Clubs Social Media BV

Afstudeerverslag

Realisatie van een interface voor het inloggen op Clubs.nl

Nico Vogelaar
21-3-2011

Afstudeerverslag: Realisatie van een interface voor het inloggen op Clubs.nl

Student:	*Nico Vogelaar*
Studentnummer:	*07019025*
Opleiding:	*Informatica*
Onderwijsinstelling:	*Haagse Hogeschool*
Locatie:	*Zoetermeer*
Examinatoren:	*G.A. Mijnarends*
	E.M. van Doorn

Afstudeerperiode:	*15 november t/m 18 maart 2011*

Bedrijf:	*Clubs Social Media BV*
Adres:	*Hanzeweg 14, Gouda*
Begeleider:	*Paul Brinkhof*
Datum:	*21 maart 2011*

Voorwoord

Dit document beschrijft de afstudeeropdracht die ik in de afgelopen 17 weken heb uitgevoerd bij Clubs Social Media BV in Gouda.

Hierbij wil ik Clubs Social Media BV bedanken voor de opdracht die zij mij hebben gegeven en ook voor de kennis die ik tijdens dit afstudeerproject heb kunnen opdoen.

Verder wil ik de volgende personen bedanken voor hun hulp bij dit afstudeerproject:

- G.A. Mijnarends
- E.M. van Doorn

21 maart 2010, Zoetermeer

Nico Vogelaar

Inhoudsopgave

1 Inleiding

Dit document is een verslag waarin ik mijn afstudeeropdracht beschrijf die ik tijdens mijn afstudeerperiode gerealiseerd heb. Mijn afstudeeropdracht heb ik gerealiseerd bij het bedrijf Clubs Social Media BV. In dit verslag zal ik de opdracht beschrijven en hoe ik deze heb uitgevoerd met de keuzes die ik hierbij gemaakt heb.

Allereerst zal ik beschrijven bij wat voor bedrijf ik mijn opdracht heb uitgevoerd met een korte uitleg over het core product van het bedrijf.

In hoofdstuk 3.3 beschrijf ik de fasering van het project, waarin duidelijk wordt dat de opdracht uit twee delen bestaat. In hoofdstuk 4 zal ik de aanpak van het project beschrijven.

In hoofdstuk 7 “Deelproject I: Social Media Connectivity Interface” beschrijf ik het eerste deelproject. Deze beschrijf ik per RUP-fase. In hoofdstuk 8 “Deelproject II: Centrale login” beschrijf ik het tweede deelproject. Deze beschrijf ik net als deelopdracht I per RUP-fase.

In hoofdstuk 10 evalueer ik het project op basis van de projecten en het proces.

2 Clubs Social Media BV

Het bedrijf Clubs Social Media BV is een jong startend mediabedrijf in Gouda dat zich bezighoudt met de ontwikkeling en het management van online communities. Een groot project waar Clubs Social Media BV zich nu voornamelijk mee bezighoudt, is www.clubs.nl. Deze website behoort tot de grootste social communities van Nederland.

2.1 Clubs.nl

Clubs is opgericht in 1999 door "Het Net" (KPN). Clubs Social Media BV heeft Clubs eind 2008 overgenomen van de KPN. In 2009 is het overgenomen systeem van Clubs gerenoveerd op een vernieuwd platform met een nieuw design en functionaliteiten. In de toekomst zal Clubs verder uitgebreid en geoptimaliseerd gaan worden. De ideeën en wensen van Cluppers spelen daarbij een belangrijke rol.

