gebeurt door middel van het invoeren van een gebruikersnaam en bijbehorend wachtwoord.

2.5.1 Algemeen

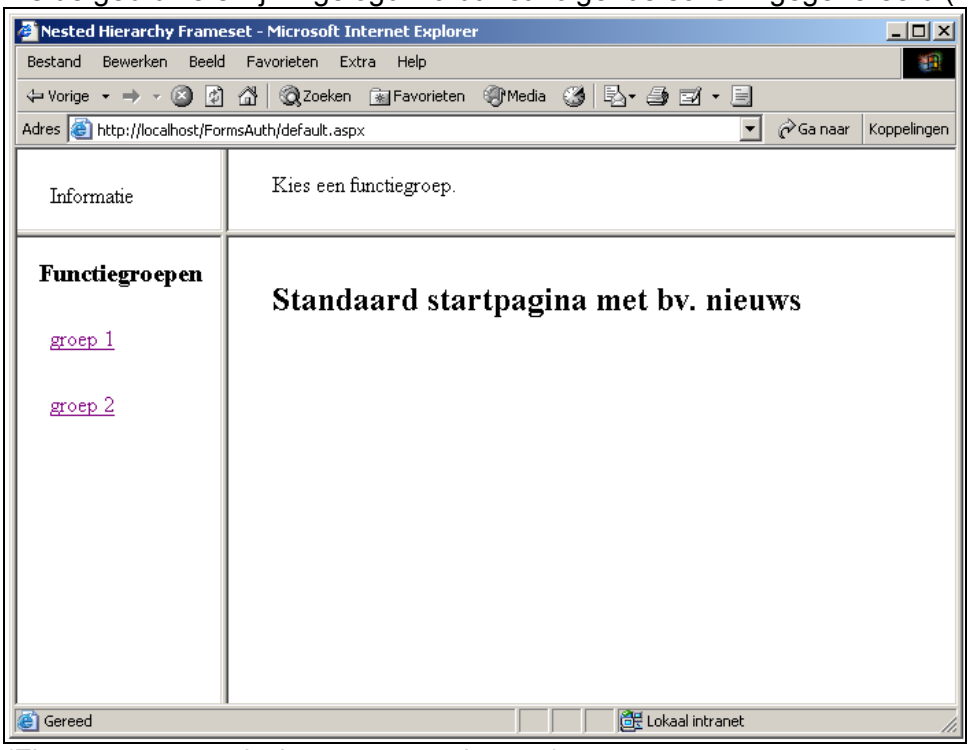
Geldig voor SMRA beheerders, gemeentelijke beheerders en functionarissen.

Hieronder is het login scherm weergegeven waarin gebruikers hun gebruikersnaam en bijbehorend wachtwoord moeten invoeren (Figuur 4).



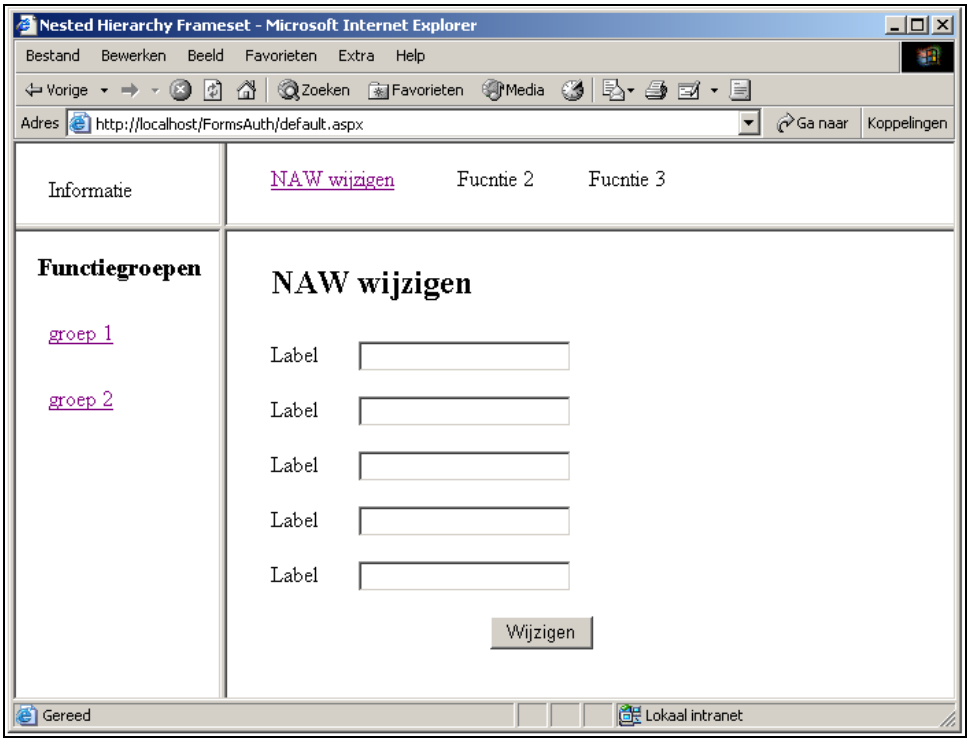
(Figuur 4: login Scherm)

Als de gebruikers zijn ingelogd wordt het volgende scherm gegenereerd (Figuur 5).



(Figuur 5: automatisch gegenereerd menu)

De functiegroepen staan aan de linker zijde van de pagina. Als één van de functiegroepen wordt aangeklikt, worden de onderliggende functies uit de functiegroep geopend in het frame met daarin “Kies een functiegroep”. Als bijvoorbeeld de eerste functie groep wordt gekozen verschijnt het volgende scherm (Figuur 6):

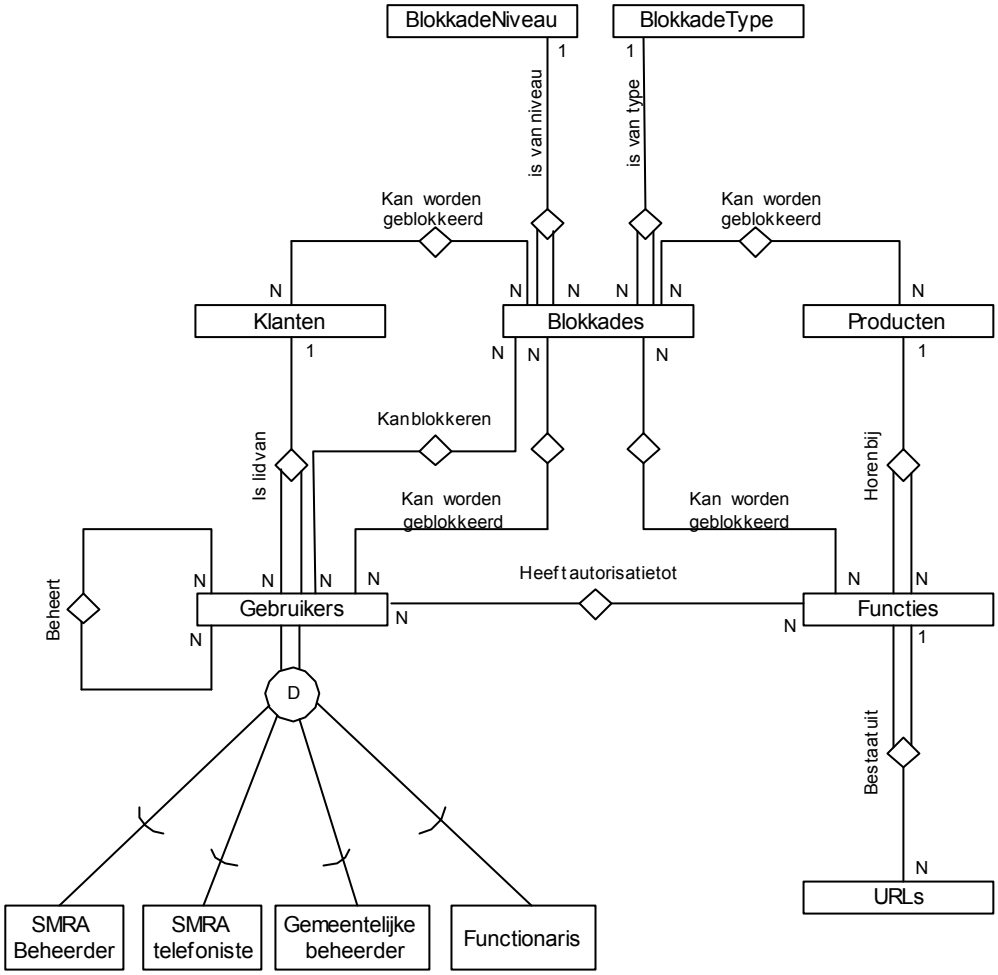


(Figuur 6: NAW gegevens wijzigen)

Dit scherm is alleen van toepassing voor gemeentelijke beheerders en functionarissen. In dit scherm is de functie: “NAW wijzigen” gekozen. Elke functie wordt geopend in het “hoofd” scherm waar de functie kan worden uitgevoerd.

2.6 Conceptueel gegevensmodel (ERD)

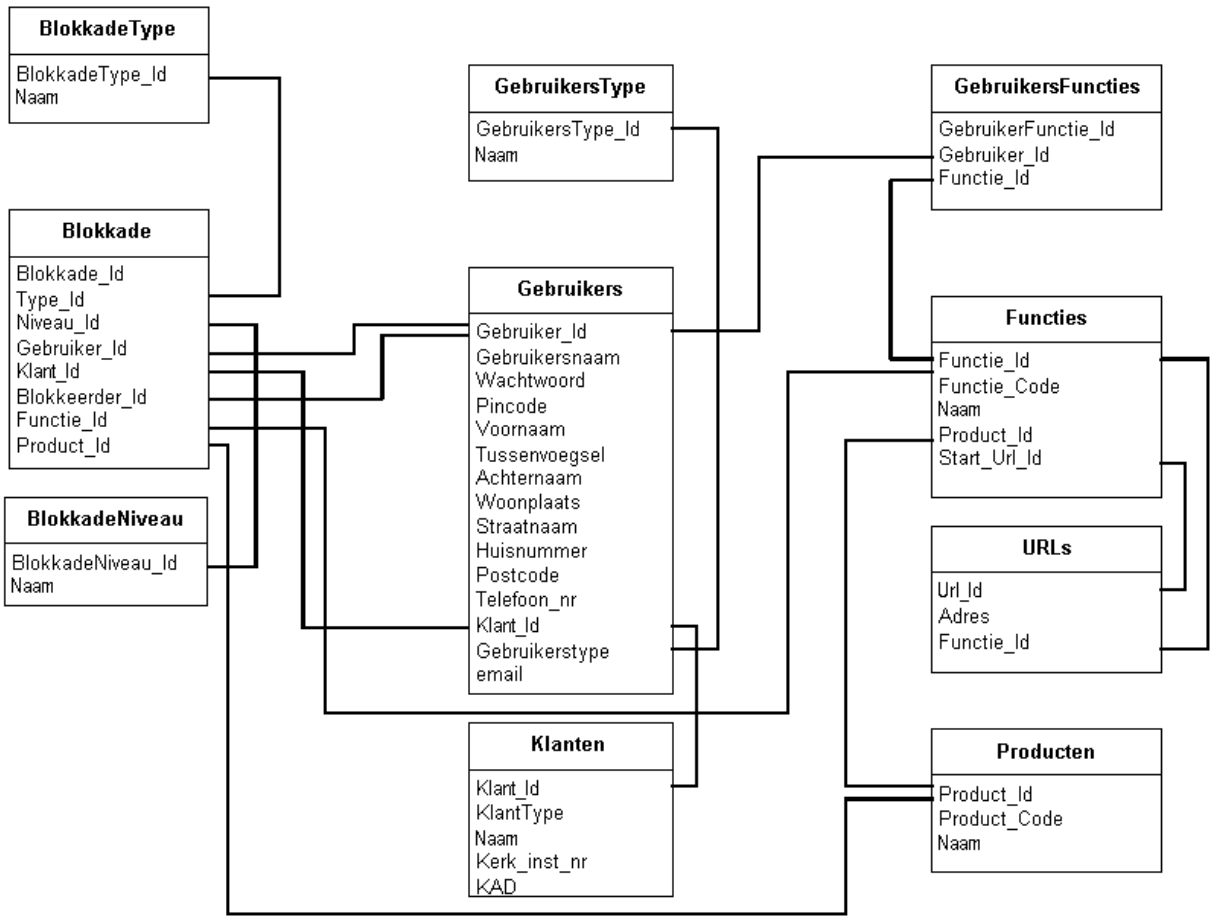
Hieronder is het conceptueel gegevenmodel gegeven. Dit model geeft een representatie van de gegevensstructuur zoals die wordt geïmplementeerd in de database.



De entiteiten BlokkadeNiveau en BlokkadeType zijn als entiteiten opgenomen omdat dit de mogelijkheid creëert om in de toekomst via een user interface een blokkadeniveau of blokkadetype toe te voegen.

Overzicht tabellen, attributen en relaties

Hieronder is een schema gegeven met daarin een overzicht van de tabellen met bijbehorende gegevens en de relaties tussen de tabellen.



2.7 Functionele aanpassingen externe systemen

2.7.1 Aanpassingen aan de webserver

Om ASP.NET op de webserver te kunnen gebruiken is het noodzakelijk dat op de webserver het .NET framework wordt geïnstalleerd. Zo wordt IIS 5.0 naar een nieuwe versie gebracht (5.1) en kunnen ASP.NET extensies worden opgevangen door ASP.NET.

Ook moet er op de webserver een SSL certificaat worden ingesteld voor het beveiligingssysteem. Hiernaast moet er in het domein “SMRA” een gebruiker worden aangemaakt die voor Impersonation kan worden gebruikt. Deze gebruiker dient ook de nodige rechten te krijgen over de database die gekoppeld is aan de webapplicatie.

2.7.2 Gegevensencryptie

Voor de encryptie van de gegevens tussen de webserver en de cliënt wordt er gebruik gemaakt van SSL (Secure Socket layer, uitgebreidere informatie in het evaluatie rapport over Microsoft.NET en de beveiliging binnen ASP.NET).

Voor alle delen van het beveiligingssysteem zal SLL worden gebruikt zodat zogenoemde “hackers” geen wachtwoorden kunnen achterhalen of persoonlijke informatie van personen kunnen inzien.
De wachtwoorden van de gebruikers van het beveiligingssysteem zullen ge-encrypt worden opgeslagen. Dit wordt gedaan met behulp van SHA1 (Secure Hash Algorithm). Dit is een 1-way encryptiemethode. Dit houdt in dat het vrijwel onmogelijk is om vanuit het ge-encrypte opgeslagen wachtwoord het oorspronkelijke wachtwoord te herhalen. Dit voorkomt ook dat de beheerders wachtwoorden van hun leden kunnen uitlezen.

2.7.3 Impersonation, delegation

Om een connectie op te zetten tussen de webapplicatie en de SQL 2000 server zal er gebruik worden gemaakt van Impersonation. Binnen de applicatie zal er gebruik worden gemaakt van autorisatie en authentificatie op 2 niveau’s. Dit wordt gedaan door zowel IIS als het .NET framework. De gebruiker krijgt van IIS het account toegewezen dat aan de anonieme gebruiker is gekoppeld.
Binnen ASP.NET wordt de gebruiker gekoppeld aan het ASPNET user account. Als Impersonation wordt gebruikt wordt de gebruiker gekoppeld aan het IIS anonieme gebruikersaccount als de gebruiker een bepaalde bron opvraagt.
Door aan de anonieme gebruiker een gebruiksaccount van een domein te koppelen kan de webapplicatie bronnen benaderen op basis van het gekoppelde account (IIS account). Zo kan er een connectie worden gevormd tussen een SQL server en op grond van de autorisaties van het gekoppelde account kunnen bepaalde databases binnen de server worden benaderd.
De toegang tot de SQL database wordt zo verkregen door een account die rechten heeft tot de database te koppellen aan de anonieme gebruikersaccount van IIS en gebruik te maken van Impersonation.

2.7.4 Synchronisatie Adabas database gemeenten.

Binnen de Adabas database worden de kerkelijk gemeenten bijgehouden. Aan deze gemeenten worden een nummer en een naam toegekend. Omdat er relaties nodig zijn tussen de gebruikers van het beveiligingssysteem en de gemeenten is het nodig om de gegevens uit de Adabas database te kunnen synchroniseren met de gegevens uit de SQL server. In de SQL server zullen de kerkelijke gemeenten worden opgeslagen onder de naam klanten.

3. Globaal technische structuur

3.1 Inleiding

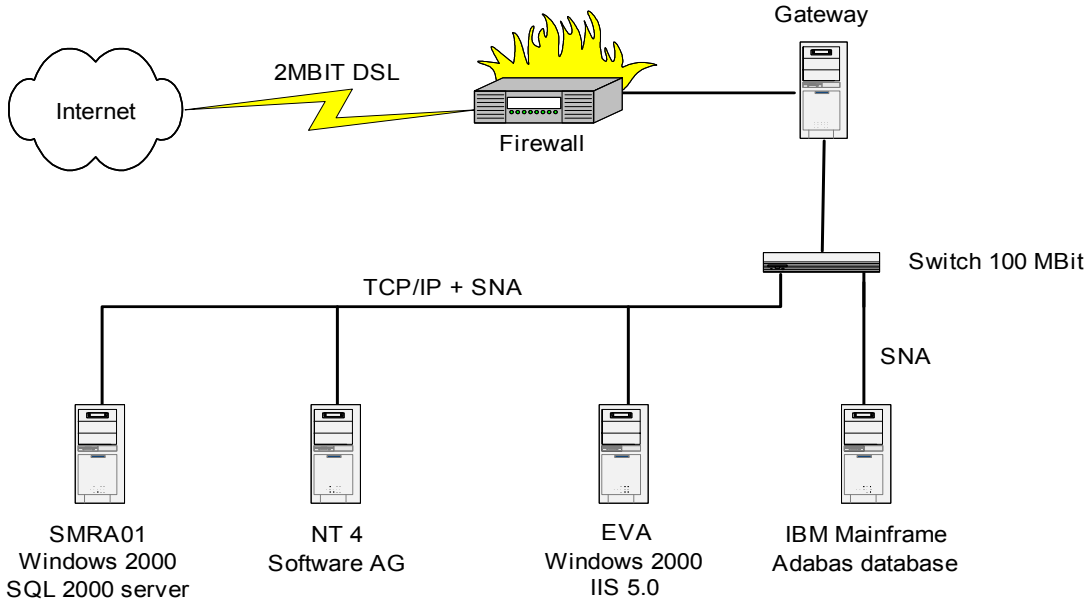
In dit onderdeel van het document worden de technische aspecten rond de pilotontwikkeling gespecificeerd. In dit onderdeel wordt gegeven: de fysieke allocatie, technisch opslagmodel, het netwerkcommunicatiemodel en de externe interfaces.

3.2 Fysieke allocatie

Voor het hosten van de webapplicatie zal er gebruikt worden gemaakt van de IIS server op de EVA (Intranetnaam) machine. Dit is een Windows 2000 machine die momenteel de standaard website van de SMRA host. De SQL server die gekoppeld zal worden aan de website zal worden geplaatst op de SMRA01 machine. Dit is een SQL 2000 server met Windows 2000 als besturingssysteem.

Ook zal de webapplicatie gebruik maken van de gemeenten die zijn opgeslagen in de Adabas database. Deze database bevindt zich in het mainframe gedeelte van het intranet van de SMRA. Deze server kan alleen worden aangesproken door gebruik te maken van een brokerservice. De brokerservice bevindt zich op de machine NT01SMRA. Dit is een Windows NT4 machine met een broker service van Software AG. De brokerservice roept een procedure aan op het mainframe die de gegevens uit de database leest. De brokerservice geeft de resultaten terug aan de procedure van de webapplicatie. Deze procedure zal de verkregen gegevens synchroniseren met de gegevens uit de SQL database.

Sommige resultaten van de queries die naar het SQL DBMS worden gestuurd worden tijdens de sessie opgeslagen in datasets. Deze gegevens zijn onder andere de autorisatiegegevens, authenticatie gegevens en de blokkade gegevens. Deze gegevens worden bij elke sessie vernieuwd en zijn gekoppeld aan de gebruiker(s).



3.3 Technisch opslagmodel

3.3.1 Implementatiemodel

```
CREATE TABLE BlokkadeNiveau (
        BlokkadeNiveau_Id       tinyint    IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Naam                    varchar (20) NOT NULL
)
primary key (BlokkadeNiveau_Id)


CREATE TABLE BlokkadeType (
        Blokkadetype_ID         tinyint IDENTITY (1, 1) NOT NULL ,
        Naam                    char (10) NOT NULL
)
primary key (BlokkadeType_Id)


CREATE TABLE Blokkades (
        Blokkade_Id             numeric(18, 0) IDENTITY (1, 1) NOT NULL ,
        Type_Id                 tinyint NOT NULL ,
        Niveau_Id               tinyint NULL ,
        Gebruiker_Id            numeric(18, 0) NULL ,
        Klant_Id                numeric(18, 0) NULL ,
        Functie_Id              numeric(18, 0) NULL ,
        Product_Id              numeric(18, 0) NULL ,
        Blokkeerder_Id          numeric(18, 0) NOT NULL
)
primary key (Blokkade_Id)
foreign key (Type_id)           references BlokkadeType         on delete cascades, on update no action
foreign key (Niveau_id)         references BlokkadeNiveau       on delete cascades, on update no action
foreign key (Gebruiker_id)      references Gebruikers           on delete cascades, on update cascades
foreign key (Klant_id)          references Klanten              on delete cascades, on update cascades
foreign key (Blokkeerder_id)    references Gebruikers           on delete no action, on update cascades
foreign key (Functie_id)        references Functies             on delete cascades, on update cascades
foreign key (Product_id)        references Producten            on delete cascades, on update cascades


CREATE TABLE Functies (
        Functie_ID              numeric(18, 0) IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Functie_Code            varchar (10) NOT NULL ,
        Naam                    varchar (100) NOT NULL ,
        Product_Id              numeric(18, 0) NOT NULL
)
primary key (Functie_Id)
foreign key (Product_id)        references producten            on delete no action, on update no action


CREATE TABLE GebruikerFuncties (
        GebruikerFuncties_ID    numeric(10, 0) IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Gebruiker_ID            numeric(18, 0) NOT NULL ,
        Functie_ID              numeric(18, 0) NOT NULL
)
primary key (GebruikerFuncties_ID)
foreign key (Gebruiker_id)      references Gebruikers           on delete cascades, on update cascades
foreign key (functie_id)        references Functies             on delete cascades, on update cascades
```

```
CREATE TABLE Gebruikers (
        Gebruiker_ID            numeric(18, 0) IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Gebruikersnaam          varchar (10) NOT NULL ,
        Wachtwoord              varchar (50) NOT NULL ,
        Voornaam                varchar (20) NULL ,
        Tussenvoegsel           varchar (10) NULL ,
        Achternaam              varchar (40) NULL ,
        Woonplaats              varchar (30) NULL ,
        Straatnaam              varchar (30) NULL ,
        Huisnummer              varchar (6) NULL ,
        Postcode                varchar (10) NULL ,
        PinCode                 numeric(18, 0) NULL ,
        Telefoon_nr             char (11) NULL ,
        GebruikersType          tinyint NOT NULL ,
        Klant_Id                numeric(18, 0) NULL
)
primary key (Gebruiker_ID)
foreign key (Gebruikerstype)    references GebruikerType        on delete cascades, on update cascades
foreign key (Klant_Id)          references Klanten              on delete cascades, on update cascades


CREATE TABLE GebruikersType (
        GebruikersType_ID       tinyint NOT NULL ,
        Naam                    nvarchar (30) NULL
)
primary key (GebruikersType_ID)


CREATE TABLE KlantType (
        KlantType_Id            tinyint NOT NULL ,
        Naam                    varchar (20) NOT NULL
)
primary key (KlantType_Id)



CREATE TABLE Klanten (
        Klant_Id                numeric(18, 0) IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Klant_Type              tinyint NOT NULL ,
        Naam                    varchar (100) NOT NULL ,
        Kerk_Inst_Nr            char (10) NULL ,
        KAD                     smallint NULL
)
primary key (Klant_Id)
foreign key (Klant_Id)          references Klanten              on delete cascades, on update cascades


CREATE TABLE Producten (
        Product_ID              numeric(18, 0) IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Product_Code            varchar (10) NULL ,
        Naam                    varchar (50) NULL
)
primary key (Product_ID)


CREATE TABLE URLs (
        URL_ID                  numeric(18, 0) IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
        Adres                   varchar (40) NULL ,
        Functie_ID              numeric(18, 0) NULL
)
primary key (URL_ID)
```

3.3.2 Constraints

Blokkadetabel

```
([Type_Id] = 1
and [Niveau_Id] is not null
and [Functie_Id] is not null          (Functieniveau)
and [Product_Id] is null
or
[Type_Id] = 2
and [Niveau_Id] is not null
and [Product_Id] is not null          (Productniveau)
and [Functie_Id] is null
or
[Type_Id] = 3
and [Niveau_Id] is not null
and [Functie_Id] is null              (Gebruikerniveau, alle gebruikers)
and [Gebruiker_Id] is null
and [Klant_Id] is null
        or
        and [Niveau_Id] is not null
        and [Functie_Id] is null              (Gebruikerniveau, één gebruiker)
        and [Gebruiker_Id] is not null
        and [Klant_Id] is null
                or
                and [Niveau_Id] is not null
                and [Functie_Id] is null              (Gebruikerniveau, een gehele klant)
                and [Gebruiker_Id] is null
                and [Klant_Id] is not null)
```

3.4 Interfaces andere systemen

De Adabas database kan niet direct vanuit de ASP.NET applicatie worden aangesproken. De ASP.NET Applicatie zal een aanvraag moeten sturen naar de Software AG Broker Service, welke de aanvraag van TCP/IP zal moeten omzetten naar SNA. Deze aanvraag moet hierna worden doorgestuurd naar het mainframe en moet hier worden opgevangen door een Natural functie. De Natural functie zal aanvraag moeten opvangen en het resultaat van de aanvraag moeten terug geven aan de broker service. De brokerservice moet vertaald het resultaat van SNA naar TCP/IP en stuurt het terug naar de webapplicatie. Deze zal de gegevens moeten synchroniseren met de gegevens uit de SQL server.