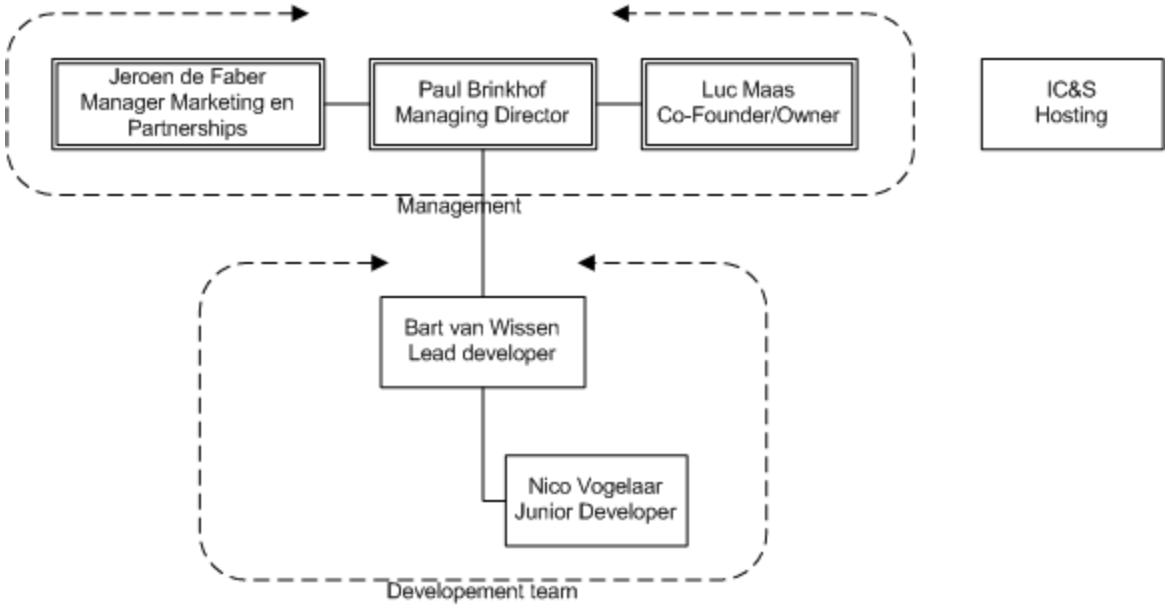
Clubs is een website waar het mogelijk is om je eigen community te bouwen over je hobby of een onderwerp waar je gepassioneerd over bent. De functionaliteiten die Clubs te bieden heeft zijn bijvoorbeeld:

- Het uploaden van foto's
- Het toevoegen van video's
- Het uploaden van bestanden
- Het plaatsen van nieuwsberichten
- Een forum
- Een blog

2.2 Organisatie

Het bedrijf Clubs Social Media BV bestaat uit ongeveer 20 medewerkers. De medewerkers bestaan uit mensen die vast in dienst zijn, stagiaires, vrijwilligers en externe mensen. Het kantoor is opgesplitst in twee delen. Er is een commerciële kant en een technische kant. De commerciële mensen zijn vooral met ideeën bezig om meer geld binnen te halen. De technische kant is vooral bezig met de ontwikkeling van nieuwe features aan het Clubs platform.

De productieomgeving waarop het Clubs platform draait wordt door een externe partij verzorgd. Het bedrijf IC&S[1] zorgt ervoor dat de Clubs website draaiende blijft en de updates uitvoert. Het Clubs platform wordt door IC&S in Dublin gehost op een Cloud bij Amazon[2].



Figuur 1 organigram

[1] http://www.ic-s.nl/clubsnl-klant-bij-ics/
[2] http://aws.amazon.com/ec2/

3 Afstudeeropdracht

3.1 Aanleiding

Het bedrijf Clubs Social Media BV heeft een communitydienst Clubs.nl. De communitydienst is op 2 november 2009 gerelaunched nadat het was overgenomen van Het Net. De communitydienst is bij de relaunch ontwikkeld op een nieuw platform. Het doel van Clubs is om een z´n groot mogelijke community te krijgen waarbij zoveel mogelijk mensen actief zijn. De groei en de activiteiten nemen echter sinds de komst van het nieuwe platform af.

Om gebruik te kunnen maken van de functionaliteiten die Clubs te bieden heeft moeten de mensen die nog geen lid zijn op Clubs zich vrij snel registreren. Het registreren zou een te hoge drempel kunnen zijn waardoor mensen weggaan en niet actief worden op Clubs, wat de opbrengst vermindert. Mensen hebben vaak al een bestaand online identiteit bij websites als Hyves[3] of Facebook. Sinds enkele jaren is het mogelijk[4][5] om gebruik te maken van deze bestaande identiteiten.

Het Clubs platform is verdeeld over een aantal verschillende domeinen voor verschillende interessegebieden. Door beperkingen in het huidige inlogsysteem is het niet mogelijk voor gebruikers om tussen deze verschillende domeinen te schakelen zonder steeds opnieuw te moeten inloggen.

Er zijn plannen om samenwerkingsverbanden aan te gaan met bedrijven zoals uitgeversconcern Sanoma en de ANWB bond. Het aantal domeinen zal hierbij in de nabije toekomst fors uitgebreid worden. Het zal dan steeds vaker voorkomen dat een gebruiker overschakelt naar een ander domein, omdat de meeste gebruikers lid zijn van meerdere clubs. Het steeds opnieuw moeten invullen van gebruikersnaam en wachtwoord werpt hierbij een hoge drempel op en gaat ten koste van het gebruiksgemak.