3.5 Configuratie-instellingen beveiligingssysteem.

De werking van de webapplicatie maar ook het beveiligingsdeel is zeer afhankelijk van verschillende configuratie-instellingen. Om deze reden worden de belangrijke instellingen in dit onderdeel besproken.

3.5.1 Web.config

De web.config is een bestand dat wordt geplaatst in de root van een virtuele map. De web.config bevat de configuratie-instellingen die worden gebruikt binnen de virtuele map. (de machine.config levert de configuratie-instellingen voor ASP.NET, de web.config overerft deze configuratiesettings en kan deze wijzigen voor de betreffende map). Er wordt gebruik gemaakt van 2 virtuele mappen. Voor elk van deze virtuele mappen moet de web.config worden ingesteld.

Authentication
In de web.config moet de configuratie worden gegeven voor de authenticatie.

```
<authentication mode="Forms" >
      <forms loginUrl="LogIn.aspx">
      </forms>
</authentication>
```

Binnen het beveiligingssysteem wordt er gebruik gemaakt van de authenticatie mode: Forms. Er wordt geen gebruik gemaakt van Cookies omdat de applicatie wordt gekoppeld aan de sessie van de gebruiker(s). Het is dus noodzakelijk dat er geen cookie wordt meegegeven aan de forms authentication node.

Autorisation
In de web.config moet de configuratie worden gegeven voor de autorisatie.

```
<authorization>
      <deny users="?" />
</authorization>
```

omdat de autorisatie wordt gedaan door ASP.NET en niet door IIS wordt er gebruik gemaakt van de instelling: deny users=”?”. Dit zorgt ervoor dat alle onbekende gebruikers (door IIS toegewezen aan het anonymous user account) worden afgevangen en worden doorgestuurd om zich te authentificeren. Door de login pagina mee te geven in de authentication forms loginUrl=”login.aspx” worden alle gebruikers die de webapplicatie gebruiken doorgestuurd naar de login.aspx pagina. OP deze pagina kunnen zij zich authentificeren voor de webapplicatie en zodoende autorisatie tot de webapplicatie verkrijgen.

Gebruik Impersonation
Om gebruik te maken van Impersonation dient dit in de web.config te worden opgegeven. Dit wordt gedaan door de volgende instelling:

```
    <identity impersonate="true"/>
```

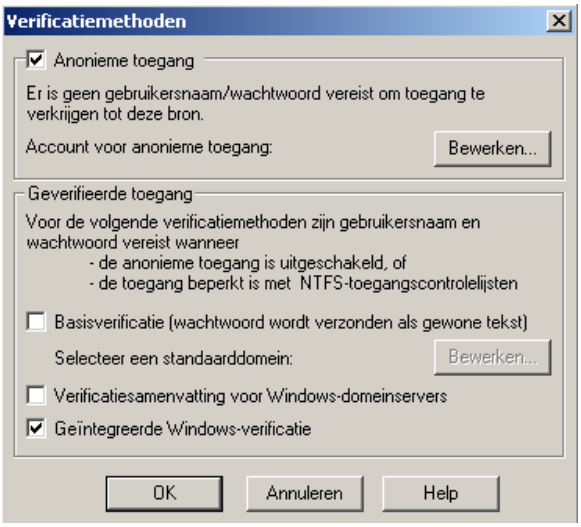
HTTP Module
Deze instelling moet alleen worden verwerkt in de map waarin de functies zijn opgeslagen. De httpmodule zal dan alleenw orden gebruikt zodra een gebruiker een pagina wil opvragen uit deze virtuele map. Dit wordt gedaan door de volgende instelling:

```
       <httpModules>
           <add name="AuthModule" type="AutModule.AuthModule, AutModule" />
       </httpModules>
```

3.5.2 IIS

Om de authenticatie en autorisatie te laten plaatsvinden door ASP.NET moeten er een aantal instellingen worden gedaan in de configuratie van IIS.

Anonymous user, ASP.NET user (Impersonation)
Om ASP.NET de authentificatie en autorisatie te laten afhandelen moet de beveiligingsapplicatie worden ingesteld om gebruik te maken van de anonymous user. De volgende instellingen dienen te worden overgenomen:



Als anonieme toegang is geselecteerd moet hieraan een account worden toegewezen. Klik op bewerken. Kies het domein gebruikersaccount gekoppeld met de nodige toegang op de SQL server. Vul het bijbehorende wachtwoord in en vink Wachtwoordbeheer door IIS niet aan!

SSL

Om de webapplicatie gebruik te laten maken van een SSL verbinding dient er een certificaat te worden aangevraagd. Zodra het certificaat ontvangen in kan deze worden ingesteld in IIS. Als het certificaat eenmaal is ingesteld kan deze worden toegewezen aan één of meerdere webapplicaties. In dit geval moet hij worden ingesteld voor de beveiligingsapplicatie en de virtuele map met de functies.

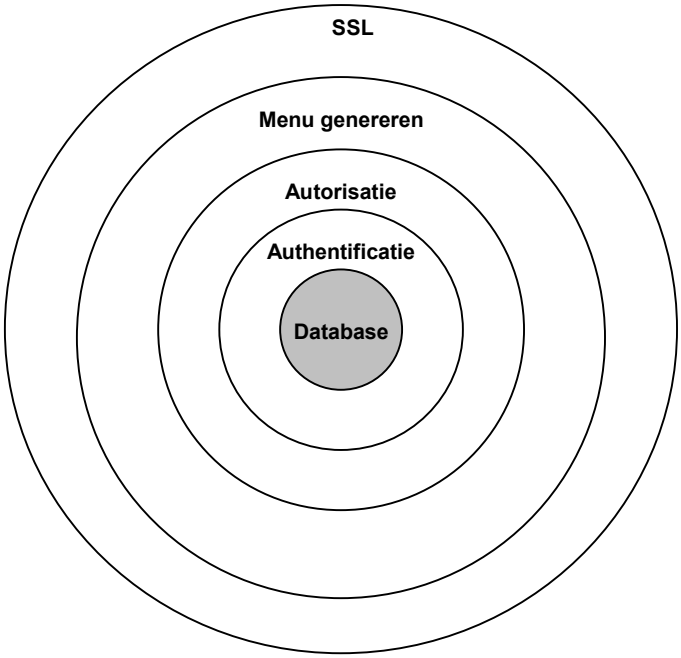
4. Pilotontwikkelplan

4.1 Inleiding

In dit onderdeel van het document wordt er een pilot ontwikkelplan gerealiseerd. Deze wordt ontworpen op basis van de gemaakte globale functionele en technische specificaties. Het pilotplan beschrijft hoe de pilot gaat worden gebouwd. De pilot dient te worden opgedeeld in pilotdelen en bouweenheden die achtereenvolgens dan wel parallel kunnen worden ontwikkeld. Tevens worden er voorzieningen getroffen voor het integreren en testen van de pilotdelen.

4.2 Pilotdelen

De pilot wordt opgedeeld in samenhangende componenten die afzonderlijk kunnen worden ontwikkeld, beoordeeld en getest. Er zal worden begonnen met het ontwikkelen van de database. Dit is kern van de applicatie. Hierna kan worden gestart met het ontwikkelen van de authentificatie en autorisatie mechanismen. Indien deze juist functioneren is het mogelijk om het menu te genereren.



Het menu zal worden gegenereerd aan de hand van de gebruikersinformatie die is opgedaan tijdens de authenticatie.

4.2.1 Database opzetten

In de database worden de gegevens geplaatst. Dit geeft de mogelijkheid aan de webapplicatie om gegevens te kunnen opvragen en muteren. Als de database is opgezet, is het mogelijk om tijdelijke testgegevens te plaatsen in de database waarmee later kan worden getest. De database zal worden opgezet zoals is aangegeven in hoofdstuk 3.3.
Nadat de database is opgezet is het mogelijk de webservice te gaan ontwikkelen. Voordat de webapplicatie de gegevens uit de database kan ophalen, zal er een koppeling moeten worden opgezet. Deze zal moeten worden ontwikkeld voordat een gebruiker zich kan authentificeren. De gegevens van een gebruiker, waaronder de gebruikersnaam en het wachtwoord, zijn opgeslagen in een aparte database.

4.2.2 Authentificatie

De authentificatie dient de gebruiker te authentificeren voordat hij wordt toegelaten in het systeem. Dit gebeurd op grond van een inlognaam en een wachtwoord. Een gebruiker voert zijn gebruikersnaam in met het bijbehorende wachtwoord. Het authentificatie deel dient het wachtwoord van de gebruiker te valideren. De authentificatie procedure zal het wachtwoord eerst moeten coderen om het te kunnen vergelijken met het wachtwoord in de database. Als een gebruiker zijn juiste inloggegevens heeft ingevoerd dient hij te worden doorgestuurd naar de hoofdpagina (default.aspx).
Eenmaal aangekomen bij de hoofdpagina dienen de autorisatie gegevens van de gebruiker te worden geladen in een sessie object (ADO dataset). Deze gegevens zullen tijdens de autorisatie en het genereren van het menu nodig zijn.

4.2.3 Autorisatie

Nadat een gebruiker is ingelogd heeft deze de bevoegdheid om alle standaard pagina's op te vragen binnen de virtuele map (Pagina's behorend bij de webapplicatie). IIS regelt zelf dat de bestanden zoals configuratie files en klasse bestanden van ASP.NET niet kunnen worden ingezien of worden gedownload.
Als een gebruiker eenmaal is ingelogd moet bij het aanvragen van een pagina met een functie wel worden gecontroleerd of de gebruiker de bevoegdheid heeft om deze op te vragen.
Om deze reden wordt er gebruik gemaakt van een http module. Deze module wordt in werking gesteld bij elke pagina die wordt opgevraagd uit de betreffende virtuele map. Bij het opvragen van deze pagina zal dan worden gecontroleerd of de gebruiker de bevoegdheid toegewezen heeft gekregen om deze pagina op te vragen en of de pagina geblokkeerd is of niet.

4.2.4 Menu genereren

Als een gebruiker is geauthentificeerd worden zijn autorisatie gegevens ingelezen in een dataset. Het menu dat gegenereerd zal worden zal worden gebaseerd op deze gegevens. Zo zullen de gebruiker zijn toegestane functiegroeppen worden gegeven en de mogelijkheid genereren om de daar onderliggende functies te kunnen gebruiken.

4.2.5 SSL

Als laatste moet de beveiligde verbinding worden opgezet. Dit is niet een onderdeel dat zelf ontwikkeld moet worden maar wel moet worden ingesteld tijdens deze pilot.

4.3 De bouweenheden

In dit deel van het document wordt elk pilotdeel opgesplitst in bouweenheden die afzonderlijk kunnen worden gebouwd. Ook zijn de stappen gegeven die moeten worden uitgevoerd voordat er kan worden gebouwd.

4.3.1 Database opzetten

Als eerste dient de database te worden opgezet omdat deze de kern vormt van de applicatie. Hierna dient een koppeling te worden ontwikkeld om het mogelijk te maken gegeven te muteren vanuit de webapplicatie. Om een koppeling te kunnen maken dient er gebruik te worden gemaakt van impersonation (zie paragraaf 2.5.3).

1. Implementeren databasemodel;
2. Impersonation configureren.
3. Koppeling opzetten tussen de database en de webapplicatie;

4.3.2 Aanpassingen externe systemen

Alle genoemde aanpassingen aan de externe systemen vinden niet allen plaats in deze pilotfase. De veranderingen aan de webserver worden namelijk pas bij de invoering doorgevoerd. Wel moet hiermee rekening worden gehouden.
Omdat in de Adabas database de gegevens van de klanten worden opgeslagen moet er een koppeling worden gemaakt met de Adabas database. Er wordt een procedure geschreven in Natural (op het mainframe waar de Adabas server zich bevindt) die direct de gegevens kan opvragen uit de Adabas database. Een ASP.NET functie zal deze functie aanroepen door middel van de brokerservice en de ontvangen gegeven verwerken in de SQL database. Dit zal een dagelijks of wekelijks synchronisatie proces zijn.

Op de domein server zal er een account worden aangemaakt met rechten voor toegang tot de SQL server en alle rechten tot de betreffende Database.

1. Procedure in Natural voor gegevens synchronisatie.
2. Procedure in ASP.NET voor het aanvragen en wegschrijven van de ontvangen klanten.

4.3.3 Authentificatie

Een gebruiker moet geïdentificeerd worden en geauthentificeerd voor toegang tot de webapplicatie. Dit zal worden gedaan door de gebruiker te koppelen aan een gebruikersnaam en daaraan een wachtwoord te koppelen. Tijdens de authentificatie moet het systeem controleren of de gebruikersnaam bestaat en of het ingetoetste wachtwoord klopt. Het wachtwoord in de database zal worden opgeslagen met behulp van SHA1. Dit is een 1-way encryptie methode. In de database zal het wachtwoord ge-encrypt worden opgeslagen zodat derden nooit het wachtwoord kunnen uitlezen.
Tijdens de authentificatie wordt er ook gecontroleerd of er geen gebruiker blokkade is geplaatst op systeem niveau of dat er een gebruikersblokkade is geplaatst op het niveau van klant of gebruiker die voor hem relevant is.
Als laatste wordt er een deel autorisatie meegenomen. De autorisatie gegevens van de gebruiker worden vast gelegd en aan de hand van deze gegevens kan zo het menu worden gegenereerd en kan er een controle plaatsvinden bij het opvragen van een pagina met een functie.

1. Controleren gebruikersnaam en wachtwoord;
2. Wachtwoord encryptie;
3. Controleren systeem-, klant- of gebruikersblokkade voor de gebruiker;
4. Inlezen autorisatie gegevens.

4.3.4 http Module

Met behulp van de http module wordt het andere deel van de autorisatie afgehandeld. De http module dient ervoor te zorgen dat nadat een gebruiker is ingelogd een gebruiker niet zomaar een willekeurige pagina kan opvragen. De module moet per opgevraagde pagina controleren of de gebruiker bevoegd is om de pagina (in de ogen van de SMRA, functionaliteit) te mogen gebruiken.

1. Controle of de gebruiker die de pagina opvraagt deel neemt aan een sessie;
2. Controle autorisatie bij het opvragen van een pagina gekoppeld aan een functie.

4.3.5 Hoofdpagina

Als de hoofdpagina wordt getoond moet deze aangepast zijn aan de gebruiker zijn beschikbare producten, functies en de blokkades die voor deze gebruiker relevant zijn.

1. Genereren persoonlijk productenmenu;
2. Onderscheid aangeven tussen beschikbare en geblokkeerde producten;
3. Genereren persoonlijk functiemenu;
4. Onderscheid aangeven tussen beschikbare en geblokkeerde producten.

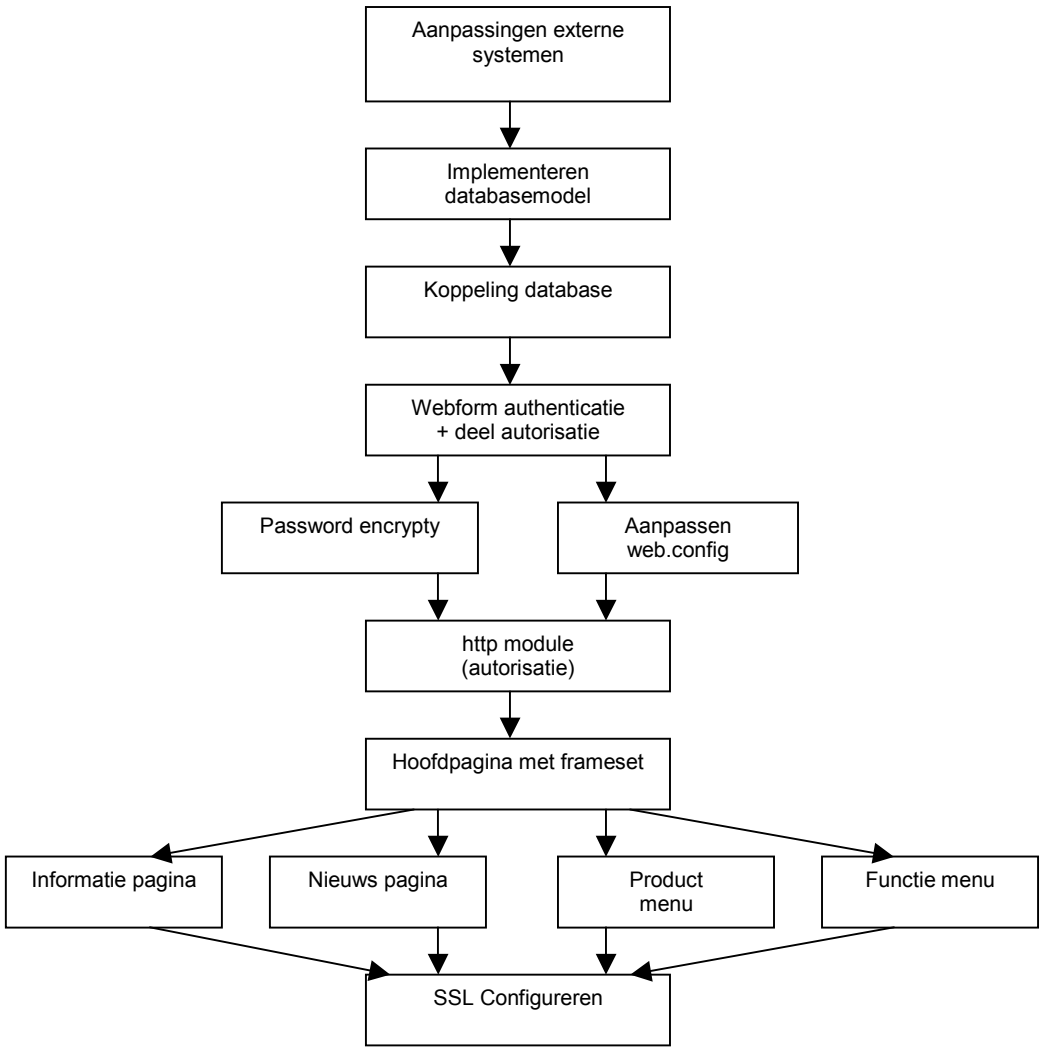
4.3.6 Blokkade afhandeling

Als een gebruiker een geblokkeerd product of een geblokkeerde functie kiest dient te worden aangegeven door wie de blokkade is geplaatst.

1. Webpagina voor het opvangen van gekozen blokkades.

4.4 Synchronisatie bouweenheden

In dit model wordt duidelijk in welke volgorde, per bouweenheid, de eerste pilot van het beveiligingssysteem ontwikkeld zal gaan worden.

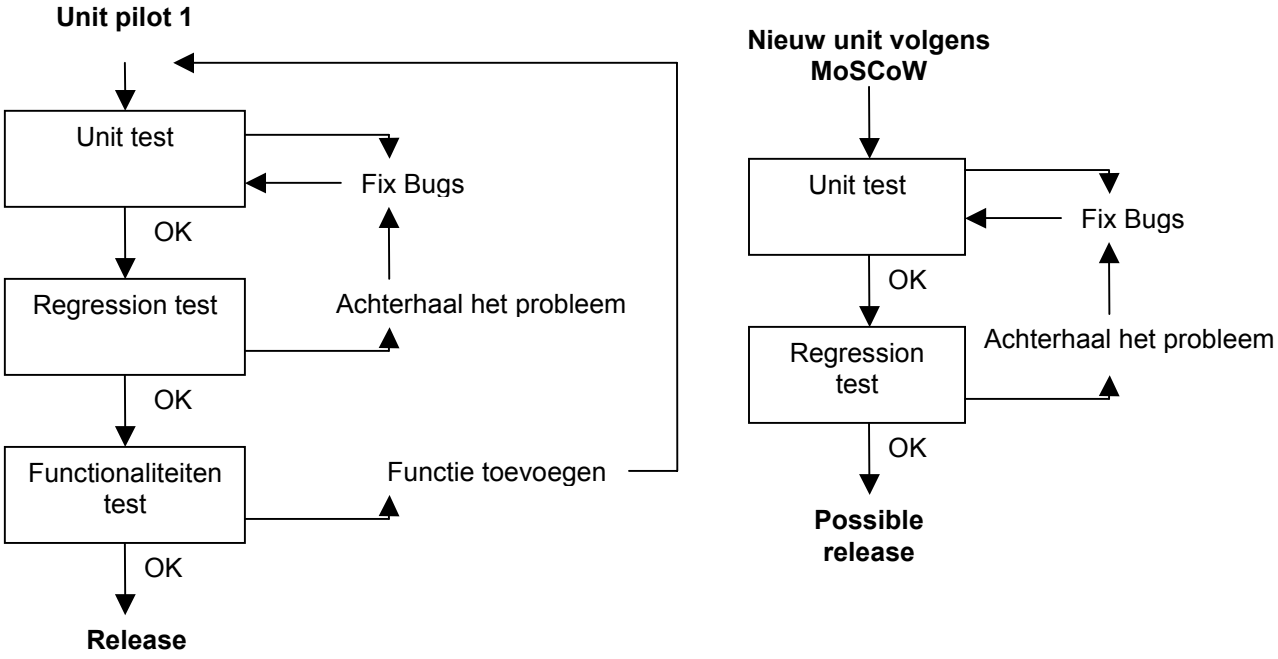


4.5 Voorbereiden beoordelen en testen pilotdelen

4.5.1 Het testen

Voor het testen van de webapplicatie zal er een unittest per unit plaatsvinden, een regressietest en een functionaliteitentest worden gedaan. Bij de Unit test worden alle aparte units (functionaliteiten in de software) apart getest.
Na het samenvoegen van de units wordt er een regression test gedaan. Mochten er fouten blijken te zitten in de applicatie tijdens het uitvoeren van de regression test, worden deze problemen achterhaald en worden de nodige aanpassingen gedaan om het probleem op te lossen.
Hierna zal er worden getest op functionaliteiten. In deze test wordt er simpelweg gekeken of de betreffende functionaliteiten uit de definitiestudie in het pilotdeel zijn verwerkt. Afhankelijk van de MoSCoW prioriteit wordt dit bepaald.