3.2 Probleemstelling

Hoe kan een nieuw inlogsysteem ontworpen worden, waarbij het mogelijk wordt om

- een bestaande online identiteit te gebruiken, zodat niet opnieuw een login en wachtwoord aangemaakt hoeft te worden.
- op een centraal punt in te loggen en ingelogd te blijven, ook als de gebruiker van Clubs-website A naar Clubs-website B schakelt.

3.3 Fasering

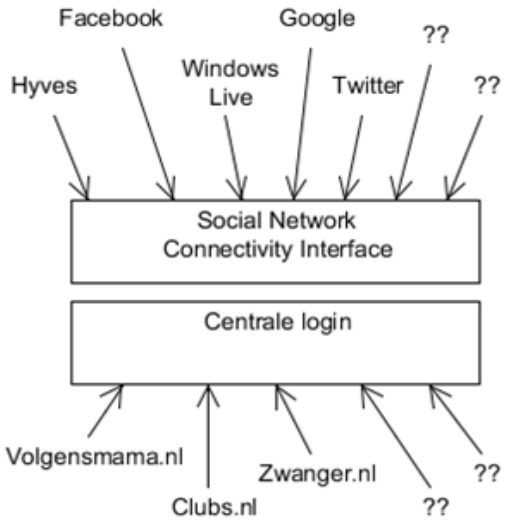
De afstudeeropdracht is opgesplitst in twee deelopdrachten. Ik heb de opdracht in tweeën gesplitst, omdat ik vond dat het eigenlijk twee losse opdrachten waren. De eerste deelopdracht gaat over het verlagen van de drempel bij het registreren (Social Network Connectivity Interface). De tweede deelopdracht gaat over het opnieuw moeten inloggen bij het overschakelen naar een ander domein (Centrale login).

[3] http://www.telegraaf.nl/digitaal/6535899/__Hyves_op_grens_tien_miljoen_leden__.html
[4] http://www.marketingfacts.nl/berichten/20090424_hyves_als_openid_provider/
[5] http://developers.facebook.com/blog/post/108/

3.4 Doelstelling

Het doel van deze opdracht is het verlagen van de drempel om lid te worden op Clubs, zodat mensen op een makkelijkere en snellere manier lid kunnen worden, waarbij het indirecte doel is dat de groei in het aantal leden toeneemt en om de mensen meer gebruik te laten maken van de functionaliteiten die Clubs te bieden heeft. Om dit te kunnen bereiken dient er een mogelijkheid te zijn om vanuit Clubs in te kunnen loggen met een bestaand identiteit van websites als Hyves, Facebook, Google, Twitter en Windows Live ID waar mensen vaak al lid van zijn. Deze websites worden ook wel identity providers genoemd.

Voor mensen die nog geen lid zijn van Clubs wordt de gebruiker tijdens de inlog procedure ook meteen geregistreerd, waardoor het inloggen eigenlijk een soort van registratie wordt. Hierdoor wordt de gebruiker toch lid op Clubs zonder zijn gegevens te hoeven invullen.