Tijdens pilot 3 zullen er extra functionaliteiten worden toegevoegd en zal elke nieuwe unit afzonderlijk worden getest en zal er gebruikt worden gemaakt van een aangepaste of aangevulde regression test. Hieronder is een schematische weergave gegeven van het verloop van het testen.



5. Ontwerp software bouweenheden

5.1 Inleiding

In dit hoofdstuk wordt de technische structuur van de software-bouweenheden die deel uitmaken van de pilot besproken. In dit onderdeel worden de detailprocesspecificaties en de testspecificaties besproken.

5.2 detailprocesspecificaties

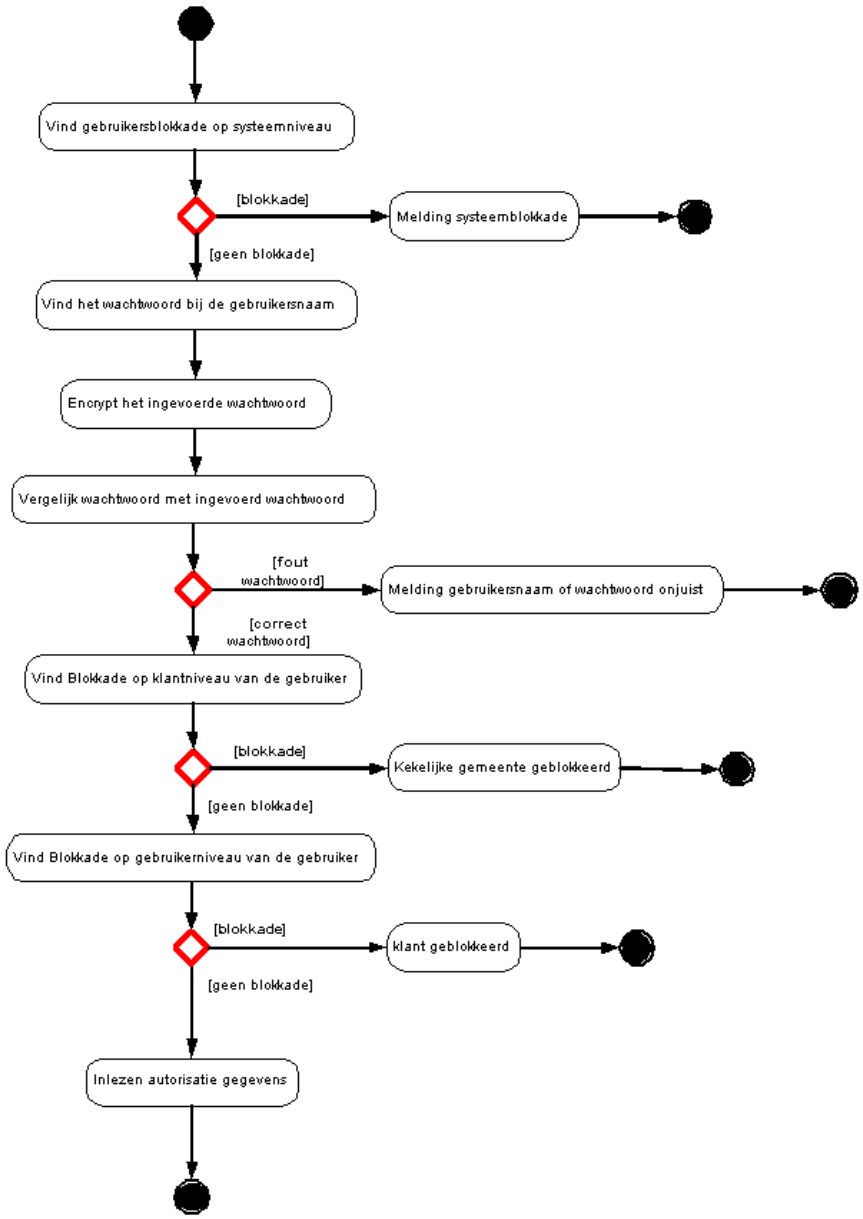
5.2.1 Aanpassingen externe systemen

Deze functionaliteit is uitgesteld omdat deze procedure aanpassingen vereist op het mainframe. Deze aanpassingen worden verricht door een specifieke medewerker van de SMRA en worden niet uitgevoerd door de student. Zowel de interface als de vorm van communicatie moet nog worden vastgelegd tijdens een overleg met een SMRA medewerker.

1. Procedure in Natural voor gegevens synchronisatie.
2. Procedure in ASP.NET voor het aanvragen en wegschrijven van de ontvangen klantgegevens.

5.2.2 Authentificatie

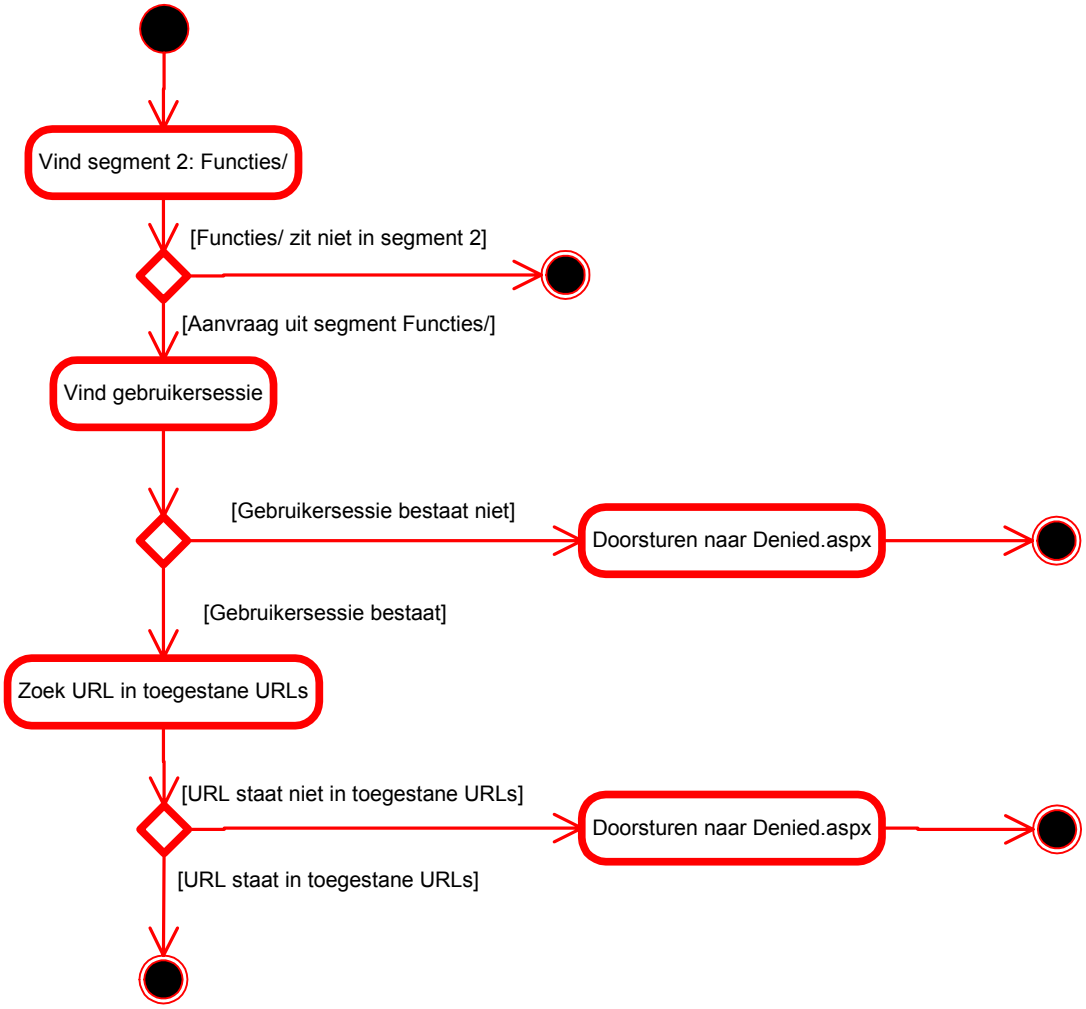
Hieronder is het activiteitendiagram gegeven van het authentificatie proces.



Deze procedure controleert of er geen relevante blokkades zijn voor de gebruiker die inlogd. Ook zorgt deze procedure ervoor dat een gebruiker alleen mag inloggen op het systeem als hij zijn juiste gebruikersnaam met het bijbehorende wachtwoord heeft ingevuld. Indien er een relevante blokkadeaanwezig is, een onbestaande gebruikersnaam is ingevoerd of een verkeerd wachtwoord is ingevoerd wordt hierover melding gegeven.
Als er geen relevante blokkades zijn en de gebruiker heeft zijn gebruikersnaam en wachtwoord correct opgegeven, worden de autorisatie gegevens ingelezen. Deze gegevens worden op een later tijdstip door de autorisatie module en bij het genereren van de menu's gebruikt.

5.2.3 http Module (autorisatie module)

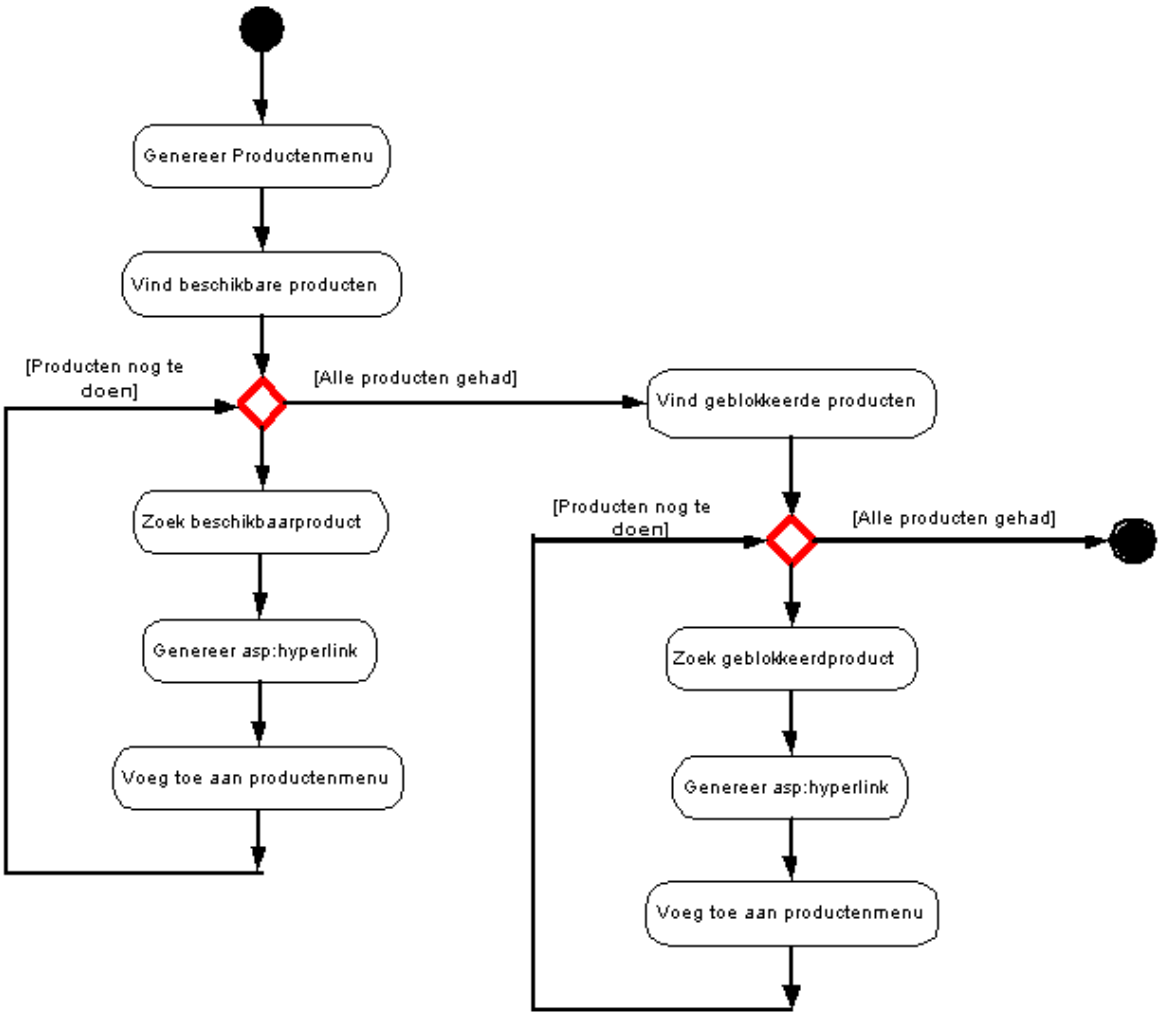
Hieronder is een activiteitendiagram gegeven voor een visuele beschrijving van het proces binnen de http module.



De autorisatie module controleert eerst of er een pagina wordt opgevraagd uit het 2e segment “Functies/”. Indien dit het geval is moet er gecontroleerd worden of de gebruiker is aangemeld bij een sessie. Als dit niet het geval is wordt de gebruiker doorgestuurd naar de webpagina: /Public/Denied.aspx.
Als een gebruiker deelneemt aan een sessie en een pagina opvraagt uit het 2^{e} segment Functies/ moet worden gecontroleerd of de gebruiker toegang heeft tot het opvragen van de functie. Dit wordt gedaan door een zoekopdracht uit te voeren in de toegestane Urls. Als het url wordt gevonden wordt de opgevraagde functie weergegeven.

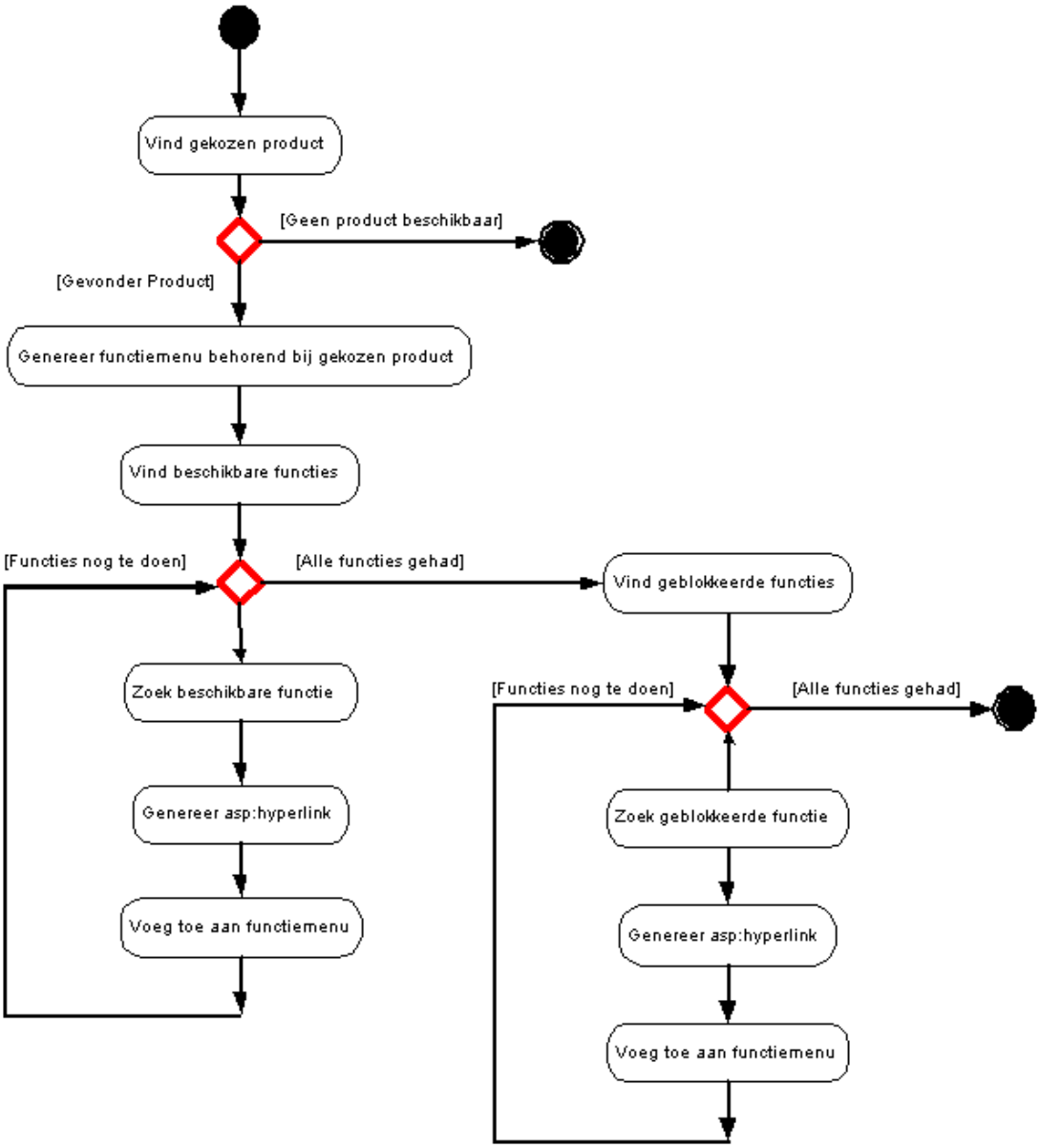
5.2.4 Hoofdpagina

Hieronder is een activiteitendiagram gegeven met daarin zichtbaar het proces voor het genereren van het persoonlijke productenmenu. Het menu genereert een lijst met hyperlinks met productkeuzen. De geblokkeerde producten worden apart aangegeven en de link wordt geplaatst naar de pagina die de blokkades afhandelt.



Het hieruit voortgekomen menu dient ervoor te zorgen dat als er een product wordt gekozen dat het gekozen product bekend wordt gemaakt bij het functiemenu. Als er een geblokkeerd product wordt gekozen wordt er een melding hierover gegeven in de blokkade afhandeling.

Door het kiezen van een product wordt het proces gestart om het functie menu te genereren. Het activiteitendiagram van dit proces is op de volgende pagina weergegeven.



Ook in dit proces wordt er onderscheid gemaakt tussen geblokkeerde en beschikbare functies. Als er een functie wordt gekozen wordt de functie geopend in het scherm. Als een geblokkeerde functie wordt opgevraagd wordt de aanvraag doorgestuurd naar de pagina voor de blokkade afhandeling.

5.2.5 Blokkade afhandeling

Hieronder is het proces van de blokkade afhandeling schematisch weergegeven.

Als er een geblokkeerd product of een geblokkeerde functie wordt opgevraagd treed de blokkade afhandeling in werking. Deze afhandelingprocedure zoekt het gebruikertype van de gebruiker die de blokkade heeft geplaatst. Uiteindelijk wordt er een bericht gegenereerd die doorgeeft aan de gebruiker dat de opgevraagde functie of het opgevraagde product is geblokkeerd en wordt hierbij het gebruikerstype van de blokkeerder gegeven. Dit kan bijvoorbeeld zijn: SMRA beheerder of gemeentelijke beheerder.

6. Testspecificaties

6.1 Inleiding

In dit onderdeel van het document worden de testspecificaties gegeven zoals deze in het pilotontwikkelplan zijn aangegeven.

6.2 Unittest testsets

6.2.1 Authentificatie units

Unit: CheckPassword(txtUserName.Text, txtPassword.Text)
Deze procedure leest eerst het wachtwoord van de bijbehorende gebruikersnaam. Hierna wordt het door de gebruiker ingevoerde wachtwoord gecodeerd naar SHA1 en vergeleken met het wachtwoord uit de database. Ook wordt tijdens deze procedure de gebruiker_Id naar de Session("Gebruiker_Id") en de Klant_Id naar de Session("Klant_Id") gelezen.
Voor het testen van deze unit dient er een gebruiker te worden aangemaakt in de database met de gebruikersnaam: testgebruiker en het wachtwoord: test. Let op, het wachtwoord: test moet in de database worden opgeslagen als SHA1!

Invoer	**Verwachte uitvoer**
Gebruikersnaam: testgebruiker, wachtwoord: test	True
Gebruikersnaam: testgebruiker, wachtwoord:	False
Gebruikersnaam: testgebruiker, wachtwoord: test2	False
Gebruikersnaam: GeenBekendeGebruiker, wachtwoord: test	False
Gebruikersnaam: Wachtwoord:	False
Gebruikersnaam: Wachtwoord: test	False

Unit: BlokSysteemNiv()
Deze unit controleert het of er een blokkade is met het type gebruiker en niveau systeem. Deze unit geeft en boolean terug. Deze functie werkt met gegevens uit de database uit de tabel blokkades. Hieronder zijn gegevens gegeven die zullen worden gebruikt om de werking van deze unit te testen.

Testgegevens in tabel Blokkades:

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id	Verwacht resultaat
Auto	3	2		1			1	False
Auto	2	3					1	False
Auto	3	3					1	True
Auto	3	1	1				1	False

Unit: BlokKlantNiv()
Deze unit controleert het of er een blokkade is met het type gebruiker en niveau klant. Deze unit geeft en boolean terug. Er dient een gebruiker te worden opgegeven met klant_Id: 1.

Testgegevens in tabel Blokkades:

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id	Verwacht resultaat
Auto	3	2		1			1	True
Auto	3	2		2			1	False
Auto	2	3					1	False
Auto	3	3					1	False
Auto	3	1	1				1	False

Unit: BlokGebruikerNiv()
Deze unit controleert het of er een blokkade is met het type gebruiker en gebruikerniveau. Deze unit geeft en boolean terug. Er dient een gebruiker te worden opgegeven met Gebruiker_Id: 1.

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id	Verwacht resultaat
Auto	3	2		1			1	False
Auto	2	3					1	False
Auto	3	1	1				1	True
Auto	3	3					1	False
Auto	3	1	2				1	False

6.2.2 Globale testset

Voor het testen van de ingelezen gegevens in de datasets wordt gebruikt gemaakt van de volgende testgegevens:

Testgegevens tabel Producten:

Product_ID	Product_Code	Naam
1	12	Product1
2	13	Product2
3	14	Product3

Testgegevens tabel Functies:

Functie_ID	Functie_Code	Naam	Start_url_Id	Product_Id
1	1	Functie1 product1	1	1
2	2	Functie2 product1	2	1
3	3	Functie3 product1	3	1
4	4	Functie1 product2	4	2
5	5	Functie2 product2	5	2
6	6	Functie3 product2	6	2
7	7	Functie1 product3	7	3
8	8	Functie2 product3	8	3

Testgegevens in GebruikerFuncties:

ID	Gebruiker_Id	Functie_Id
Auto	10	1
Auto	10	2
Auto	10	3
Auto	10	4
Auto	6	5
Auto	10	6
Auto	5	7
Auto	5	8

Testgegevens in blokkades:

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id
Auto	1	1	10		6		1
Auto	2	2		4		1	1
Auto	1	1	6		4		1

Testgegevens in Urls:

Url_Id	Functie_Id	Adres
1	1	Functie1.aspx
2	2	Functie2.aspx
3	3	Functie3.aspx
4	4	Functie4.aspx
5	5	Functie5.aspx
6	6	Functie6.aspx
7	7	Functie7.aspx
8	8	Functie8.aspx

Om te kunnen controleren of de datasets de juiste gegevens bevatten dienen er webpagina's te worden aangemaakt die de gegevens uit een dataset presenteren in een datagrid. Ook dient er een gebruiker te worden aangemaakt met gebruiker_Id 10 en klant_Id 4.

6.2.3 Authentificatie units inlezen datasets

Unit: LeesProductgroepen()
Deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Deze unit leest de productnaam en het Product_Id uit de tabel producten behorend bij de functies waarvan de gebruiker gebruik mag maken, in een dataset in.

Verwachte uitvoer: dstProducten

Product_Id	Naam
2	Product2

Unit: LeesBlokProductgroepen()
Ook deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Deze unit leest de productnaam en het Product_Id uit de tabel producten behorend bij de functies waarvan de gebruiker gebruik mag maken en die zijn geblokkeerd, in een dataset in. Ook wordt de blokkeerder_Id ingelezen vanuit de tabel blokkades.

Verwachte uitvoer: dstBlokProducten

Product_Id	Naam	Blokkeerder_Id
1	Product1	1

Unit: LeesFuncties()
Ook deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Deze unit leest de functie_Id, naam en het adres van de start_url van de toegestane functies.