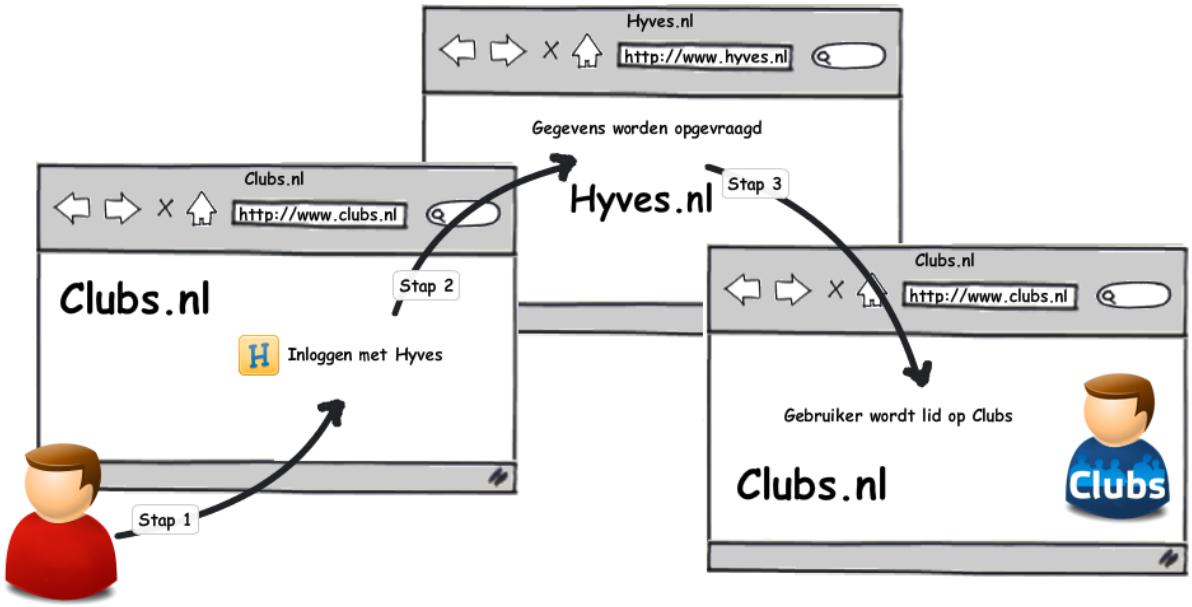


Figuur 2 Social Network Connectivity Interface & Centrale login

Social Network Connectivity Interface

De verschillende identity providers maken gebruik van verschillende protocollen. Deze protocollen kunnen in de toekomst wijzigen en het aantal netwerken moet uitgebreid kunnen worden. Hiervoor is een slimme interface noodzakelijk. De interface voorziet in een koppeling tussen de verschillende identity providers en de Clubs.nl website. In Figuur 3 is een visualisatie van de inlog procedure vanuit Clubs via een identity provider (in dit voorbeeld Hyves) met daarin de volgende stappen:

1. De gebruiker logt vanuit Clubs in met Hyves.
2. De gebruiker wordt doorverwezen naar Hyves. Wanneer de gebruiker is ingelogd op Hyves en toestemming heeft gegeven voor het delen van zijn gegevens met Clubs, dan zal de gebruiker deze stap niet te zien krijgen.
3. De gebruiker is ingelogd op Clubs. Wanneer de gebruiker nog geen lid was op Clubs, dan is de gebruiker nu geregistreerd op Clubs.



Figuur 3 Inloggen vanuit Clubs met Hyves

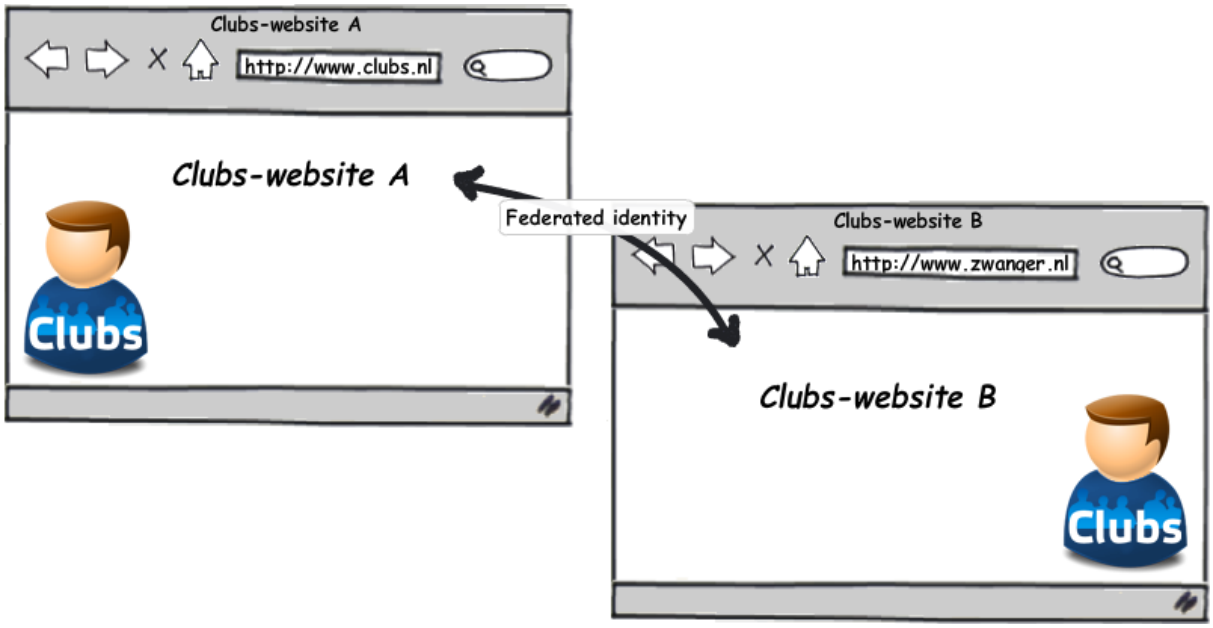
Voor de release zal er een 0-meting worden gedaan van het aantal aanmeldingen per dag om na de release te kunnen zien of deze specifieke implementatie effect heeft gehad op de groei in het aantal leden.

Centrale login

Door de samenwerking die Clubs is aangegaan met uitgeversconcern Sanoma zullen volgend jaar satellietsites van Clubs ontstaan rondom een specifiek blad/interesse gebied. Deze satellietsites zullen onder een eigen domeinnaam draaien. Gebruikers moeten probleemloos tussen de verschillende domeinen kunnen schakelen. Momenteel moet er opnieuw worden ingelogd bij het overschakelen van Club-website A naar Club-website B. Het is vanuit een beveiligingsoogpunt namelijk niet mogelijk om een ingelogde gebruiker op website B op een voor de gebruiker onzichtbare manier te laten verder werken op website A.

Het doel is om de mensen maar één keer te hoeven laten inloggen en zowel op Club-website A als Club-website B ingelogd zijn. Hierbij wil ik gebruik maken van een Federated identity. Dit is een Single Sign-on authenticatie proces dat over meerdere (externe) systemen gebruikt kan worden zonder opnieuw te hoeven inloggen. Hierbij moet het authenticatie proces op een centraal punt worden gebracht.

De implementaties zullen voordat deze in productie worden genomen getest worden in een acceptatietest-omgeving. Dit zal worden gedaan door een groep van ongeveer 60 mensen. De groep testers bestaan uit medewerkers van Clubs en mensen die lid zijn op Clubs.



Figuur 4 Federated identity

4 Aanpak

In dit hoofdstuk beschrijf ik de aanpak van het project. Hierin wordt de gehanteerde methodiek met de daarbij behorende onderdelen in het kort uitgelegd.

4.1 Projectmethodiek

De projectmethodiek die ik gekozen heb voor het uitvoeren van de opdrachten is Rational Unified Process (RUP). Dit is een gestructureerde ontwikkelmethodiek voor het ontwikkelen van software. Voor het visueel modelleren heb ik gekozen voor Unified Modeling Language (UML).

Rational Unified Process

Een van de hoofdredenen waarom ik voor de projectmethodiek RUP heb gekozen was omdat ik hier de meeste ervaring mee had. Een andere reden om deze methodiek te hanteren was omdat hier veel aandacht aan de documentatie besteed wordt. Een andere projectmethodiek genaamd eXtreme Programming (XP) heb ik ook in overweging genomen. Ik had met deze methodiek ook al een beetje ervaring opgedaan, maar een stuk minder dan met RUP.

Een reden om voor XP te kiezen en niet voor RUP is dat er te lang over het ontwerp nagedacht en gedocumenteerd wordt waardoor er pas veel later begonnen wordt met bouwen. Dit heeft aan de andere kant ook zijn voordelen, omdat een goed gedocumenteerd ontwerp vaak ook tijd bespaart met het bouwen. Het leek me mede daarom i.v.m. de bewijsvoering het beste om voor RUP te kiezen.

Ik heb voor beide deelopdrachten een apart RUP-project gestart. Ik heb hiervoor gekozen, omdat ik de deelopdrachten los van elkaar zag. Doordat er twee projecten zijn is het geheel ook overzichtelijker.

Een RUP-ontwikkelproject bestaat uit vier faseringen:

1. Inceptionfase – initialiseren van het project
2. Elaborationfase – ontwerpen van de software
3. Constructionfase – ontwikkelen van de software
4. Transitionfase – afronding

Initialiseren van het project (Inception fase)

De activiteiten van de inception fase zijn het bepalen van de haalbaarheid van het project, de inhoud en de begrenzingen (scope). De initiële functionele eisen worden tijdens deze fase verzameld en opgesteld. Deze worden vervolgens gespecificeerd in use case diagrammen.

Ontwerpen van de software (Elaboration fase)

Het opstellen van het ontwerp wordt tijdens de elaboration fase gedaan. Dit wordt gedaan op basis van de functionele eisen die vastgesteld zijn. Tijdens het uitvoeren van deze fase kunnen sommige problemen al vroegtijdig worden gesignaleerd. Het van maken van een ontwerp zorgt ervoor dat nagedacht is over de software die ontwikkeld wordt en goedkeuring voor gegeven kan worden. Door de goedkeuring wordt voorkomen dat achteraf blijkt dat het resultaat niet voldoet aan de functionele eisen.

Ontwikkelen van de software (Construction fase)

Het ontwikkelen van de software is tijdens de construction fase gedaan. Dit is gemaakt op basis van het ontwerp dat gemaakt is tijdens de elaboration fase. Aan het einde van deze fase wordt de code gedocumenteerd en de werking van de code uitgelegd.