Verwachte uitvoer: dstFuncties

Functie_Id	Naam	Adres
4	Functie1 product2	Functie4.aspx
5	Functie2 product2	Functie5.aspx

Unit: LeesBlokFuncties()
Ook deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Deze unit leest de functie_Id en de naam in van de geblokkeerde functies.

Verwachte uitvoer: dstBlokFuncties

Functie_Id	Naam
1	Functie1 product1
2	Functie2 product1
3	Functie3 product1
6	Functie1 product2

Unit: LeesUrls()
Ook deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Deze unit leest de adressen in van de functies die een gebruiker mag gebruiken.

Adres
Functie1.aspx
Functie2.aspx
Functie3.aspx
Functie4.aspx

6.2.4 Autorisatie

De autorisatiemodule is tijdens het opbouwen getest omdat deze niet bestaat uit losse units. De werking van deze module wordt getest tijdens de regression test. Dit is besloten omdat het apart testen van de units van de autorisatiemodule veel tijd in beslag zou nemen.

6.2.5 Productenmenu generatie

Ook deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Als de juiste testgegevens zijn ingevoerd zou de volgende HTML code voor de opbouw van het menu in de pagina moeten verschijnen:

“HTML opmaak”

Producten:

```
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=Header
ahref=Header.aspx?Product_Id=2>Product2</asp>
</tr>
</table>
```

Geblokkeerd:

```
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main
ahref=Denied.aspx?Product_Id=2&Blokkeerder_Id=1>Product1</asp>
</tr>
</table>
```

“HTML opmaak”

6.2.7 Functiemenu generatie

Ook deze unittest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Er van uitgaande dat product 2 zojuist uit het product menu is gekozen

“HTML opmaak”

Functies:

```
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Functie4.aspx>Functie1
Product2</asp>
</tr>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Functie5.aspx>Functie2
Product2</asp>
</tr>
</table>
```

Geblokkeerd:

```
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main
ahref=Denied.aspx?Product_Id=2&Blokkeerder_Id=1>Product1</asp>
```

</tr>
</table>

"HTML opmaak"

6.3 Regressiontest testset

Deze test vindt plaats als de software units zijn samengevoegd of gewijzigd. Deze test vindt ook plaats als er nieuwe functionaliteiten aan het systeem worden toegevoegd op grond van enkele units. Bij het toevoegen van functionaliteiten in wordt de regressie test simpelweg uitgebreid of worden er onderdelen toegevoegd.

6.3.1 Authentificatie test (SignOn)

Deze procedure controleert roept de onderliggende procedures en functies aan en bepaald of de gebruiker wordt doorgestuurd naar de beveiligde webpagina of bepaald het type melding dat wordt weergegeven aan de gebruiker.

Deze procedure maakt gebruik van onderliggende units:

- Unit: CheckPassword(txtUserName.Text, txtPassword.Text)
- Unit: BlokSysteemNiv()
- Unit: BlokKlantNiv()
- Unit: BlokGebruikerNiv()

Bij het uitvoeren van de units controleert de SignOn de blokkades en leest, indien van toepassing, de blokkades en toegewezen functies in. Ook zorgt de SignOn ervoor dat de gebruiker de bijbehorende melding ontvangt als hij niet toegelaten is tot het systeem.

Bij het testen van de authentificatie dient er een gebruiker te worden aangemaakt met de gebruikersnaam: testgebruiker en het wachtwoord: test. Let op, ook hier moet het wachtwoord in de database worden opgeslagen als SHA1!

Invoer	**Verwachte uitvoer**
Gebruikersnaam: testgebruiker, wachtwoord: test	(gebruiker wordt doorgestuurd naar default.aspx)
Gebruikersnaam: testgebruiker, wachtwoord:	Vul uw wachtwoord in.
Gebruikersnaam: testgebruiker, wachtwoord: test2	Uw gebruikersnaam of wachtwoord is onjuist.
Gebruikersnaam: GeenBekendeGebruiker, wachtwoord: test	Uw gebruikersnaam of wachtwoord is onjuist.
Gebruikersnaam: Wachtwoord:	Vul uw gebruikersnaam in. Vul uw wachtwoord in.
Gebruikersnaam: Wachtwoord: test	Vul uw gebruikersnaam in.
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn gebruikersnaam.	Uw gebruikersnaam is geblokkeerd door (gebruikersnaam van de blokkeerder)
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn klant_Id.	U bent als klant geblokkeerd door (gebruikersnaam van de blokkeerder).
Gebruikers en wachtwoord van een gebruiker. Er moet een gebruikersblokkade worden geplaatst voor alle gebruikers.	Alle gebruikers zijn op dit moment geblokkeerd door (gebruikersnaam van de blokkeerder).
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn gebruikersnaam. Er moet ook een gebruikersblokkade worden geplaatst voor alle gebruikers.	Alle gebruikers zijn op dit moment geblokkeerd door (gebruikersnaam van de blokkeerder).
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn klant_Id. Er moet ook een gebruikersblokkade worden geplaatst voor alle gebruikers.	Alle gebruikers zijn op dit moment geblokkeerd door (gebruikersnaam van de blokkeerder).
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn gebruikersnaam en op zijn klant_Id.	U bent als klant geblokkeerd door (gebruikersnaam van de blokkeerder).

De blokkadeniveau's moeten zichtbaar blijven als een gebruiker geblokkeerd is.

6.3.2 Autorisatie test

Ook dit deel van de regressiontest maakt gebruik van de testgegevens uit paragraaf 6.2.2.. Deze module controleert of de gebruiker een pagina opvraagt met als 2e segment “Functies/”. Tevens wordt er ook gecontroleerd of de gebruiker deelneemt aan een sessie en of de gebruiker bevoegd is om de pagina op te vragen. Omdat dit een gehele module is binnen de ASP.NET webapplicatie kunnen de onderlinge functies niet specifiek worden getest. Om deze reden wordt de werking getest door handmatige controles uit te voeren. Om te kunnen testen of deze autorisatie module correct werkt dienen de volgende handelingen te worden verricht.

Deze test is sterk afhankelijk van de domeinnaam waarin de test wordt uitgevoerd. In dit geval wordt deze test uitgevoerd op de locale machine. Het domein is in dit geval http://localhost .

Log in met de gebruikersnaam en het wachtwoord van de gebruiker met gebruiker_Id: 10. Probeer het url: domein/Publid/Denied.aspx op te vragen.

Url	**Omschrijving**	**Verwacht:**
Domein/Public/Denied.aspx	Publieke pagina, mag altijd worden opgevraagd.	Toegang
Domein/default.aspx	Pagina mag worden opgevraagd door een gebruiker die deelneemt aan een sessie.	Toegang, indien ingelogd.
Domein/Functies/Functie6.aspx	Deze aanvraag bevindt zich in het 2e segment Funcies/. Het adres Functie6.aspx staat niet in de toegestane urls!	**Geen toegang**
Domein/Functies/Functie1.aspx	Deze aanvraag bevindt zich in het 2e segment Funcies/. Het adres Functie6.aspx staat wel in de toegestane urls!	**Toegang**

6.3.3 Menu generatie test

Ook dit deel van de regressiontest maakt gebruik van de testgegevens uit paragraaf 6.2.2. Als de juiste testgegevens zijn ingevoerd zou het volgende menu moeten verschijnen:

Producten:
Product2

Geblokkeerd:
Product1

Ook de gegenereerde links moeten worden gecontroleerd. Als er een beschikbaar product wordt gekozen dan dienen de onderliggende functies te verschijnen in het functiemenu. Als er een geblokkeerd product wordt gekozen dan dient in het main scherm te worden aangegeven door wie het product is geblokkeerd. In dit geval zou het functiemenu bestaat uit:

Functies:
Functie1 Product 2
Functie2 Product 2

Geblokkeerd:
Functie 3 Product2

Ook hier geld dat als er een geblokkeerde functie wordt gekozen dat er in het main scherm wordt weergegeven door wie deze functie is geblokkeerd. Als er een niet geblokkeerde functie wordt gekozen dan dient de functie te worden geopend in het main deel van de pagina.

Om dit te testen wordt er een testfunctie aangemaakt in de map Functies/ met de naam Functie4.aspx. Als nu product 2 wordt gekozen en de onderliggende functie: Functie1 Product2 dan dient de test functie te verschijnen in het main gedeelte van de pagina.

6.3.4 Sessie test

De instellingen die zijn opgegeven in de web.config dienen ook te worden getest. ZO dient er te worden getest of er een sessie wordt aangemaakt als een gebruiker zich aanmeldt op de pagina. Maar ook dient er te worden getest wanneer deze sessie vervalt en wanneer niet. Hieronder zijn een aantal handmatige handelingen gegeven waarmee dit getest kan worden.

Omschrijving	**Verwacht resultaat**
Een gebruiker vraagt de default.aspx pagina op.	De webapplicatie herkent een nieuwe gebruiker en wijst hem een unieke sessie toe. Het ID van de sessie wordt meegegeven in het adres.
Een gebruiker typt direct een adres in naar een pagina op de site na te zijn ingelogd.	De gebruiker krijgt toegang tot de pagina.
Een gebruiker sluit de webapplicatie, opent een nieuw navigatiescherm en typt het url in van een functie.	De gebruiker krijgt geen toegang tot de functie omdat de sessie gegevens zijn vervallen.
Een gebruiker vraagt na het afsluiten van de webapplicatie nogmaals het URL op en geeft daarbij de sessie ID mee in het URL.	Geen toegang.

6.3.5 Forms authentificatie test (Cookie)

Binnen de configuratie van de webapplicatie is er aangegeven dat er gebruik wordt gemaakt van een sessie zonder het bijhouden van een cookie (cookieless). Forms authentication maakt echter wel gebruik van een cookie. Deze test is er om te controleren hoe de webapplicatie omgaat met cookies.

Omschrijving	**Verwacht resultaat**
Een gebruiker vraagt de default.aspx pagina op.	Er wordt nog geen cookie toegewezen
Een gebruiker logt in en accepteert cookies.	Er wordt een authentificatie cookie aan de gebruiker toegewezen.
Een gebruiker logt in en accepteert geen cookies.	Er wordt geen cookie aangemaakt maar de gebruiker krijgt wel toegang tot de webpagina.
Een gebruiker heeft een cookie geaccepteerd en sluit de webapplicatie. Hierna opent hij de default.aspx pagina opnieuw na de cookie timeout (5 min).	De gebruiker krijgt geen toegang tot de default.aspx pagina. Hiervoor moet hij eerst inloggen.

6.4 Functionaliteitentest

De “must have” eisen en wensen die volgens de MoSCoW regels zijn vastgelegd tijdens de definitiestudie komen hier aan bod. In dit onderdeel wordt er gecontroleerd of de “must have” eisen en wensen behorende bij deze pilot, zijn verwerkt in deze pilot. Hieronder zijn de must have eisen en wensen gegeven die na afronding van deze pilot gerealiseerd zijn.

6.4.1 Algemene functionele eisen

F27	Een gemeentelijke beheerder of functionaris is altijd gekoppeld aan 1 kerkelijke gemeente (werkingsgebied)	**M**
F32	Elke gebruiker heeft een Pincode waarmee hij gevalideerd kan worden	**M**
F31	Van een gebruiker worden de bijbehorende persoonlijke gegevens opgeslagen	**M**
F33	Elke gebruiker heeft een status (geblokkeerd of niet geblokkeerd)	**M**
F36	Identificatie gebruikers voor alle applicatie onderdelen	**M**
F37	Authentificatie gebruikers voor alle applicatie onderdelen	**M**
F41	Elke gebruiker heeft zijn eigen unieke ID	**M**

6.4.2 Technische eisen

E2	Gebruik maken van een beveiligde verbinding (SSL of dergelijke)
E3	Alle gegevens worden opgeslagen in een aparte MS-SQL database
E4	Het systeem dient op te vragen te zijn over het internet
E5	De applicatie dient geschreven te worden gebruikt maken van ASP.net (VB)
E6	Het systeem dient op te vragen te zijn over het intranet
E7	Koppeling met de main-database voor het opvragen van gemeente
E10	Synchronisatie mogelijkheid met de main-database
E11	Het systeem moet voorbereidend zijn voor uitbreidingen

7. Testresultaten

7.1 Inleiding

In dit hoofdstuk zijn de resultaten gegeven van de uitgevoerde testen zoals die zijn beschreven in het voorgaande hoofdstuk.

7.2 Resultaten unit test

7.2.1 Authentificatie units

Unit: CheckPassword(txtUserName.Text, txtPassword.Text)

Invoer	**Verwachte uitvoer**	**Uitvoer:**
Gebruikersnaam: testgebruiker, wachtwoord: test	True	True
Gebruikersnaam: testgebruiker, wachtwoord:	False	False
Gebruikersnaam: testgebruiker, wachtwoord: test2	False	False
Gebruikersnaam: GeenBekendeGebruiker, wachtwoord: test	False	False
Gebruikersnaam: Wachtwoord:	False	False
Gebruikersnaam: Wachtwoord: test	False	False

Unit: BlokSysteemNiv()

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id	Verwacht resultaat	**Res.**
Auto	3	2		1			1	False	False
Auto	2	3					1	False	False
Auto	3	3					1	True	True
Auto	3	1	1				1	False	False

Unit: BlokKlantNiv()

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id	Verwacht resultaat	**Res.**
Auto	3	2		1			1	True	True
Auto	3	2		2			1	False	False
Auto	2	3					1	False	False
Auto	3	3					1	False	False
Auto	3	1	1				1	False	False

Unit: BlokGebruikerNiv()

Blokkade_Id	Type_Id	Niveau_Id	Gebruiker_Id	Klant_Id	Functie_Id	Product_Id	Blokkeerder_Id	Verwacht resultaat	**Res.**
Auto	3	2		1			1	False	False
Auto	2	3					1	False	False
Auto	3	1	1				1	True	True
Auto	3	3					1	False	False
Auto	3	1	2				1	False	False

7.2.2 Authentificatie units inlezen datasets

Unit: LeesProductgroepen()

Verwachte uitvoer: dstProducten

Product_Id	Naam
2	Product2

Uitvoer:dstProducten

Product_Id	Naam
2	Product2

Unit: LeesBlokProductgroepen()

Verwachte uitvoer: dstBlokProducten

Product_Id	Naam	Blokkeerder_Id
1	Product1	1

Uitvoer: dstBlokProducten

Product_Id	Naam	Blokkeerder_Id
1	Product1	1

Unit: LeesFuncties()

Verwachte uitvoer: dstFuncties

Functie_Id	Naam	Adres
4	Functie1 product2	Functie4.aspx
5	Functie2 product2	Functie5.aspx

Uitvoer: dstFuncties

Functie_Id	Naam	Adres
4	Functie1 product2	Functie4.aspx
5	Functie2 product2	Functie5.aspx

Unit: LeesBlokFuncties()

Verwachte uitvoer: dstBlokFuncties

Functie_Id	Naam
1	Functie1 product1
2	Functie2 product1
3	Functie3 product1
6	Functie1 product2

Uitvoer: dstBlokFuncties

Functie_Id	Naam
1	Functie1 product1
2	Functie2 product1
3	Functie3 product1
6	Functie1 product2

Unit: LeesUrls()

Verwachte uitvoer: dstUrls

Adres
Functie1.aspx
Functie2.aspx
Functie3.aspx
Functie4.aspx

Uitvoer: dstUrls

Adres
Functie1.aspx
Functie2.aspx
Functie3.aspx
Functie4.aspx

7.2.3 Productenmenu generatie

Verwacht:

"HTML opmaak"

Producten:

```
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=Header
ahref=Header.aspx?Product_Id=2>Product2</asp>
</tr>
</table>
```

Geblokkeerd:

```
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main
ahref=Denied.aspx?Product_Id=2&Blokkeerder_Id=1>Product1</asp>
</tr>
</table>
```

"HTML opmaak"

Resultaat:

“HTML opmaak”

Producten:
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=Header
ahref=Header.aspx?Product_Id=2>Product2</asp>
</tr>
</table>

Geblokkeerd:
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main
ahref=Denied.aspx?Product_Id=2&Blokkeerder_Id=1>Product1</asp>
</tr>
</table>

“HTML opmaak”

7.2.5 Functiemenu generatie

Verwacht:

“HTML opmaak”

Functies:
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Functie4.aspx>Functie1
Product2</asp>
</tr>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Functie5.aspx>Functie2
Product2</asp>
</tr>
</table>

Geblokkeerd:
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main
ahref=Denied.aspx?Product_Id=2&Blokkeerder_Id=1>Product1</asp>
</tr>
</table>

“HTML opmaak”

Resultaat:

"HTML opmaak"

Functies:
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Functie4.aspx>Functie1 Product2</asp>
</tr>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Functie5.aspx>Functie2 Product2</asp>
</tr>
</table>

Geblokkeerd:
<table width=100%>
<tr>
<asp:hyperlink id=Hyperlink1 runat=server target=main ahref=Denied.aspx?Product_Id=2&Blokkeerder_Id=1>Product1</asp>
</tr>
</table>

"HTML opmaak"

7.3 Regressiontest testset

7.3.1 Authentificatie test (SignOn)

Invoer	Verwachte uitvoer	Resultaat:
Gebruikersnaam: testgebruiker, wachtwoord: test	(gebruiker wordt doorgestuurd naar default.aspx)	√
Gebruikersnaam: testgebruiker, wachtwoord:	Vul uw wachtwoord in.	√
Gebruikersnaam: testgebruiker, wachtwoord: test2	Uw gebruikersnaam of wachtwoord is onjuist.	√
Gebruikersnaam: GeenBekendeGebruiker, wachtwoord: test	Uw gebruikersnaam of wachtwoord is onjuist.	√
Gebruikersnaam: Wachtwoord:	Vul uw gebruikersnaam in. Vul uw wachtwoord in.	√
Gebruikersnaam: Wachtwoord: test	Vul uw gebruikersnaam in.	√
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn gebruikersnaam.	Uw gebruikersnaam is geblokkeerd door (gebruikersnaam van de blokkeerder)	√
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn klant_Id.	U bent als klant geblokkeerd door (gebruikersnaam van de blokkeerder).	√
Gebruikers en wachtwoord van een gebruiker. Er moet een gebruikersblokkade worden geplaatst voor alle gebruikers.	Alle gebruikers zijn op dit moment geblokkeerd door (gebruikersnaam van de blokkeerder).	√
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn gebruikersnaam. Er moet ook een gebruikersblokkade worden geplaatst voor alle gebruikers.	Alle gebruikers zijn op dit moment geblokkeerd door (gebruikersnaam van de blokkeerder).	√
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn klant_Id. Er moet ook een gebruikersblokkade worden geplaatst voor alle gebruikers.	Alle gebruikers zijn op dit moment geblokkeerd door (gebruikersnaam van de blokkeerder).	√
Gebruikers en wachtwoord van een gebruiker met een gebruikersblokkade op zijn gebruikersnaam en op zijn klant_Id.	U bent als klant geblokkeerd door (gebruikersnaam van de blokkeerder).	√

7.3.2 Autorisatie test

Url	Omschrijving	Verwacht:	Resultaat:
Domein/Public/Denied.aspx	Publieke pagina, mag altijd worden opgevraagd.	Toegang	Toegang
Domein/default.aspx	Pagina mag worden opgevraagd door een gebruiker die deelneemt aan een sessie.	Toegang, indien ingelogd.	Toegang
Domein/Functies/Functie6.aspx	Deze aanvraag bevindt zich in het 2e segment Funcies/. Het adres Functie6.aspx staat niet in de toegestane urls!	**Geen toegang**	Geen toegang
Domein/Functies/Functie1.aspx	Deze aanvraag bevindt zich in het 2e segment Funcies/. Het adres Functie6.aspx staat wel in de toegestane urls!	**Toegang**	Toegang

7.3.3 Menu generatie test

Verwacht Productenmenu:

Producten:
Product2

Geblokkeerd:
Product1

Resultaat Productenmenu:

Producten:
Product2

Geblokkeerd:
Product1

Verwacht functiemenu

Functies:
Functie1 Product 2
Functie2 Product 2

Geblokkeerd:
Functie 3 Product2

Resultaat functiemenu:

Functie1 Product 2
Functie2 Product 2

Geblokkeerd:
Functie 3 Product2

7.3.4 Sessie test

De instellingen die zijn opgegeven in de web.config dienen ook te worden getest. ZO dient er te worden getest of er een sessie wordt aangemaakt als een gebruiker zich aanmeldt op de pagina. Maar ook dient er te worden getest wanneer deze sessie vervalt en wanneer niet. Hieronder zijn een aantal handmatige handelingen gegeven waarmee dit getest kan worden.

Omschrijving	**Verwacht resultaat**	**Resultaat**
Een gebruiker vraagt de default.aspx pagina op.	De webapplicatie herkent een nieuwe gebruiker en wijst hem een unieke sessie toe. Het ID van de sessie wordt meegegeven in het adres.	√
Een gebruiker typt direct een adres in naar een pagina op de site na te zijn ingelogd.	De gebruiker krijgt toegang tot de pagina.	√
Een gebruiker sluit de webapplicatie, opent een nieuw navigatiescherm en typt het url in van een functie.	De gebruiker krijgt geen toegang tot de functie omdat de sessie gegevens zijn vervallen.	√
Een gebruiker vraagt na het afsluiten van de webapplicatie nogmaals het URL op en geeft daarbij de sessie ID mee in het URL.	Geen toegang.	**Toegang of geen toegang, afhankelijk van garbage collection. Een sessie wordt niet altijd beeindigd nadat een gebruiker zijn scherm sluit! De garbage collection moet dit eert detecteren!**

7.3.5 Forms authentificatie test (Cookie)

Binnen de configuratie van de webapplicatie is er aangegeven dat er gebruik wordt gemaakt van een sessie zonder het bijhouden van een cookie (cookieless). Forms authentication maakt echter wel gebruik van een cookie. Deze test is er om te controleren hoe de webapplicatie omgaat met cookies.

Omschrijving	**Verwacht resultaat**	**Resultaat**
Een gebruiker vraagt de pagina: default.aspx op.	Er wordt nog geen cookie toegewezen	√
Een gebruiker logt in en accepteert cookies.	Er wordt een authentificatie cookie aan de gebruiker toegewezen.	√
Een gebruiker logt in en accepteert geen cookies.	Er wordt geen cookie aangemaakt maar de gebruiker krijgt wel toegang tot de webpagina.	√
Een gebruiker heeft een cookie geaccepteerd en sluit de webapplicatie. Hierna opent hij de default.aspx pagina opnieuw na de cookie timeout (5 min).	De gebruiker krijgt geen toegang tot de default.aspx pagina. Hiervoor moet hij eerst inloggen.	**De cookie verloopt niet! Tiemout anders ingesteld. Cookie verloopt nog steeds niet!**

7.4 functionaliteitentest

De “must have” eisen en wensen die volgens de MoSCoW regels zijn vastgelegd tijdens de definitiestudie komen hier aan bod. In dit onderdeel wordt er gecontroleerd of de “must have” eisen en wensen behorende bij deze pilot, zijn verwerkt in deze pilot. Hieronder zijn de must have eisen en wensen gegeven die na afronding van deze pilot gerealiseerd zijn.

7.4.1 Algemene functionele eisen

ID	Omschrijving	**MoSCoW**	**Verwerkt?**
F27	Een gemeentelijke beheerder of functionaris is altijd gekoppeld aan 1 kerkelijke gemeente (werkingsgebied)	**M**	**Ja.**
F32	Elke gebruiker heeft een Pincode waarmee hij gevalideerd kan worden	**M**	**Ja.**
F31	Van een gebruiker worden de bijbehorende persoonlijke gegevens opgeslagen	**M**	**Ja.**
F33	Elke gebruiker heeft een status (geblokkeerd of niet geblokkeerd)	**M**	**Opmerking 1**
F36	Identificatie gebruikers voor alle applicatie onderdelen	**M**	**Ja.**
F37	Authentificatie gebruikers voor alle applicatie onderdelen	**M**	**Ja.**
F41	Elke gebruiker heeft zijn eigen unieke ID	**M**	**Ja.**

Opmerking 1:
Een gebruiker heeft niet een directe status. Een gebruiker kan op meerdere niveau’s worden geblokkeerd. Dit is mogelijk gemaakt omdat door een blokkade mechanisme te ontwikkelen. Aan een gebruiker kan nog wel de status geblokkeerd worden toegekend.

7.4.2 Technische eisen

ID	Omschrijving	**Verwerkt?**
E2	Gebruik maken van een beveiligde verbinding (SSL of dergelijke)	**Ja.**
E3	Alle gegevens worden opgeslagen in een aparte MS-SQL database	**Ja.**
E4	Het systeem dient op te vragen te zijn over het internet	**Ja.**
E5	De applicatie dient geschreven te worden gebruikt maken van ASP.net (VB)	**Ja.**
E6	Het systeem dient op te vragen te zijn over het intranet	**Ja.**
E7	Koppeling met de main-database voor het opvragen van gemeente	**Nee.**
E10	Synchronisatie mogelijkheid met de main-database	**Ja.**
E11	Het systeem moet voorbereidend zijn voor uitbreidingen	**Ja.**

Eis: E7 is tijdelijk uitgesteld wegens een vereiste functionaliteit op het mainframe voor het ophalen van de gegevens. Het ontwikkelen van deze functie is zal worden gedaan door een medewerker van de SMRA. Voordat deze functionaliteit kan worden ontwikkeld dienen er eerst nog afspraken te worden gemaakt met een medewerker van de SMRA. Deze afspraken zullen voornamelijk betrekking hebben op de interface en de manier waarop wordt gecommuniceerd tussen de te ontwikkelen functionaliteiten.

8. Conclusie

In het eerste pilotdeel is de beveiligingsapplicatie ontworpen en ontwikkeld. Er is gestart met het functionele ontwerp. In het functionele ontwerp is de functionele inhoud van de pilot bepaald. Het systeemconcept dat is gegeven in de voorgaande definitiestudie is hier uitgebreid.
Op basis van het functionele ontwerp is het technische ontwerp gerealiseerd. In het technische ontwerp zijn de technische aspecten rond de pilotontwikkeling gegeven. Hierbij zijn besproken: de fysieke allocatie, het technischopslagmodel en de externe interfaces. Hierna is er een pilotontwikkelplan gemaakt waarin is omschreven hoe de pilot is opgebouwd. De pilot is hier opgedeeld in pilotdelen en de bouweenheden zijn hier bepaald. Als volgt zijn er ontwerpen gemaakt van de softwarebouweenheden en zijn er testspecificaties opgesteld. Na het bouwen van de softwarebouweenheden zijn deze getest met behulp van de unit testsets. Het uitvoeren van de unittest verliep vrijwel vlekkeloos. Uiteindelijke zijn de software-bouweenheden samengevoegd en heeft er een regressiontest plaats gevonden. De in de regressiontest aangetroffen “fouten” zijn verholpen maar bij sommige testen zijn er opmerkingen geplaatst.
Ook heeft er een functionaliteitentest plaatsgevonden om te controleren of alle genoemde onderdelen volgens de definitiestudie zijn verwerkt in de eerste pilot. Alle functionaliteiten zijn verwerkt behalve de koppeling met de main database en de functie voor het synchroniseren van de kerkelijke gemeenten.

Na het afronden van de eerste pilot is er een pilotworkshop gehouden waarin het product van de eerste pilot is gepresenteerd. De voornaamste belangen van deze workshop was het verkrijgen van feedback op het systeem en de gebruikers een beeld te geven of de mogelijkheden van het systeem.

Omdat er gewerkt wordt met de iteratiestrategie Big-Bang-Invoeren wordt na het afronden van deze pilot de definitiestudie herzien. In de definitie studie worden de nodige wijzigingen aangebracht en wordt er een evaluatie gegeven over deze pilot. Als de wijzigingen zijn aangebracht zal worden gestart aan de ontwikkeling van Pilot 2.

Ontwikkeling webbased beveiligingssysteem bij de SMRA

- **Document**
  Pilotontwikkeling: Pilot 2/3
- **Auteur**
  Mark van Houten (20015081)
- **Datum**
  26 juli 2004
- **Opleiding**
  Haagse Hogeschool Sector Informatica
  Johanna Westerdijkplein 75
  2521 Den Haag
- **Opleidingsvariant**
  MBIT
- **Periode**
  November 2003 – april 2004
- **Opdrachtgever**
  Roel Mehlkopf
  Kees Schaaf
  SMRA, Delft
- **Examinatoren**
  Breukel, P. R. C.
  Mimoun, S.N.

Voorwoord

Voor u ligt het rapport “pilotontwikkeling: pilot 2” ten behoeve van een Web based beveiligingssysteem met functie en gebruikersbeheer voor de SMRA. Dit rapport is tot stand gekomen in overleg met dhr R. Mehlkopf en dhr K. Schaaf.

De basis van dit rapport is de eerder opgeleverde definitiestudie. Het deel van het daarin ontworpen systeemconcept zal in dit document tot in detail worden uitgewerkt. Dit rapport vormt de basis voor het bouwen van het systeem.

Delft, SMRA, 26 juli 2004

Inhoudsopgave

1. Inleiding

Het doel van de pilotontwikkeling is het ontwerpen van een pilot. De pilot zal worden ontwikkeld aan de hand van de eisen die eerder zijn opgesteld in de definitiestudie.

Als eerste zal er een globale functiestructuur van de pilot worden gemaakt. Het systeemconcept zal hier worden verfijnd.

Hierna zal de technische functiestructuur globaal rond de pilotontwikkeling worden gerealiseerd. De fysieke allocatie, het technische opslagmodel, het netwerkcommunicatiemiddel en de externe interfaces worden hier beschreven.

Er zal een pilotontwikkelplan moeten worden opgesteld op basis van de gemaakte globale functionele en technische specificaties. Het pilotontwikkelplan beschrijft hoe de pilot gaat worden gebouwd.

Op basis van bovenstaande onderdelen is het mogelijk om een ontwerp te maken van de software bouweenheden. De componenten zullen worden afgebeeld op de software architectuur. De detailgegevenspecificaties, detailprocesspecificaties en de testspecificaties worden besproken.

Nadat het ontwerp is gemaakt van de software bouweenheden is het mogelijk de bouweenheden daadwerkelijk te gaan bouwen. De fysieke database zal worden gerealiseerd, de gebruikersinterfaces zullen worden gebouwd en de gebouwde bouweenheden zullen worden getest.

Mochten er externe componenten of informatiesystemen (herbruikbare componenten of informatiesystemen) worden gebruikt, dan zullen deze wellicht moeten worden aangepast. Voor elk te gebruiken externe component of informatiesysteem zal dan ook een ontwerp met aanpassingen worden gemaakt, die zullen worden geïmplementeerd en getest.

Nadat alle bouweenheden zijn gebouwd en getest, kunnen deze worden geïntegreerd en kan worden gecontroleerd of de bouweenheden op de juiste manier met elkaar communiceren.

Uiteindelijk vindt het beoordelen van de pilotontwikkeling plaats. Er wordt een gezamenlijke test gedaan en hierover wordt een oordeel gegeven.

2. Globale functiestructuur

2.1 Inleiding

In dit deel van het document wordt de functionele inhoud van de pilot bepaald. In de definitiestudie is de functionele architectuur besproken (in het systeemconcept) die hier zal worden verfijnd. In dit onderdeel wordt gegeven: het functionele procesmodel en prototype van de gebruikers interface.

Er worden prototypen gegeven van de gebruikersinterface zodat de opdrachtgever een indruk krijgt over de gebruikersinterface van het te ontwikkelen systeem. Dit geeft de opdrachtgever de mogelijkheid om zijn argumenten te laten verwerken in de werkelijke applicatie. Ook neemt het vaak twijfels bij de opdrachtgever weg over de functionaliteiten en de structuur van de applicatie.

In het conceptueel gegevensmodel worden de entiteiten en hun relaties gedetailleerder beschreven, inclusief de gegevenselementen die ze bevatten (attributen) en de toegestane toestanden van de entiteiten.

Als laatste deel van dit onderdeel wordt er gekeken naar eventuele functionele aanpassingen van externe systemen voor de integratie met het te ontwikkelen systeem.

2.2 Gebruikersbeheer

Voor de SMRA beheerders moet het mogelijk zijn gemeentelijke beheerders aan te maken voor het systeem. Ook moeten deze gebruikers beheerd kunnen worden. Met beheren wordt in dit geval bedoeld gebruikers: opvragen, aanmaken, wijzigen en verwijderen. Gemeentelijke beheerders maken ook gebruik van gebruikersbeheer. Alleen mogen zij alleen onderliggende functionarissen beheren.

Hiernaast zijn ook nog andere beheersmogelijkheden nodig zoals het printen van een aangemaakte gebruiker. Op deze print worden de belangrijkste gegevens van de gebruiker geplaatst zoals onder andere de gebruikersnaam, het wachtwoord en zijn pincode. De beheerder kan deze gegevens aan de gebruiker toesturen en met behulp van deze gegevens kan de gebruiker zich aanmelding op de webapplicatie.

2.3 Functiebeheer

Het functiebeheer geeft op de eerste plaats de SMRA beheerder de mogelijkheid om functies aan te maken en te verwijderen. De aangemaakte functies kunnen worden toegevoegd of verwijderd voor gebruik bij een onderliggende gebruiker. Zo stelt een SMRA beheerder functies beschikbaar voor gemeentelijke beheers en kan een gemeentelijke beheerder zijn toegestane functies doorgeven aan onderliggende functionarissen binnen zijn kerkelijke gemeente.

Binnen het functiebeheer mogen alleen functies worden doorgegeven aan gebruikers die gekoppeld zijn aan een gebruikerstype waaraan de functie ook is gekoppeld. Zo mogen beheerfuncties alleen worden doorgegeven aan gemeentelijke beheerders maar niet aan functionarissen. Ook mogen gemeentelijke beheerders niet alleen functies toegewezen krijgen. Er zijn functies die specifiek zijn bedoeld voor de SMRA beheerders. Deze mogen ook niet worden doorgegeven aan gebruikers met een ander gebruikerstype.

2.4 Blokkadebeheer

Het systeem dient ook de mogelijkheid te beiden om blokkades te plaatsen. Op de eerste plaats dient de mogelijkheid aanwezig te zijn om klanten (kerkelijke gemeenten) en

gebruikers te blokkeren. Dit geeft de mogelijkheid om gebruikers of klanten te blokkeren als er op een onwenselijke manier gebruik wordt gemaakt van de beschikbaar gestelde functies. Een gemeentelijke beheerder mag alleen onderliggende functionarissen blokkeren of deblokkeren, mits de blokkade was geplaatst door een SMRA beheerder. Een gemeentelijke beheerder mag geen blokkade weghalen die door een SMRA beheerder is geplaatst.

Hiernaast dient ook de mogelijkheid tot het blokkeren en deblokkeren van alle gebruikers van het systeem aanwezig te zijn. Deze mogelijkheid mag niet worden uitgevoerd door een gemeentelijke beheerder.

Een ander onderdeel van het Blokkadebeheer is het blokkeren van functies. Dit vindt plaats op 3 niveau's. Een SMRA beheerder kan functies blokkeren en deblokkeren op het niveau van het systeem of voor een klant en een gemeentelijke beheerder kan functies blokkeren en deblokkeren op het niveau van functionarissen (alleen functionarissen binnen zijn kerkelijke gemeente).

2.5 Algemeen

Er is ook een functie die algemeen beschikbaar is voor de gebruikers van het systeem, met uitzondering van een gebruiker van het type SMRA telefoniste. Deze functie is voor het wijzigen van NAW gegevens. Met behulp van deze functie worden de gebruikers van het systeem in staat gesteld hun NAW gegevens te wijzigen.

2.6 Controles

Bij het uitvoeren van de functies zijn er een aantal verplichtingen binnen het systeem die moeten worden aan gehouden. Om ervoor te zorgen dat het systeem zich houdt aan deze verplichtingen moeten er controles in de functies worden verwerkt.

Bij het ontwikkelen van de functies moet rekening worden gehouden met de volgende verplichtingen:

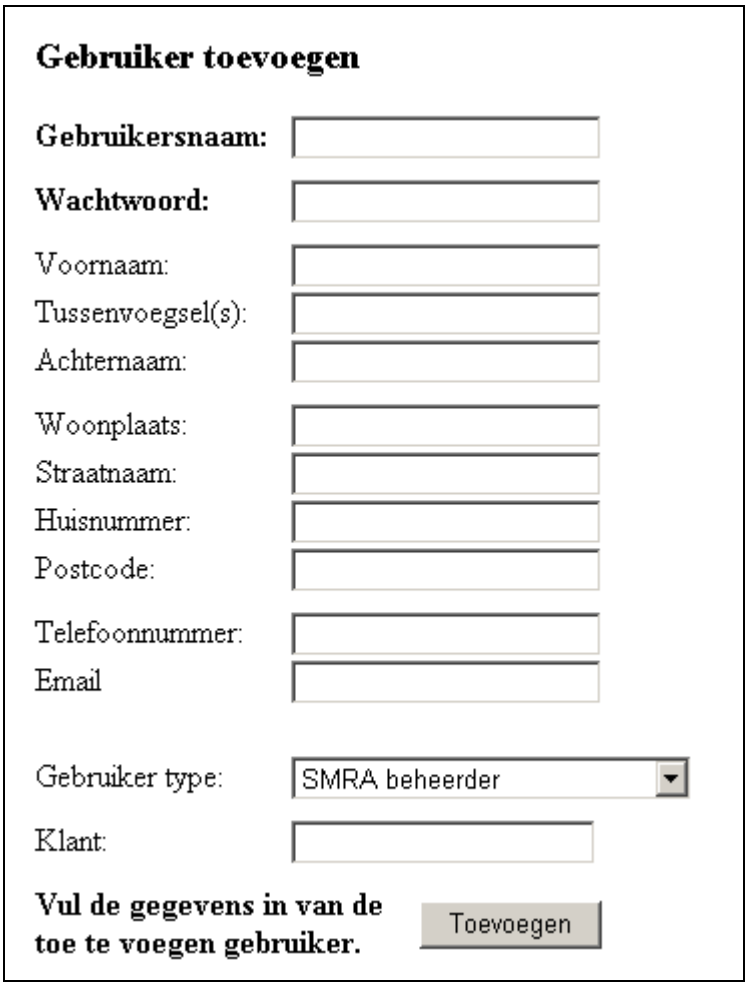
- Een gemeentelijke beheerder, functionaris of een SMRA beheerder kan een pincode niet wijzigen
- Een gebruikersaccount kan alleen schriftelijk worden aangevraagd bij een gemeentelijke pasbeheerder of een SMRA beheerder
- Een pincode kan alleen gewijzigd worden door een gebruiker te verwijderen en opnieuw aan te maken (schriftelijk aanvragen met opgave van oude toegangspas en pincode)
- De gemeente die bij een gebruiker hoort, mag niet worden gewijzigd
- Van een gebruiker moet een print kunnen worden gemaakt die kan worden verstuurd als brief naar een gemeentelijke beheerder of functionaris. Hierop is zijn de mogelijke functies, de persoonlijke gegevens, de toegangsgegevens en de pincode gegeven.
- De beheergegevens van de gemeentelijke pasbeheerders zijn uitsluitend te wijzigen door de SMRA.
- De gemeentelijke beheerders en de lokale beheerders krijgen uitsluitend toegang tot de website door het correct invoeren van hun toegangspascode en hun pincode
- Per gemeente maximaal 1 gemeentelijke beheerder

2.7 Prototype gebruikersinterface

In dit onderdeel zullen de prototypen van de gebruikersinterfaces aan bod komen.

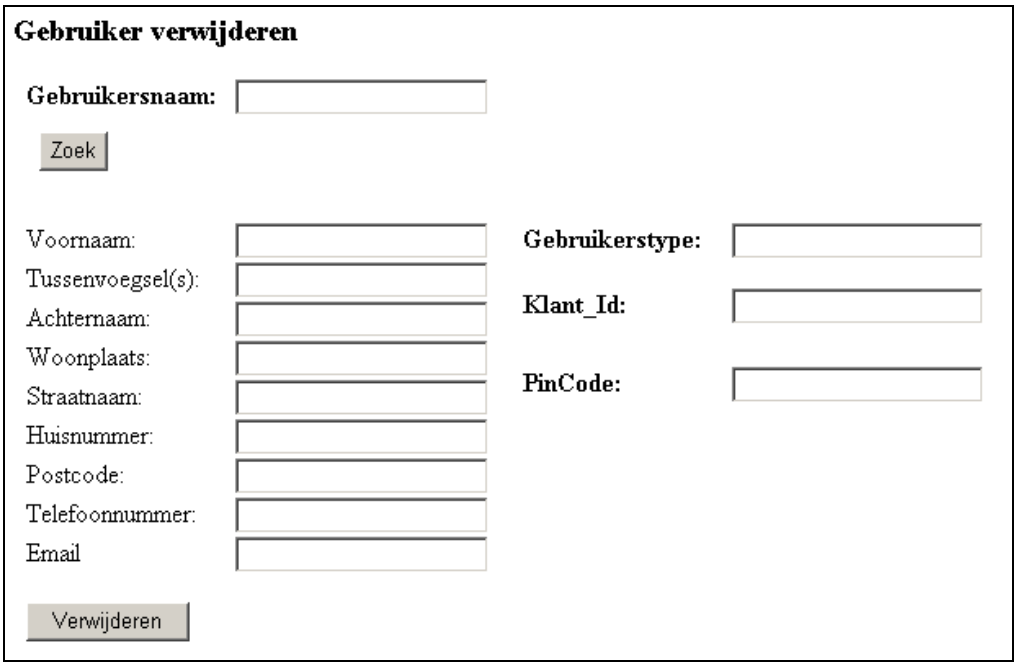
2.7.1 Gebruikersbeheer (SRMA beheerder)

Het onderstaande scherm is bedoeld voor de SMRA beheerder. Dit scherm geeft de mogelijkheid om nieuwe gebruikers aan te maken.

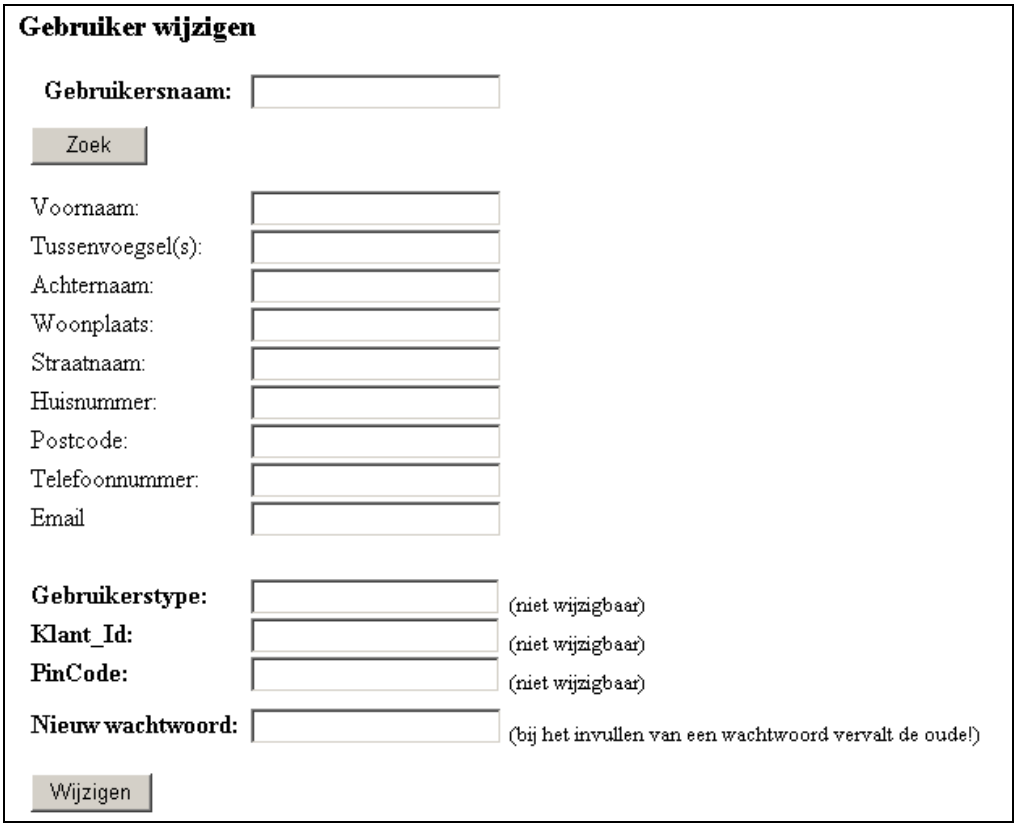


Dit scherm geeft de mogelijkheid een SMRA beheerder, gemeentelijke functionaris of een SMRA telefoniste aan te maken.

Het volgende scherm geeft een SMRA beheerder de mogelijkheid om een gebruiker te verwijderen.



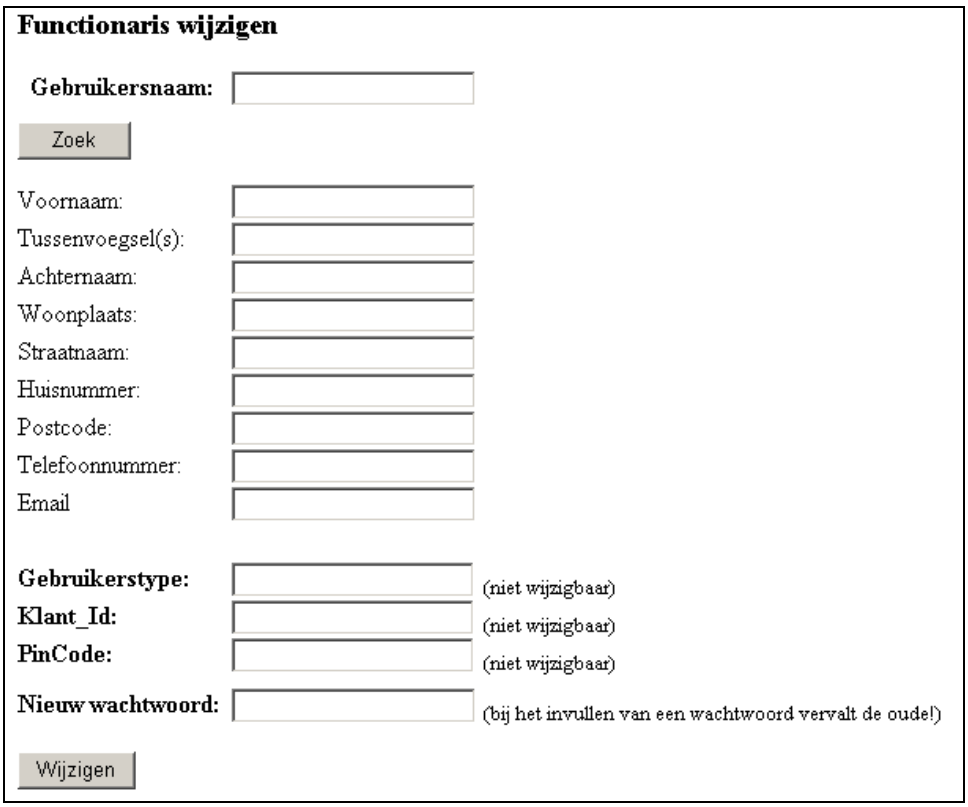
Binnen het systeem kunnen ook de gegevens van de gebruikers worden gewijzigd. Hieronder is een prototype gegeven van het scherm om de gebruikers te wijzigen.