Afronding (Transition fase)

De activiteiten van de transitional fase zijn de acceptatietest, fouten oplossen en het uitvoeren van de deployment op de productieomgeving. Het testen is door een andere groep mensen uitgevoerd. Het uitvoeren van de deployment mocht ik zelf niet doen. Ik had hiervoor niet de bevoegdheid. De deployment wordt verzorgd door een externe partij IC&S die de applicatie van Clubs beheert.

Unified Modeling Language (UML)

Voor het moduleren van de software heb ik veel gebruik gemaakt van UML. Met UML had ik zelf ook al enige ervaring. Ik heb ervoor gekozen om UML te gebruiken, omdat ik dit een duidelijke manier vind om software architectuur weer te geven. Ik heb niet altijd voor UML gekozen. Ik heb per onderdeel gekozen voor een bepaald diagram. Dit geeft de werking het beste weer. In de meeste gevallen heb ik hierbij gekozen voor een sequentiediagram. De sequentiediagrammen geven een duidelijke weergaven van de informatiestromen en de communicatie tussen de verschillende objecten.

4.2 Voortgangsbewaking

Voor mijn afstudeerproject stond een periode van 17 weken gepland, waarbij twee werkende producten opgeleverd moesten worden. Om de voortgang te bewaken heb ik een planning gemaakt en voortgangsgesprekken met Bart van Wissen (lead developer) gehad. In deze voortgangsgesprekken bespraken we in het kort waar ik mee bezig was en bekeken we of dat klopt met de planning van het plan van aanpak.

4.2.1 Planning

Aan het begin van mijn afstudeerperiode had ik een globale planning gemaakt voor de hele afstudeerperiode voor beide deelopdrachten. Ik had deze opdrachten ingepland op basis van een schatting van de hoeveelheid werk wat ik per onderdeel denk te moeten besteden. Deze heb ik in latere documenten niet nogmaals opgenomen, omdat ik het niet nodig vond om deze uitgebreider te specificeren.

Social Network Connectivity Interface

Activiteiten	Resulterend product	Startdatum	Einddatum
Werkplek inrichten	-	15-11-2010	-
Onderzoek verrichten	De mogelijkheden m.b.t. OpenID, Facebook connect, Twitter etc.	15-11-2010	03-12-2010
Plan van aanpak opstellen	Plan van aanpak	06-12-2010	10-12-2010
Inception fase	Inceptionrapport Prototype	13-12-2010	17-12-2010
Elaboration fase	Elaborationrapport	20-12-2010	28-12-2011
Construction fase	Constructionrapport Code	29-12-2011	14-01-2011

Centrale login

Activiteiten	Resulterend product	Startdatum	Einddatum
Onderzoek verrichten	Mogelijkheden voor een centrale login	17-01-2011	28-01-2011
Plan van aanpak opstellen	Plan van aanpak	31-01-2011	04-02-2011
Inception fase	Inceptionrapport Prototype	07-02-2011	18-02-2011
Elaboration fase	Elaborationrapport	21-02-2011	04-03-2011
Construction fase	Constructionrapport Code	07-03-2011	18-03-2011

4.2.2 Voortgangsgesprekken

Om te voorkomen dat ik uitliep op de planning en de voortang te bewaken heb ik om de één tot twee weken een voortgangsgesprek gehouden met de lead developer. Dit waren meestal korte gesprekken waar ik vertelde waar ik op dat moment mee bezig was. Hierbij bekeek ik of de planning niet in gevaar kwam. Tijdens een voortgangsgesprek kon ik ook dingen bespreken waar ik op dat moment tegenaan liep wat de planning eventueel zou kunnen belemmeren.

5 Technieken

In dit hoofdstuk heb ik beschreven welke technieken ik heb gebruikt. Eerst beschrijf ik van welke software ik gebruik heb gemaakt en vervolgens welke design patterns ik heb gebruikt.

5.1 Software

Symfony Framework

Symfony is een MVC web applicatie framework. Symfony is een open source framework dat volledig in PHP5 ontwikkeld is. Het is gericht op het versnellen van het onderhouden en het ontwikkelen van web applicaties. Het framework is makkelijk te installeren, met als randvoorwaarde dat je een UNIX of Windows systeem hebt met PHP erop. Symfony ondersteunt de meeste beschikbare databases, zoals MySQL, PostgreSQL, Oracle en MSSQL. Het framework streeft naar programmeer principes, zoals "*Keep It Simple Stupid"* (KISS) en "*Don't Repeat Yourself*" (DRY). Het Symfony framework is gratis en gepubliceerd onder MIT license.