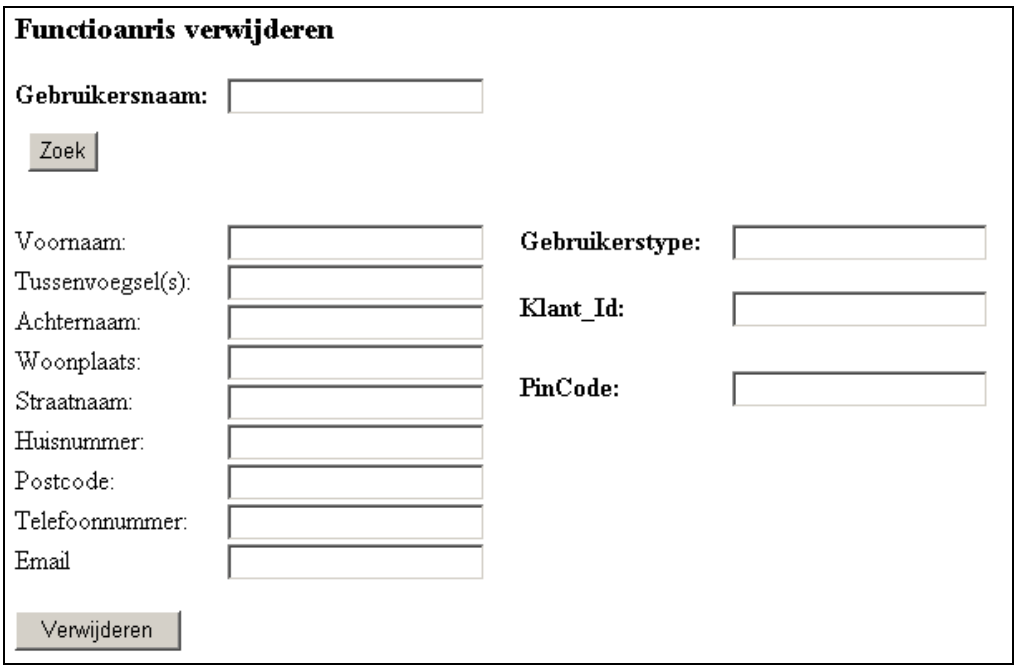
2.7.2 Gebruikersbeheer (Gemeentelijke beheerder)

Het gebruikersbeheer voor gemeentelijke beheerders is ongeveer gelijk aan die van de SMRA beheerders. Alleen zijn de gemeentelijke beheerders alleen in staat functionarissen te beheren. Een gemeentelijke beheerder krijgt het volgende scherm als hij een nieuwe functionaris wilt toevoegen.

Ook de gegeven van de functionarissen kunnen worden gewijzigd. Hieronder is een prototype gegeven van het scherm om de gegevens van een functionaris te wijzigen.

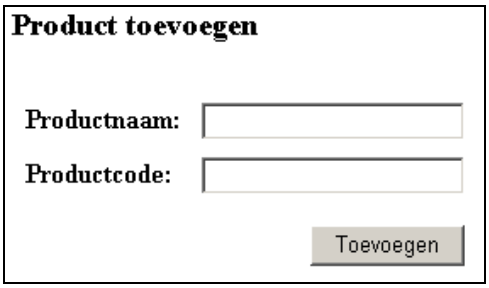


Een gemeentelijke beheerder krijgt ook de mogelijkheid om onderliggende functionarissen te verwijderen. Hieronder is een prototype gegeven van het scherm waarin functionarissen kunnen worden verwijderd.

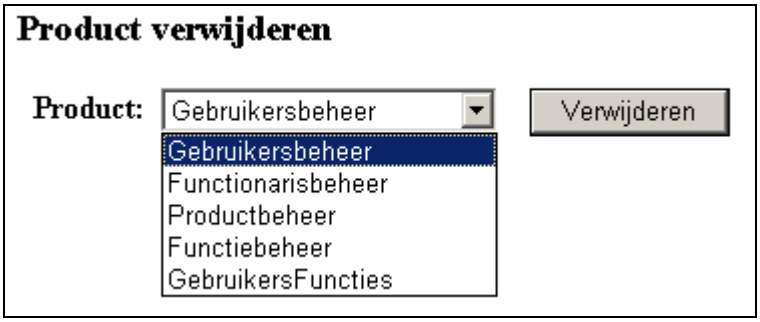


2.7.3 Productbeheer (Functiebeheer)

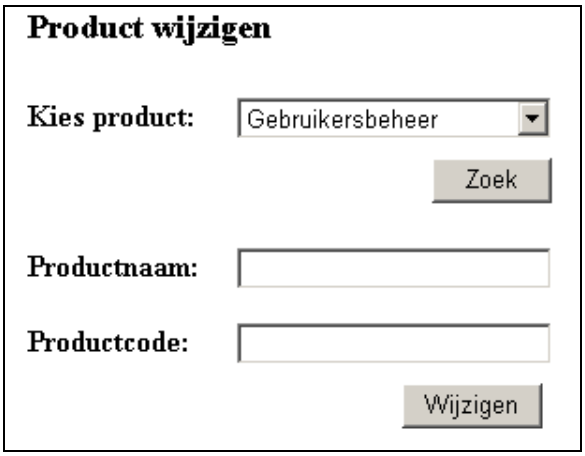
De producten binnen het systeem dienen te kunnen worden beheerd. Zo is het noodzakelijk dat er producten kunnen worden toegevoegd. Hieronder is een prototype gegeven van het scherm om producten toe te voegen.



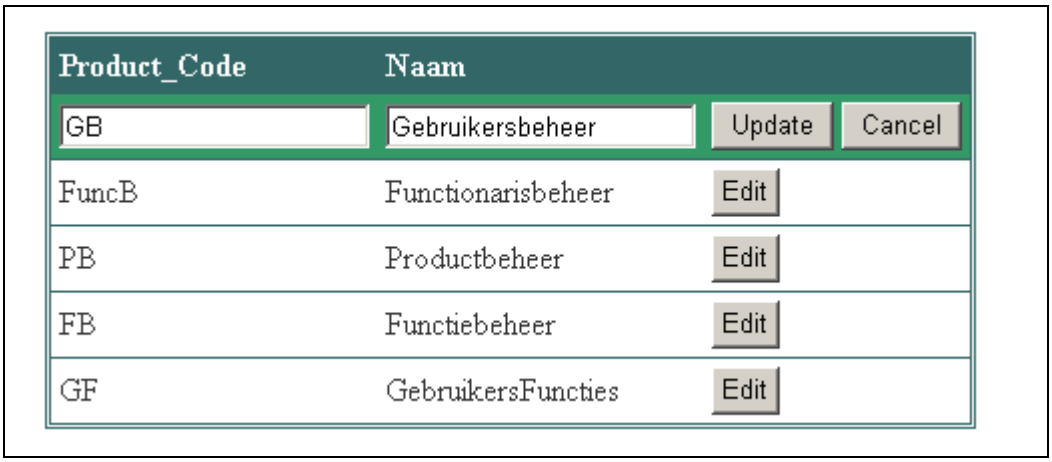
Producten moeten ook kunnen worden verwijderd. Hieronder is een prototype gegeven van het scherm om producten te verwijderen.



Het kan voorkomen dat de productnaam of code van een product gewijzigd moet worden. Ook van dit scherm is een prototype gemaakt. Deze is hieronder weergegeven.

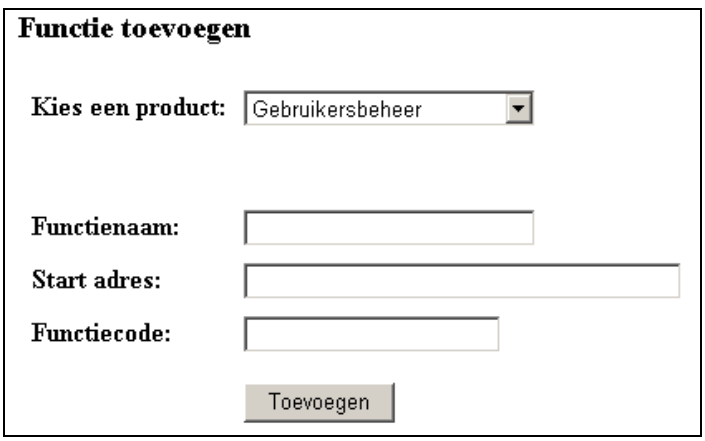


Een andere mogelijkheid is om dit te doen met editable datagrids. Hieronder is een prototype gegeven van het scherm product wijzigen indien er gebruik gemaakt zal worden van editable datagrids.

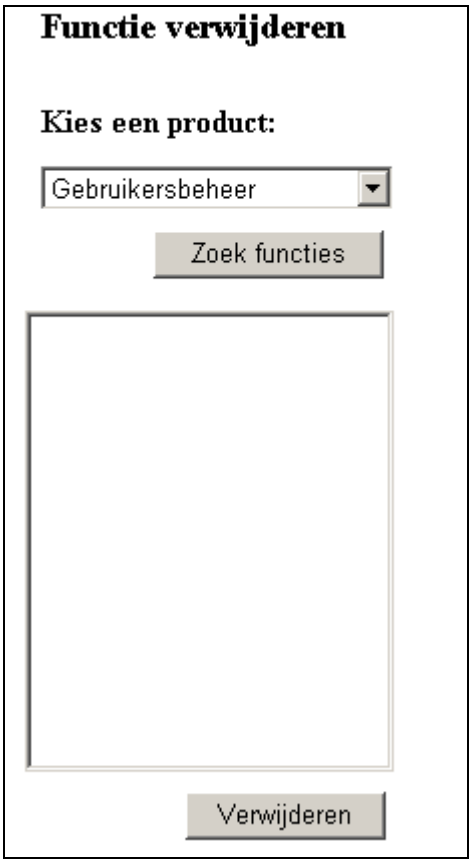


2.7.4 Functiebeheer (Functiebeheer)

Naast de producten moeten ook de functies kunnen worden beheerd. Hieronder is een prototype gegeven van het scherm om functies toe te voegen.



Functies die niet meer worden gebruikt kunnen worden verwijderd. Hieronder is een prototype gegeven van het scherm om functies te verwijderen.



3. Globaal technische structuur

3.1 Inleiding

In dit onderdeel van het document worden de technische aspecten rond de pilotontwikkeling gespecificeerd. In dit onderdeel wordt gegeven: de fysieke allocatie en het technischopslagmodel. Het netwerkcommunicatiemodel is in de voorgaande pilot besproken.

3.2 Fysieke allocatie

De functies die tijdens deze pilot worden ontwikkeld worden geplaatst onder de webapplicatie die reeds in de voorgaande pilot is opgebouwd. Binnen de webapplicatie is er een map geplaatst genaamd Functies. De reden hiervoor is dat functiepagina's alleen opgevraagd mogen worden door personen die deze expliciet toegewezen hebben gekregen. De http module controleert bij elke aanvraag uit deze map of de gebruiker toegang heeft tot het opvragen van de bron. Alle functies worden dus uiteindelijk geplaatst in:

http://www.smra.nl/beveiligingssysteem/Functies/

In de map worden de algemene functies geplaatst. Verder bevinden zich in deze map ook andere mappen genaamd: SMRAbeheerder, GemBeheerder en Functionaris. Hiernaast kunnen in een later stadium ook andere mappen of submappen worden aangemaakt voor de ordening van de functies.

4. Pilotontwikkelplan

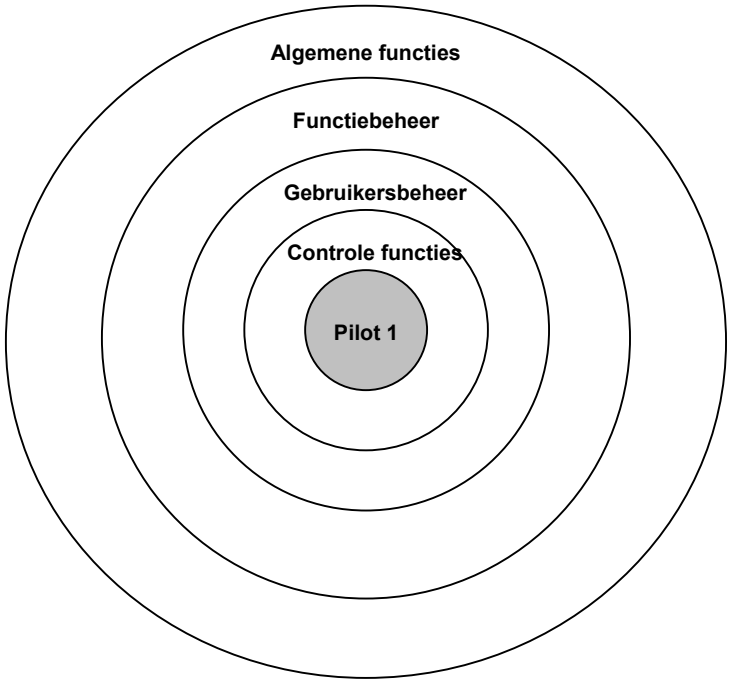
4.1 Inleiding

In dit onderdeel van het document wordt er een pilot ontwikkelplan gerealiseerd. Deze wordt ontworpen op basis van de gemaakte globale functionele en technische specificaties. Het pilotplan beschrijft hoe de pilot gaat worden gebouwd. De pilot dient te worden opgedeeld in pilotdelen en bouweenheden die achtereenvolgens dan wel parallel kunnen worden ontwikkeld. Tevens worden er voorzieningen getroffen voor het integreren en testen van de pilotdelen.

4.2 Pilotdelen

De pilot wordt opgedeeld in samenhangende componenten die afzonderlijk kunnen worden ontwikkeld, beoordeeld en getest. Er zal worden begonnen met het ontwikkelen van het gebruikersbeheer. Als het systeem de mogelijkheid kent om gebruikers aan te maken kan er worden gestart aan het ontwikkelen van het functiebeheer. Een aangemaakte gebruiker kan zo de functies beheren voor de bestaande gebruikers.

Hierna is het mogelijk om de algemene functies aan te maken en beschikbaar te stellen via het functiebeheer. Tijdens het ontwikkelen van de functies (alle functies uit deze pilot) dient er rekening te worden gehouden met de controle functies. De beperkingen die hier worden opgegeven dienen niet verloren te gaan bij het ontwikkelen van een functie. Uiteindelijk dienen de functies de beperkingen niet te overschrijden en mocht dit toch gebeuren dan dient er alsnog een controle voor de functie te worden gebouwd.



4.3 De bouweenheden

In dit deel van het document wordt elk pilotdeel opgesplitst in bouweenheden die afzonderlijk kunnen worden gebouwd. Ook zijn de stappen gegeven die moeten worden uitgevoerd voordat er kan worden gebouwd.

4.3.1 Gebruikersbeheer

GebruikerAanmaken
Zowel de SMRA beheerders als de gemeentelijke beheerders krijgen de mogelijkheid om onderliggende gebruikers aan te maken. Omdat deze twee gebruikertypen gebruikers mogen aanmaken op een verschillend niveau is het van belang dat er bij het aanvragen van de functie, het gebruikerstype wordt gecontroleerd. Een SMRA beheerder kan alleen gemeentelijke beheerders aanmaken en een gemeentelijke beheerder kan alleen functionarissen aanmaken in de kerkelijke gemeenten waarin hij zich bevindt.

GebruikerWijzigen
Deze functie is voor de SMRA beheerder en de gemeentelijke beheerder. Deze functie geeft de mogelijkheid om onderliggende gebruikers op te vragen en hun gegevens te wijzigen.

GebruikerPrinten
Deze functie wordt opgeroepen na het aanmaken van een gebruiker en geeft de mogelijkheid de gebruikergegevens af te drukken.

GebruikerVerwijderen
Deze functie is ook beschikbaar voor SMRA beheerders en gemeentelijke beheerders. Als een gebruiker wordt verwijderd moet zijn gebruikersnaam worden ingegeven.

GebruikerOpvragen
Deze functie is beschikbaar voor de SMRA beheerder, gemeentelijke beheerder en de smra telefoniste. Deze functie moet het gebruikertype controleren van de gebruiker die de functie opvraagt. Als een smra telefoniste deze functie opvraagt dient de gebruikersnaam en de pincode van de gebruiker te worden ingegeven. Een SMRA beheerder of een gemeentelijke beheerder hoeft alleen de gebruikersnaam in te voeren, maar een gemeentelijke beheerder kan alleen gebruikers opvragen die zich in zijn kerkelijke gemeenten bevinden. .

4.3.2 Functie beheer

FunctieAanmaken
Alleen voor de SMRA beheerder. Een functie dient in een .aspx pagina te worden geplaatst en in de map functies. Daarna dient de locatie van de nieuwe functie inclusief de bestandsnaam te worden opgegeven en dient er een naam te worden opgegeven voor de functie.

FunctieVerwijderen
Alleen voor de SMRA beheerder. Voor het verwijderen van een functie dient de locatie en de bestandsnaam te worden opgegeven.

FunctiesGebruikerToevoegen
Deze functie maakt het mogelijk om functies aan een gebruiker toe te kennen. Deze functie dient te aan de hand van het gebruikerstype en zijn beschikbare functies te bepalen welke functies hij aan een onderliggende gebruiker kan toekennen. Hierbij komt kijken dat bepaalde beheersfuncties niet mogen worden doorgegeven aan een gebruikerstype dat deze functies niet zou mogen uitvoeren.

FunctiesGebruikerVerwijderen
Voor de SMRA beheerders en gemeentelijke beheerders. Deze functie doet het tegenovergestelde als de vorige functie.

4.3.3 Blokkadefuncties

GebruikerBlokkeren
Deze functie is voor zowel de smra beheerder als een gemeentelijke beheerder. De functie bepaald het gebruikerstype en het bereik waarin de beheerder een gebruikersblokkade mag plaatsen op het niveau van een gebruiker.

GebruikerDeblokkeren
Deze functie is voor zowel de smra beheerder als een gemeentelijke beheerder. De functie bepaald het gebruikerstype en het bereik waarin de beheerder een gebruikersblokkade mag opgeheven op het niveau van een gebruiker.

KlantBlokkeren
Deze functie is uitsluitend voor de SMRA beheerder. Deze functie geeft de smra beheerder de mogelijkheid om een gebruikersblokkade te plaatsen op alle gebruikers uit de opgegeven gemeente.

KlantDeblokkeren
Deze functie is uitsluitend voor de SMRA beheerder. Deze functie geeft de smra beheerder de mogelijkheid om een gebruikersblokkade op te heffen voor gebruikers uit de opgegeven gemeente.

GebruikersSysteemBlokkeren
Deze functie is ook uitsluitend voor de SMRA beheerder. Deze functie geeft de SMRA beheerder de mogelijkheid om alle gebruikers van het systeem te blokkeren (behalve smra beheerders).

GebruikersSysteemDeBlokkeren
Deze functie is ook uitsluitend voor de SMRA beheerder. Deze functie geeft de SMRA beheerder de mogelijkheid om alle gebruikers van het systeem te deblokkeren.

FunctieBlokkeren
Voor de SMRA beheerders en gemeentelijke beheerders. Een functie kan worden geblokkeerd op 3 niveau's. Een functie kan worden geblokkeerd:

- Voor een gebruiker;
- Voor een klant;
- Voor alle gebruikers.

Een SMRA beheerder kan functie blokkeren voor alle gebruiker of voor de gebruikers van een klant. Een gemeentelijke beheerder kan alleen een functie blokkeren voor onderliggende gebruikers.
Een blokkade kan worden geplaatst door de functie te kiezen en het niveau van de blokkade aan te geven. Afhankelijk van het niveau dient er een gebruikersnaam of **klant_Id** te worden meegegeven.

FunctieDeblokkeren
Voor de SMRA beheerders en gemeentelijke beheerders. Deze functie doet het tegenovergestelde als de vorige functie. Bij het deblokkeren van een functie dient het niveau te worden opgegeven en afhankelijk van het niveau een gebruikersnaam of **klant_Id**.

4.3.4 Algemene functies

NawWijzigen()
Elke gebruiker (behalve een SMRA telefoniste) heeft de mogelijkheid zijn eigen NAW gegevens te wijzigen.

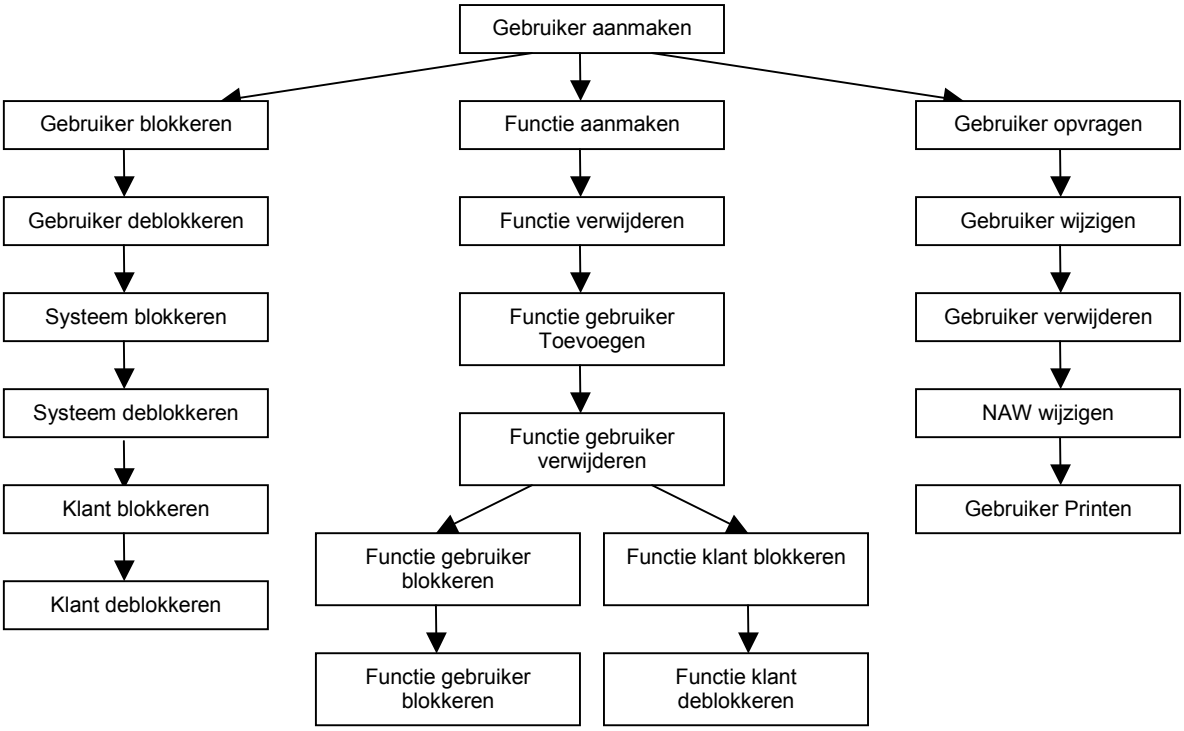
4.3.5 Controles

Bij het uitvoeren van de functies zijn er een aantal verplichtingen binnen het systeem die moeten worden aan gehouden. Om ervoor te zorgen dat het systeem zich houdt aan deze verplichtingen moeten er controles in de functies worden verwerkt.

Bij het ontwikkelen van de functies moet rekening worden gehouden met de verplichtingen die zijn genoemd in paragraaf 2.6.

4.4 Synchronisatie bouweenheden

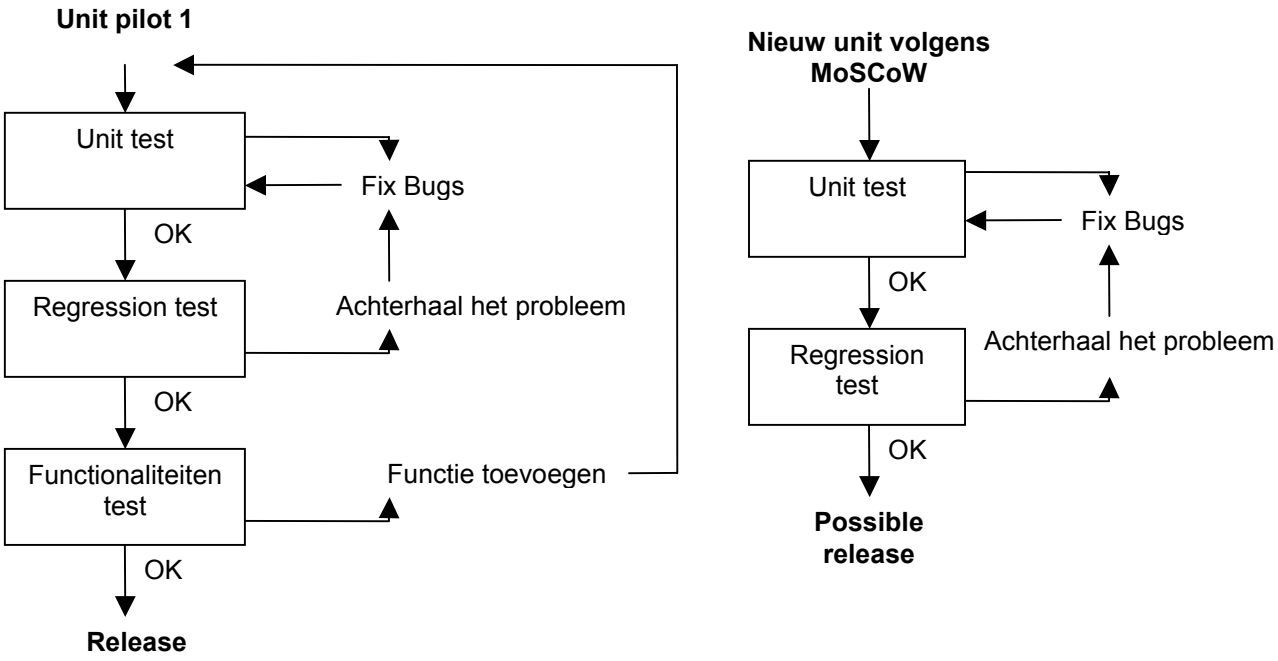
In dit model wordt duidelijk in welke volgorde, per bouweenheid, de tweede pilot van het beveiligingssysteem ontwikkeld zal gaan worden.



4.5 Voorbereiden beoordelen en testen pilotdelen

Voor het testen van de webapplicatie zal er een unittest per unit plaatsvinden, een regressietest en een functionaliteitentest worden gedaan. Bij de Unit test worden alle aparte units (functionaliteiten in de software) apart getest.
Na het samenvoegen van de units wordt er een regression test gedaan. Mochten er fouten blijken te zitten in de applicatie tijdens het uitvoeren van de regression test, worden deze problemen achterhaald en worden de nodige aanpassingen gedaan om het probleem op te lossen.
Hierna zal er worden getest op functionaliteiten. In deze test wordt er simpelweg gekeken of de betreffende functionaliteiten uit de definitiestudie in het pilotdeel zijn verwerkt. Afhankelijk van de MoSCoW prioriteit wordt dit bepaald.

Tijdens pilot 3 zullen er extra functionaliteiten worden toegevoegd en zal elke nieuwe unit afzonderlijk worden getest en zal er gebruikt worden gemaakt van een aangepaste of aangevulde regression test. Hieronder is een schematische weergave gegeven van het verloop van het testen.



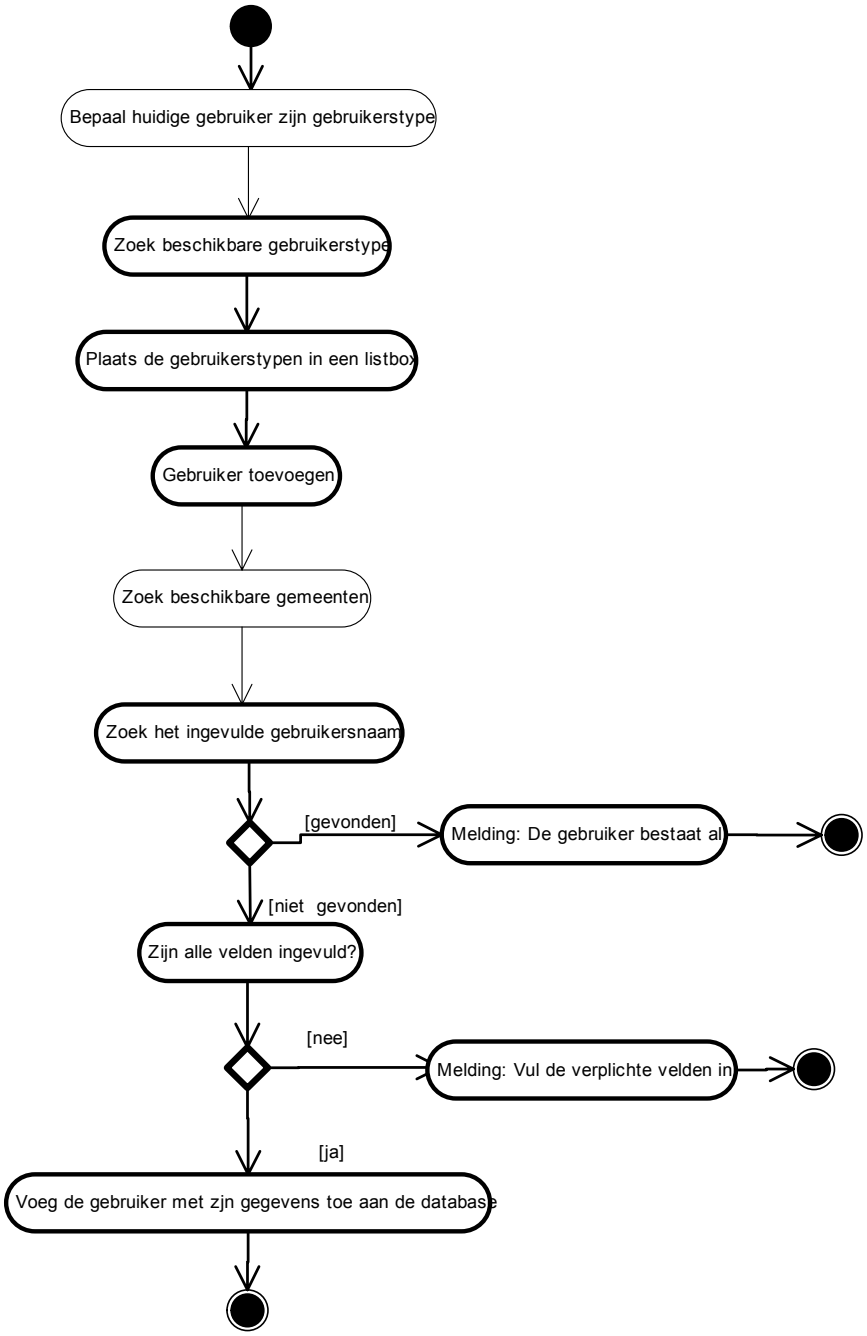
5. Ontwerp software bouweenheden

5.1 Inleiding

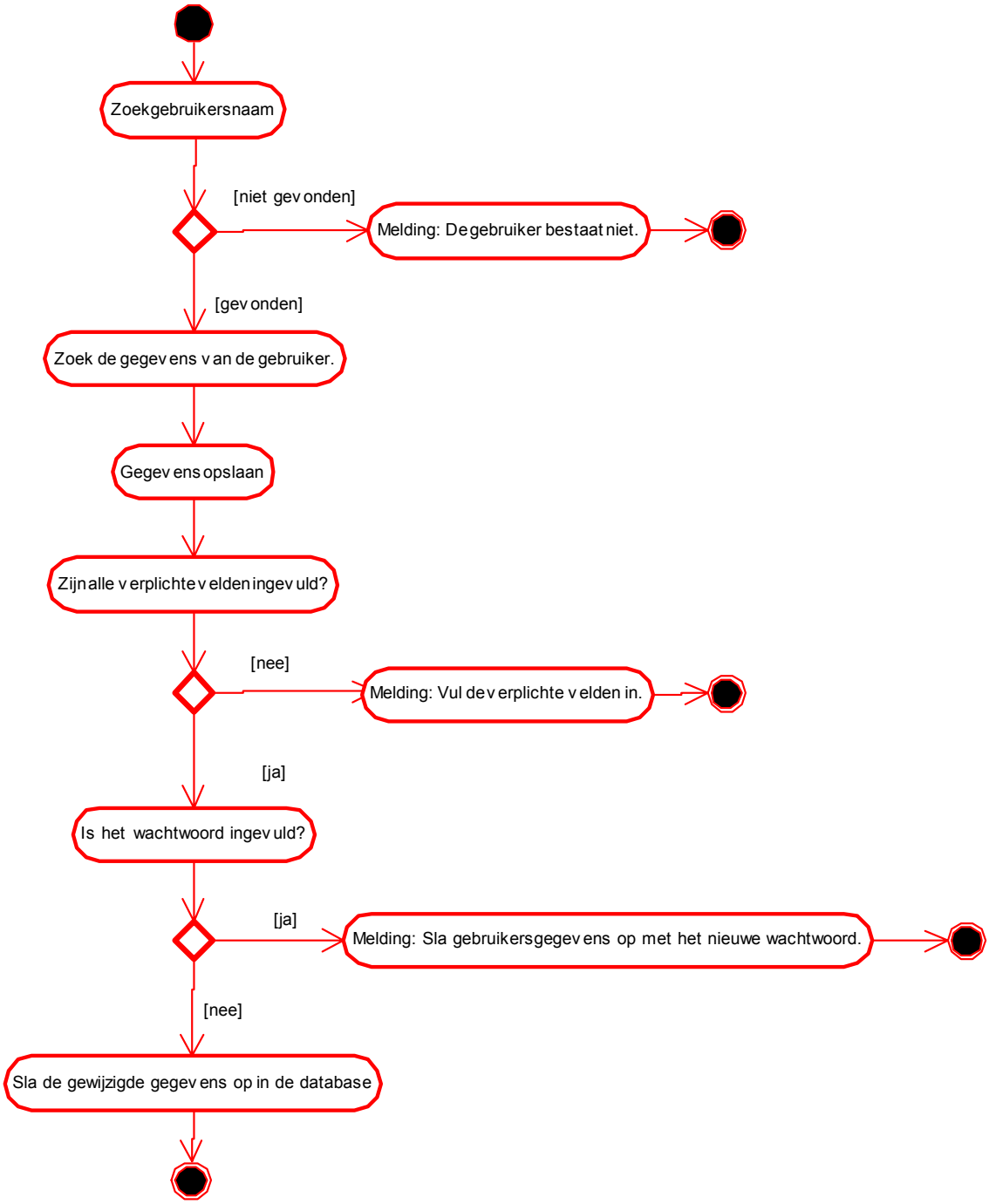
In dit hoofdstuk wordt de technische structuur van de software-bouweenheden die deel uitmaken van deze pilot besproken. In dit onderdeel worden besproken: de componenten afgebeeld op de software architectuur, de detailgegevenspecificaties, detailprocesspecificaties en de testspecificaties.

5.2 Gebruikersbeheer

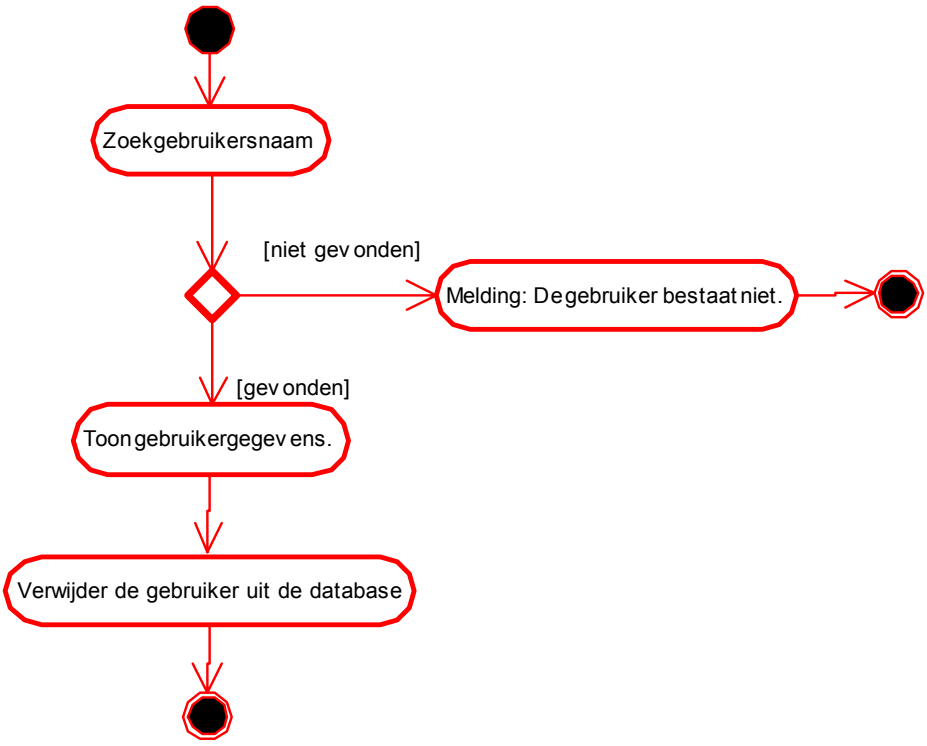
5.2.1 GebruikerAanmaken



5.2.2 GebruikerWijzigen



5.2.3 GebruikerVerwijderen



5.2.4 GebruikerBlokkeren

Nog niet ontwikkeld.

5.2.5 GebruikerDeblokkeren

Nog niet ontwikkeld.

5.2.6 KlantBlokkeren

Nog niet ontwikkeld.

5.2.7 KlantDeblokkeren

Nog niet ontwikkeld.

5.2.8 GebruikersSysteemBlokkeren

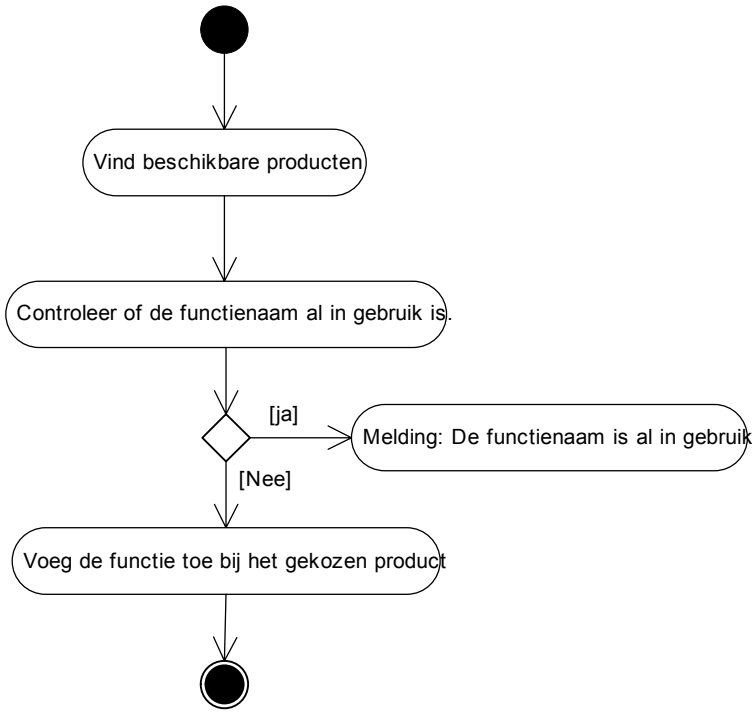
Nog niet ontwikkeld.

5.2.9 GebruikersSysteemDeBlokkeren

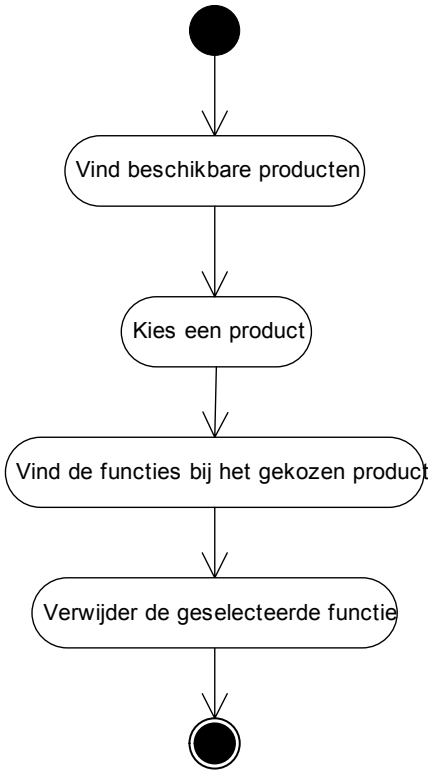
Nog niet ontwikkeld.

5.3 Functiebeheer

5.3.1 FunctieAanmaken



5.3.2 FunctieVerwijderen



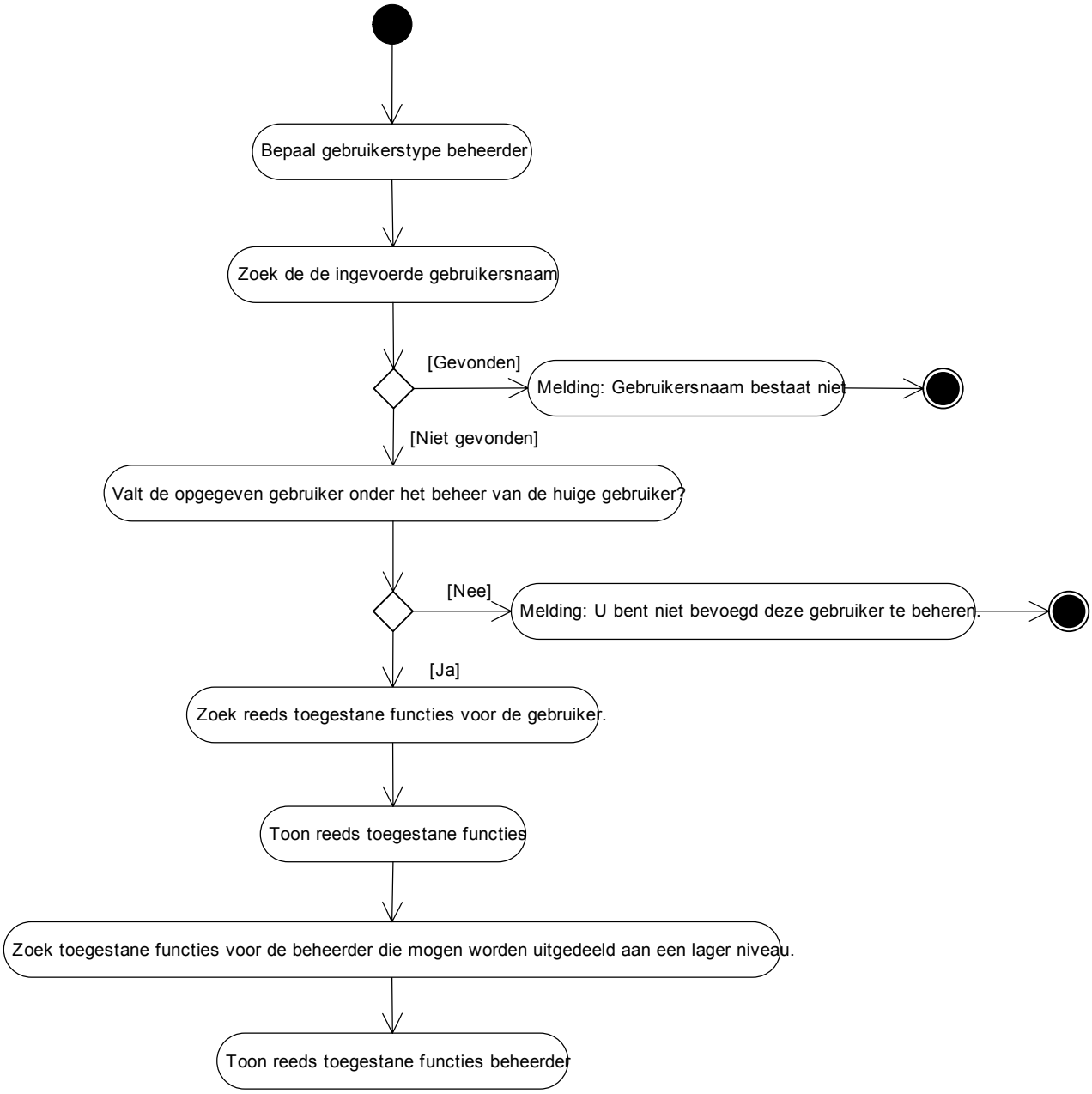
5.3.3 FunctieBlokkeren

Nog niet ontwikkeld.

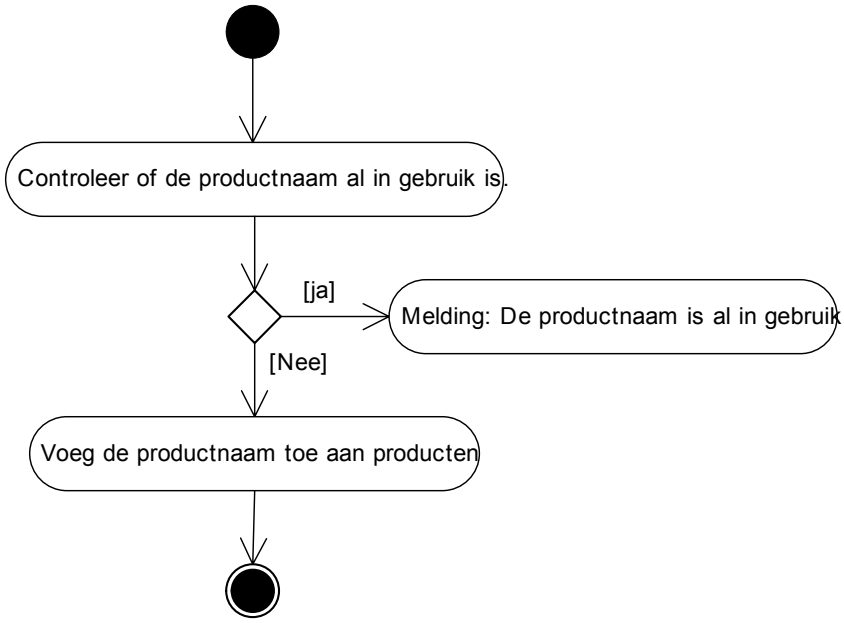
5.3.4 FunctieDeblokkeren

Nog niet ontwikkeld.

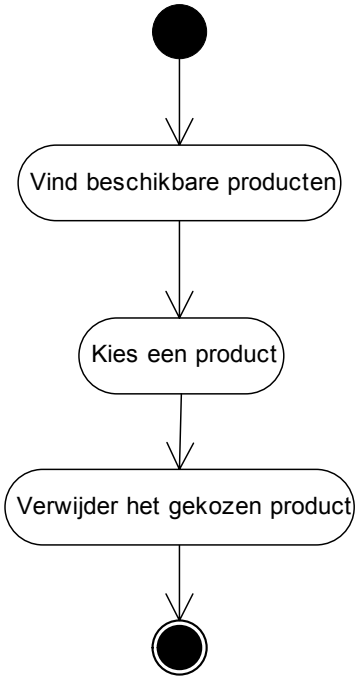
5.3.5 FunctiesGebruikerWijzigen



5.3.6 ProductAanmaken



5.3.7 ProductVerwijderen



5.4 Algemene bouweenheden

5.4.1 NawWijzigen

Nog niet ontwikkeld.

6. Testspecificaties

6.1 Inleiding

In dit onderdeel van het document worden de testspecificaties gegeven zoals deze in het pilotontwikkelplan zijn aangegeven.

6.2 Unittest testsets

6.2.1 Gebruikersbeheer

GebruikerAanmaken

Omschrijving	**Verwacht resultaat**
Nieuwe gebruiker aanmaken met alle gegevens ingevuld.	Er wordt een nieuwe gebruiker aangemaakt.
Gebruiker aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.
Gebruiker aanmaken met alle verplichte velden ingevuld en alle niet verplichte velden niet ingevuld.	Er wordt een nieuwe gebruiker aangemaakt.
Een gebruiker aanmaken met een al bestaande gebruikersnaam.	Er wordt een melding gegeven dat de gebruiker niet kan worden aangemaakt omdat de gebruikersnaam al in gebruik is.
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.

GebruikerWijzigen

Omschrijving	**Verwacht resultaat**
Een niet bestaande gebruiker opvragen.	Er wordt een melding gegeven dat de gebruikersnaam niet bestaat.
Van een bestaande gebruiker al zijn gegevens wijzigen.	De gegevens worden opgeslagen.
Verplichte velden leeg maken.	Er wordt melding gegeven dat er een verplicht veld niet is ingevuld.
Niet verplicht veld leegmaken.	De gegevens worden opgeslagen.
Een gebruikersnaam opvragen die niet onder het beheer van de ingelogde beheerder valt.	Er wordt een melding gegeven dat de gegevens van deze gebruiker niet mogen worden opgevraagd of gewijzigd.
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.