Het framework is makkelijk te installeren en te gebruiken en flexiebel genoeg voor de complexere scenario's. Een aantal voorbeelden van belangrijke features die Symfony te bieden heeft zijn:

- Geautomatiseerde formulier validatie.
- Geïntegreerd ORM (Propel of Doctrine) voor het aanspreken van-, en het schrijven naar de database.
- *Output escaping* om corrupte data tegen te gaan afkomstig van hackers.
- Routing en nette URL's voor zoekmachine optimalisatie.
- Cache management wat voor minder dataverkeer zorgt en minder server load.
- Een internationalisatie- en een content lokalisatielaag.
- De mogelijkheid om plugins te installeren.
- Authenticatie en credential facaliteiten.
- Code-genereer tools (bijvoorbeeld voor het genereren van de complete back end met enkele handelingen).

Ik heb gekozen voor het Symfony framework, omdat het huidige Clubs platform hier ook op draait. De nieuwe te maken functies moesten toegevoegd worden aan het Clubs platform.

Wamp

Wamp server is een installatie voor het opzetten van een web server. Wamp bevat de applicaties Apache, MySQL en PHP. Hiermee wordt alles in een keer geïnstalleerd en zijn verdere handelingen van het configureren bijna niet nodig. Ik heb gebruik gemaakt van WAMP tijdens de fase van het ontwikkelen. Ik heb expres niet de nieuwste versie gekozen, maar juist voor de versie met PHP die ook op de productieomgeving draait. Hier heb ik voor gekozen, zodat ik zeker wist dat er geen comptabiliteit problemen zouden ontstaan.

Eclipse

Eclipse is een ontwikkelplatform. Voor het schrijven van de code heb ik gebruik gemaakt van Eclipse voor PHP developers (Helios Release). Eclipse is een open source community, waar de focus ligt op

het bouwen van een uitbreidbaar ontwikkelplatform. Er zijn daarom veel plugins voor Eclipse beschikbaar.

Subclipse

Voor het versie beheer werd er gebruik gemaakt van SVN. Hiervoor heb ik de plugin Subclipse geïnstalleerd. Hiermee kon ik de Clubs repository inzien en er opdrachten op uitvoeren, zoals het aanmaken van een nieuwe “feature branche” en het uitchecken van bestaande branches.

phpMyAdmin

Voor het bekijken en het aanbrengen van wijzigingen in de database heb ik gebruik gemaakt van phpMyAdmin. Dit is een web applicatie voor het beheren van MySQL databases.

Memcache

Memcache is een gratis en open source, geheugen caching systeem. Het is bedoeld voor het optimaliseren van de performance van web applicaties door het verlagen van de database load. Voor de Clubs applicatie wordt dit gebruikt om gegevens die vaak opgevraagd moeten worden te cachen op de memcache server, waardoor de gegevens snel op te vragen zijn en de database server ontlast wordt en niet overbelast raakt. Ik heb memcache voor mijn projecten niet hoeven te gebruiken.

Redmine

Redmine is een project management web applicatie. Alle bugs en features kunnen hier in beheerd worden. De applicatie is geschreven op het Ruby on Rails Framework en is cross-platform en cross-database.

Microsoft Visio

Voor de modellering van diagrammen heb ik gebruik gemaakt van Visio. Ik heb ook naar alternatieve programma’s gekeken zoals Rational Rose en Dia, maar het gebruikersgemak beviel me hiervan niet. In Visio heb ik verschillende soorten diagrammen gemaakt, zoals: use case, sequentiediagrammen en flowcharts.

5.2 Design patterns

Tijdens de opdrachten heb ik gebruik gemaakt van een aantal design patterns. Een aantal design patterns had ik onbewust gebruikt die standaard in het framework verweven zaten. Hieronder een overzicht van een aantal design patterns die ik heb gebruikt met daarbij een korte uitleg:

Active Record

Met de active record pattern kan je door het veranderen van een object de gegevens direct opslaan in de database zonder een query te schrijven. Dit heb ik gedaan door gebruik te maken van Propel. Dit is een open source Object-Relational Mapping (ORM) voor PHP5. Hiermee kan je communiceren met de database met behulp van objecten, met een eenvoudige API voor het opslaan en ophalen van gegevens (zie hoodstuk “8.3 Construction fase – bouwen”).

Adapter pattern

Een adapter pattern wordt gebruikt om de interface van de ene klasse om te zetten in de interface van de andere klasse. Deze oplossing heb ik toegepast bij het implementeren van de Social Media Connectivity Interface (zie hoofdstuk “7.3 Elaboration fase – technisch ontwerp uitwerken”). Ik heb voor deze oplossing gekozen om de uitbreidbaarheid te bevorderen en de complexiteit van het implementeren van andere protocollen te verminderen.