GebruikerVerwijderen

Omschrijving	**Verwacht resultaat**
Een bestaande gebruiker verwijderen.	De gebruiker wordt uit het systeem verwijderd.
Een niet bestaande gebruiker opvragen.	Er wordt een melding gegeven dat de gebruikersnaam niet bestaat.
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.
De gebruiker verwijderen voordat zijn gegevens zijn opgezocht.	Er wordt een melding gegeven dat de gegevens van de gebruiker eerst moeten worden opgezocht en ingezien voordat een gebruiker mag worden verwijderd.

6.2.2 Functiebeheer

FunctieAanmaken

Omschrijving	Verwacht resultaat
Nieuwe functie aanmaken met alle gegevens ingevuld.	Er wordt een nieuwe functie aangemaakt.
Functie aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.
Een functie aanmaken met een al bestaande functienaam.	Er wordt een melding gegeven dat de functie niet kan worden aangemaakt omdat de functienaam al in gebruik is.
Een lege functienaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.

FunctieVerwijderen

Omschrijving	Verwacht resultaat
Een bestaande functie verwijderen.	De functie wordt uit het systeem verwijderd.
Geen functienaam opgeven maar wel verwijderen kiezen.	Melding met het bericht dat er een verplicht veld niet is ingevuld.

Product aanmaken

Omschrijving	Verwacht resultaat
Nieuw product aanmaken met alle gegevens ingevuld.	Er wordt een nieuw product aangemaakt.
Product aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.
Een product aanmaken met een al bestaande productnaam.	Er wordt een melding gegeven dat het product niet kan worden aangemaakt omdat de productnaam al in gebruik is.
Een lege productnaam opgeven.	Melding met het bericht dat er een verplicht veld niet is ingevuld.

Product verwijderen

Omschrijving	Verwacht resultaat
Een bestaand product verwijderen.	Het product wordt uit het systeem verwijderd.
Geen productnaam opgeven maar wel verwijderen kiezen.	Melding met het bericht dat er een verplicht veld niet is ingevuld.
Verwijder een product dat nog onderliggende functies bevat.	Het product wordt verwijderd. (hierop is namelijk nog geen controle gemaakt.)

FunctiesGebruikerWijzigen

Omschrijving	Verwacht resultaat
Een functie doorgeven aan een onderliggende gebruiker die hij nog niet heeft.	De onderliggende gebruiker krijgt de bevoegdheid om deze functie uit te voeren.
Een functie doorgeven aan een onderliggende gebruiker die hij wel heeft.	Melding dat de gebruiker de functie al bezit.
Een functie doorgeven aan de onderliggende gebruiker die niet geschikt is voor zijn niveau.	Deze functies mogen niet worden doorgegeven omdat deze niet in de lijst voor mogen komen.
Een functie van een onderliggende gebruiker afnemen.	De bevoegdheid van de gebruiker dient tot deze functie dient te worden opgeheven.

6.2.3 Blokkadebeheer

FunctieBlokkeren (n*og niet ontwikkeld*)
FunctieDeblokkeren (n*og niet ontwikkeld*)
GebruikerBlokkeren (n*og niet ontwikkeld*)
GebruikerDeblokkeren (n*og niet ontwikkeld*)
KlantBlokkeren (n*og niet ontwikkeld*)
KlantDeblokkeren (n*og niet ontwikkeld*)
GebruikersSysteemBlokkeren (n*og niet ontwikkeld*)
GebruikersSysteemDeBlokkeren (n*og niet ontwikkeld*)

6.2.4 Algemeen

NawWijzigen (n*og niet ontwikkeld*)

6.3 Regressiontest testset

Deze test vindt alleen plaats als er softwareunits zijn gewijzigd na de afgeronde integratietest. Deze test vindt ook plaats als er nieuwe functionaliteiten aan het systeem worden toegevoegd op grond van enkele units. Op dit moment zijn er nog geen regressiontests nodig geweest.
Bij het uitvoeren van deze test worden de “logische” verwachtingen gebruikt om te testen. Dit tenzij er expliciet een melding bij wordt gegeven waarmee rekening gehouden dient te worden.

6.3.1 Gebruikersbeheer

Deze procedure controleert of gebruikers kunnen worden aangemaakt, gewijzigd en verwijderd zonder dat er rechtstreeks veranderingen hoeven te worden verricht in de database.

Deze procedure maakt gebruik van de units:

- GebruikerAanmaken
- GebruikerWijzigen
- GebruikerVerwijderen

1. Maak een nieuwe gebruiker aan met de functionaliteit GebruikerAanmaken. Vul hierbij alle nodige testgegevens in. Maak de gebruiker aan.
2. Zoek de gebruiker op bij gebruikerWijzigen en controleer of de ingevoerde gegevens overeenkomen met de gegevens op het scherm.
3. Wijzig de gegevens van de gebruiker en sla deze op.
4. Zoek de gegevens van de gebruiker nogmaals op en controleer of deze zijn gewijzigd volgens de wijzigingen die zijn gemaakt in stap 3.
5. Ga hierna naar de functie om gebruikers te verwijderen. Voer de zojuist gewijzigde gebruiker in en kiest zoeken.
6. Controleer de gegevens nogmaals en verwijder de gebruiker.
7. Zoek de gegevens van de gebruiker nogmaals op om te kunnen controleren of de gebruiker daadwerkelijk is verwijderd uit het systeem.

Indien deze procedure correct verloop en er geen problemen zijn overgehouden tijdens de unit test van deze onderdelen, wordt aangenomen dat deze onderlinge functionaliteiten foutloos met elkaar samenwerken. Dit omdat de functionaliteiten los staat van elkaar in gebruik maar wel gebruiker maken van dezelfde

gegevensbronnen. De test in dit onderdeel is dan ook om te controleren of de gegevens daadwerkelijk in de database veranderen en dat de functionaliteiten werken met de meest recente gegevens.

6.3.2 Functiebeheer

Deze procedure controleert of producten en functies kunnen worden aangemaakt, gewijzigd en verwijderd zonder dat er rechtstreeks veranderingen hoeven te worden verricht in de database. Ook wordt er gecontroleerd of de gebruikers daadwerkelijk toegang krijgen tot de voor hun beschikbaar gestelde producten.

Deze procedure maakt gebruik van de units:

- FunctieAanmaken
- FunctieVerwijderen
- Product aanmaken
- Product verwijderen
- FunctiesGebruikerWijzigen

1. Maak een nieuw product aan. Vul hierbij een aantal testgegevens in die zijn vereist bij het aanmaken van een nieuw product.
2. Probeer nog een product aan te maken met dezelfde naam.
3. Maak een nieuwe functie aan behorend bij het zojuist aangemaakte product.
4. Maak hiernaast nog een functie aan behorend bij het zojuist aangemaakte product.
5. Maak een nieuwe gebruiker aan binnen het systeem die verder gebruikt zal worden om te testen.
6. Log in onder het beheerderaccount en geef de zojuist aangemaakte gebruiker de bevoegdheid om deze twee functies te kunnen gebruiken.
7. Log hierna in onder de nieuwe gebruikersnaam en controleer of de gebruiker toestemming heeft tot het uitvoeren van de functies.
8. Gebruik het beheeraccount om de bevoegdheid tot de zojuist toegewezen functies weg te halen en log nogmaals in met de gebruiker en controleer of de gebruiker nu geen toegang meer heeft tot deze functies.
9. Gebruik het beheeraccount om 1 functie te verwijderen.
10. Probeer hierna het product te verwijderen. (het product zal worden verwijderd omdat de controle op onderliggende functies nog ontbreekt)
11. Controleer bij het aanmaken van het product of de productnaam weer gebruikt kan worden (dit toont aan dat het product niet meer bestaat).

6.3.3 Blokkadebeheer

Het beheerdeel van het beveiligingssysteem voor de blokkades is nog niet gerealiseerd. Omdat er is gekozen om eerst de huidige functies volledig af te ronden zal dit onderdeel op een later moment nog gerealiseerd worden.

6.3.4 Algemeen

NawWijzigen (n*og niet ontwikkeld*)

6.4 functionaliteitentest

De “must have” eisen en wensen die volgens de MoSCoW regels zijn vastgelegd tijdens de definitiestudie komen hier aan bod. In dit onderdeel wordt er gecontroleerd of de “must have” eisen en wensen behorende bij deze pilot, zijn verwerkt in deze pilot. Hieronder zijn de must have eisen en wensen gegeven die na afronding van deze pilot gerealiseerd zijn.

Functionele eisen/wensen telefoniste

ID	Omschrijving	MoSCoW
F1	Telefoniste dient een gebruiker te valideren door zijn pincode te verifiëren	**M**

Functionele eisen/wensen functionaris

ID	Omschrijving	MoSCoW
F2	Mag zelf zijn NAW gegevens wijzigen	**M**

Functionele eisen/wensen functionaris

ID	Omschrijving	MoSCoW
F4	Is in staat een functie aan te maken	**M**
F5	Is in staat een functie te verwijderen	**M**
F6	Is in staat om een gemeentelijke beheerder te blokkeren	**M**
F7	Is in staat om een gemeentelijke beheerder te deblokkeren	**M**
F8	Is in staat om een gemeentelijke beheerder aan te maken	**M**
F9	Is in staat om een uitdraai met gegevens maken van een gemeentelijke beheerder	**M**
F10	Is in staat om een gemeentelijke beheerder te verwijderen	**M**
F11	Is in staat om een gemeentelijke beheerder te wijzigen	**M**
F12	Is in staat om een gemeentelijke beheerder te controleren	**M**
F13	Is in staat functies voor een gemeentelijke beheerder te blokkeren en deblokkeren	**M**
F14	Is in staat functies te blokkeren per functie	**S**
F15	Is in staat functies te deblokkeren per functie	**S**
F16	Is in staat een functie te wijzigen	**C**
F17	Is in staat om een gehele gemeente te blokkeren (alle functies)	**C**
F18	Is in staat om een gehele gemeente te deblokkeren (alle functies)	**C**

Algemene functionele eisen

ID	Omschrijving	MoSCoW
F28	Een gemeentelijke beheerder, functionaris of een SMRA beheerder kan een pincode niet wijzigen	**M**
F30	Een pincode kan alleen gewijzigd worden door een gebruiker te verwijderen en opnieuw aan te maken (schriftelijk aanvragen met opgave van oude toegangspas en pincode)	**M**
F34	De gemeente die bij een gebruiker hoort, mag niet worden gewijzigd	**M**
F35	Van een gebruiker moet een print kunnen worden gemaakt die kan worden verstuurd als brief naar een gemeentelijke beheerder of functionaris. Hierop is zijn de mogelijke functies, de persoonlijke gegevens, de toegangsgegevens en de pincode gegeven.	**M**
F38	De beheergegevens van de gemeentelijke pasbeheerders zijn uitsluitend te wijzigen door de SMRA.	**M**
F39	De gemeentelijke beheerders en de lokale beheerders krijgen uitsluitend toegang tot de website door het correct invoeren van hun toegangspascode en hun pincode	**M**
F40	Per gemeente maximaal 1 pasbeheerder	**M**
F43	Een gemeentelijke beheerder kan alleen de functies die hij zelf bezit doorgeven aan onderliggende functionarissen, exclusief het aanmaken van lokale beheerders	**C**
F44	Per gemeentelijke beheerder een maximaal aantal aan te maken gebruikers.	**C**

Technische eisen

ID	Omschrijving
E1	Eenvoudige toegang voor alle applicatie onderdelen
E8	De vormgeving van de webapplicatie moet overeenkomen met de huidige website van SMRA
E9	Gebruikers kunnen worden toegekend aan een gebruikersrol

7. Testresultaten

7.1 Inleiding

In dit hoofdstuk zijn de resultaten gegeven van de uitgevoerde testen zoals die zijn beschreven in het voorgaande hoofdstuk.

7.2 Resultaten unit test

7.2.1 Gebruikersbeheer

GebruikerAanmaken

Omschrijving	Verwacht resultaat	Resultaat
Nieuwe gebruiker aanmaken met alle gegevens ingevuld.	Er wordt een nieuwe gebruiker aangemaakt.	Zoals verwacht
Gebruiker aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht
Gebruiker aanmaken met alle verplichte velden ingevuld en alle niet verplichte velden niet ingevuld.	Er wordt een nieuwe gebruiker aangemaakt.	Zoals verwacht
Een gebruiker aanmaken met een al bestaande gebruikersnaam.	Er wordt een melding gegeven dat de gebruiker niet kan worden aangemaakt omdat de gebruikersnaam al in gebruik is.	Zoals verwacht
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht

GebruikerWijzigen

Omschrijving	Verwacht resultaat	Resultaat
Een niet bestaande gebruiker opvragen.	Er wordt een melding gegeven dat de gebruikersnaam niet bestaat.	Zoals verwacht
Van een bestaande gebruiker al zijn gegevens wijzigen.	De gegevens worden opgeslagen.	Zoals verwacht
Verplichte velden leeg maken.	Er wordt melding gegeven dat er een verplicht veld niet is ingevuld.	Zoals verwacht
Niet verplicht veld leegmaken.	De gegevens worden opgeslagen.	Zoals verwacht
Een gebruikersnaam opvragen die niet onder het beheer van de ingelogde beheerder valt.	Er wordt een melding gegeven dat de gegevens van deze gebruiker niet mogen worden opgevraagd of gewijzigd.	Zoals verwacht
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht

GebruikerVerwijderen

Omschrijving	Verwacht resultaat	Resultaat
Een bestaande gebruiker verwijderen.	De gebruiker wordt uit het systeem verwijderd.	Zoals verwacht
Een niet bestaande gebruiker opvragen.	Er wordt een melding gegeven dat de gebruikersnaam niet bestaat.	Zoals verwacht
Een lege gebruikersnaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht
De gebruiker verwijderen voordat zijn gegevens zijn opgezocht.	Er wordt een melding gegeven dat de gegevens van de gebruiker eerst moeten worden opgezocht en ingezien voordat een gebruiker mag worden verwijderd.	Zoals verwacht

7.2.2 Functiebeheer

FunctieAanmaken

Omschrijving	Verwacht resultaat	Resultaat
Nieuwe functie aanmaken met alle gegevens ingevuld.	Er wordt een nieuwe functie aangemaakt.	Zoals verwacht
Functie aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht
Een functie aanmaken met een al bestaande functienaam.	Er wordt een melding gegeven dat de functie niet kan worden aangemaakt omdat de functienaam al in gebruik is.	Zoals verwacht
Een lege functienaam invoeren.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht

FunctieVerwijderen

Omschrijving	Verwacht resultaat	Resultaat
Een bestaande functie verwijderen.	De functie wordt uit het systeem verwijderd.	Zoals verwacht
Geen functienaam opgeven maar wel verwijderen kiezen.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht

Product aanmaken

Omschrijving	Verwacht resultaat	Resultaat
Nieuw product aanmaken met alle gegevens ingevuld.	Er wordt een nieuw product aangemaakt.	Zoals verwacht
Product aanmaken met een verplicht veld niet ingevuld.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht
Een product aanmaken met een al bestaande productnaam.	Er wordt een melding gegeven dat het product niet kan worden aangemaakt omdat de productnaam al in gebruik is.	Zoals verwacht
Een lege productnaam opgeven.	Melding dat een verplicht veld niet is ingevuld.	Zoals verwacht

Product verwijderen

Omschrijving	Verwacht resultaat	Resultaat
Een bestaand product verwijderen.	Het product wordt uit het systeem verwijderd.	Zoals verwacht
Geen productnaam opgeven maar wel verwijderen kiezen.	Melding met het bericht dat er een verplicht veld niet is ingevuld.	Zoals verwacht

FunctiesGebruikerWijzigen

Omschrijving	Verwacht resultaat	Resultaat
Een functie doorgeven aan een onderliggende gebruiker die hij nog niet heeft.	De onderliggende gebruiker krijgt de bevoegdheid om deze functie uit te voeren.	Zoals verwacht
Een functie doorgeven aan een onderliggende gebruiker die hij wel heeft.	Melding dat de gebruiker de functie al bezit.	Zoals verwacht
Een functie doorgeven aan de onderliggende gebruiker die niet geschikt is voor zijn niveau.	Deze functies mogen niet worden doorgegeven omdat deze niet in de lijst voor mogen komen.	Zoals verwacht
Een functie van een onderliggende gebruiker afnemen.	De bevoegdheid van de gebruiker dient tot deze functie dient te worden opgeheven.	Zoals verwacht

7.3 Regressiontest testset

Bij het uitvoeren van de test zijn de logische verwachtingen gebruikt om te testen. Hierbij is rekening gehouden met de meldingen die hiernaast expliciet zijn gegeven.

Bij het uitvoeren van de testen zijn de stappen doorlopen zoals deze in paragraaf 6.3 per onderdeel worden gegeven.

7.3.1 Gebruikersbeheer

Bij het uitvoeren van deze test is er voornamelijk gecontroleerd of er gebruikers kunnen worden aangemaakt, gewijzigd en verwijderd en dat de functies binnen het systeem met lekaars gegevens kunnen werken.

De gegevens bij het aanmaken van de gebruiker zijn ingegeven komen overeen met de gegevens die bij gebruiker wijzigen worden getoond. Als er gegevens worden gewijzigd, worden opgeslagen en opnieuw worden opgevraagd verschijnen de gewijzigde gegevens. Bij het verwijderen van de gebruiker worden deze gegevens nogmaals weergegeven en de gebruiker kan alleen na het inzien van de gegevens worden verwijderd. Na het verwijderen van de gebruiker kan deze niet meer worden opgevraagd in het systeem.

Deze test is correct verlopen.

7.3.2 Functiebeheer

Bij het uitvoeren van deze test is er voornamelijk gecontroleerd of de juiste relaties worden gelegd tussen de aangemaakte producten en functies maar ook of de toegewezen functies beschikbaar worden gesteld voor de gebruiker.

Een product aanmaken met dezelfde naam is binnen het systeem niet mogelijk. Als er een functie wordt aangemaakt is het verplicht om een product te kiezen waar hij onder valt. Als een gebruiker de bevoegdheid krijgt tot de aangemaakte functies zal hij automatisch bevoegdheid krijgen tot het product in het productmenu. Als het product wordt gekozen worden de onderliggende functies getoond in het functie menu.
Als de bevoegdheid tot de functies is ontnomen door een beheerder mag de gebruiker deze functies niet meer gebruiken als hij opnieuw inlogt. De gebruiker mag deze functies gebruiken tijdens de sessie waarin hij op dat moment in verkeerd.
Er kan een enkele functie worden verwijderd van een product en een product kan worden verwijderd met onderliggende functies. De functies worden door de database automatisch mee verwijderd. (De controle hiervoor moet nog worden gebouwd.)
Het product wordt uit het systeem verwijderd. Er kan na het verwijderen een nieuw product worden aangemaakt met de naam van het verwijderde product.

Deze test is correct verlopen.

7.3.3 Blokkadebeheer

Het beheerdeel van het beveiligingssysteem voor de blokkades is nog niet gerealiseerd. Zodra deze functies zijn gerealiseerd zullen deze alsnog worden getest.

7.4 functionaliteitentest

Functionele eisen/wensen telefoniste

ID	Omschrijving	MoSCoW	Verwerkt?
F1	Telefoniste dient een gebruiker te valideren door zijn pincode te verifiëren	**M**	**Nee. Aan deze functionaliteit is niet toegekomen**

Functionele eisen/wensen functionaris

ID	Omschrijving	MoSCoW	Verwerkt?
F2	Mag zelf zijn NAW gegevens wijzigen	**M**	**Nee. Aan deze functionaliteit is niet toegekomen**

Functionele eisen/wensen functionaris

ID	Omschrijving	MoSCoW	Verwerkt?
F4	Is in staat een functie aan te maken	**M**	**Ja.**
F5	Is in staat een functie te verwijderen	**M**	**Ja.**
F6	Is in staat om een gemeentelijke beheerder te blokkeren	**M**	**Nee. De blokkade functionaliteiten zijn nog niet gerealiseerd.**
F7	Is in staat om een gemeentelijke beheerder te deblokkeren	**M**	**Nee. De blokkade functionaliteiten zijn nog niet gerealiseerd.**
F8	Is in staat om een gemeentelijke beheerder aan te maken	**M**	**Ja.**
F9	Is in staat om een uitdraai met gegevens maken van een gemeentelijke beheerder	**M**	**Nee. Deze functie is nog niet gerealiseerd.**
F10	Is in staat om een gemeentelijke beheerder te verwijderen	**M**	**Ja.**
F11	Is in staat om een gemeentelijke beheerder te wijzigen	**M**	**Ja.**
F12	Is in staat om een gemeentelijke beheerder te controleren	**M**	**Ja.**

F13	Is in staat functies voor een gemeentelijke beheerder te blokkeren en deblokkeren	**M**	
F14	Is in staat functies te blokkeren per functie	**S**	**Nee. De blokkade functionaliteiten zijn nog niet gerealiseerd.**
F15	Is in staat functies te deblokkeren per functie	**S**	**Nee. De blokkade functionaliteiten zijn nog niet gerealiseerd.**
F16	Is in staat een functie te wijzigen	**C**	**Nee, deze functionaliteit heeft weinig toegevoegde waarde voor het systeem.**
F17	Is in staat om een gehele gemeente te blokkeren (alle functies)	**C**	**Nee. De blokkade functionaliteiten zijn nog niet gerealiseerd.**
F18	Is in staat om een gehele gemeente te deblokkeren (alle functies)	**C**	**Nee. De blokkade functionaliteiten zijn nog niet gerealiseerd.**

Algemene functionele eisen

ID	Omschrijving	MoSCoW	Verwerkt?
F28	Een gemeentelijke beheerder, functionaris of een SMRA beheerder kan een pincode niet wijzigen	**M**	
F30	Een pincode kan alleen gewijzigd worden door een gebruiker te verwijderen en opnieuw aan te maken (schriftelijk aanvragen met opgave van oude toegangspas en pincode)	**M**	**Ja.**
F34	De gemeente die bij een gebruiker hoort, mag niet worden gewijzigd	**M**	**Ja.**
F35	Van een gebruiker moet een print kunnen worden gemaakt die kan worden verstuurd als brief naar een gemeentelijke beheerder of functionaris. Hierop is zijn de mogelijke functies, de persoonlijke gegevens, de toegangsgegevens en de pincode gegeven.	**M**	**Nee. Deze functionaliteit is omgezet naar het versturen van een automatische email naar een aangemaakte gebruiker met zijn gegevens.**
F38	De beheergegevens van de gemeentelijke pasbeheerders zijn uitsluitend te wijzigen door de SMRA.	**M**	**Ja.**
F39	De gemeentelijke beheerders en de lokale beheerders krijgen uitsluitend toegang tot de website door het correct invoeren van hun toegangspascode en hun pincode	**M**	**Ja.**
F40	Per gemeente maximaal 1 pasbeheerder	**M**	**Nee. Dit is een controle functie binnen het aanmaken van gebruikers. Deze bestaat nog niet.**
F43	Een gemeentelijke beheerder kan alleen de functies die hij zelf bezit doorgeven aan onderliggende functionarissen, exclusief het aanmaken van lokale beheerders	**C**	**Ja.**
F44	Per gemeentelijke beheerder een maximaal aantal aan te maken gebruikers.	**C**	**Nee. Dit is een controle functie binnen het aanmaken van gebruikers. Deze bestaat nog niet.**

Technische eisen

ID	Omschrijving	Verwerkt?
E1	Eenvoudige toegang voor alle applicatie onderdelen	**Ja.**
E8	De vormgeving van de webapplicatie moet overeenkomen met de huidige website van SMRA	**Nee. Er is besloten dat de visuele vormgeving in een later stadium wordt gerealiseerd.**
E9	Gebruikers kunnen worden toegekend aan een gebruikersrol	**Nee. Er is besloten dat functies vallen onderproducten en dat gebruikersrollen zijn vervallen..**

7. Conclusie

Tijdens het ontwikkelen van de eerste pilot is de kernfunctionaliteit van het beveiligingsysteem ontwikkeld. Het doel van deze tweede pilot was het ontwikkelen van de beheersfunctionaliteiten binnen het beveiligingssysteem. De beheersfunctionaliteiten voor het beheer van de gebruikers, het beheer van de functies en het beheer van de producten zijn ontwikkeld en getest. De beheersfunctionaliteiten voor het beheren van de blokkades is niet tijdens deze pilot verwezenlijkt. Dit omdat er hiervoor niet genoeg tijd ter beschikking was.