Context Object

De context object pattern biedt een efficiënte en applicatie-transparante manier van het delen van informatie tussen de verschillende lagen in een software systeem. Ik heb hier gebruik van gemaakt bij het synchronisatie filter van de centrale login (zie hoodstuk “8.3 Construction fase – bouwen”).

Intercepting Filter

Intercepting filter is een pattern verantwoordelijk van het injecteren van gedrag voor en/of na de verwerking van een verzoek. Hierdoor is het mogelijk om voor en na het renderen van de pagina bepaalde acties uit te voeren. Ik heb deze design pattern toegepast voor het maken van de synchronisatie filter van de centrale login (zie hoodstuk “8.3 Construction fase – bouwen”).

6 Opstarten van het project

6.1 Plan van aanpak

In de eerste week ben ik begonnen met het maken van een plan van aanpak (zie Bijlage I: Plan van aanpak). Hierin heb ik een plan van aanpak gemaakt voor de gehele afstudeerperiode. Ik heb er voor gekozen om beide opdrachten hierin op te nemen en niet voor beide opdrachten een apart plan van aanpak te maken. Dit heb ik gedaan omdat mij dit makkelijker leek en hierdoor een duidelijk overzicht kon maken van de gehele afstudeerperiode. Ik heb van het plan van aanpak een levend document gemaakt. Eerst heb ik een globaal plan van aanpak gemaakt met gegevens van het bedrijf en de rollen etc. Vervolgens heb ik per opdracht het plan van aanpak aangevuld.

6.2 Inwerkopdrachten

In de eerste twee weken van het project heb ik mij vertrouwd gemaakt met de toen bestaande software van het Clubs systeem door het oplossen van fouten. Hierdoor kon ik kennis opdoen van het Clubs systeem, waardoor ik kon leren wat de mogelijkheden van het systeem waren. Voor het beheren van de problemen werd er gebruik gemaakt van Redmine. In Redmine kreeg ik een overzicht van punten die aan mij toegewezen waren en nog behandeld moesten worden. De problemen bestonden uit opdrachten van het oplossen van “javascript”-errors tot het veranderen van code in het systeem. De issues waren bij mij ingedeeld op volgorde van prioriteit. Hieronder een lijst van de problemen die ik in deze periode heb afgewerkt.

#	Project	Tracker	Prioriteit	Onderwerp
666	Clubs	Change	Normaal	Leden beheer verbeteren
667	Clubs	Change	Normaal	Clubmailing in beheer mailingen in zijn geheel tonen in club mailings
660	Clubs	Change	Normaal	Punten kopen scherm laat iets van tot wederopzegging info button weer. klopt niet en moet er uit.
652	Clubs	Change	Normaal	Flash wordt niet verwijderd uit de cache bij spotlight uit clubkas
643	Clubs	Change	Normaal	Spotlight week, maand, en dag kaart kopen
644	Clubs	Change	Normaal	Links openen in nieuw venster default maken
632	Clubs	Change	Normaal	Wijziging aan club aanmaak pagina
621	Clubs	Change	Normaal	Betaalmappen geven nu een dubbele bevestiging waardoor de conversie mogelijk verminderd.
611	Clubs	Change	Normaal	Nieuwe clubs aanpassingen
601	Clubs	Change	Normaal	Reclame afkopen abonnement rekent niet goed af
604	Clubs	Change	Normaal	Max reminder levels aanpassen Credit management Buckaroo
596	Clubs	Change	Normaal	Module banners afkopen op clubhome
470	Clubs	Bug	Normaal	Banner abonnement fixen
167	Clubs	Change	Normaal	Bug in ledenbeheer
86	Clubs	Feature	Normaal	Categorie club wijzigen door operator
61	Clubs	Feature	Normaal	Foto misbruikknop maken
58	Clubs	Feature	Normaal	Club overdragen aan andere eigenaar mogelijkheid maken

7 Deelproject I: Social Media Connectivity Interface

De eerste deelopdracht was het realiseren van de Social Media Connectivity Interface. Dit is een interface waar identity providers, zoals Hyves, Facebook en Google op aangesloten kunnen worden om mee in te kunnen loggen op Clubs.nl.

Voordat ik met het project begon, ging ik er vanuit dat OpenID een open standaard was die door iedere identity provider gebruikt werd. Later tijdens het onderzoek kwam ik erachter dat dit niet zo was, maar dat bijna iedere identity provider gebruik maakt van een ander protocol.

7.1 Protocollen

7.1.1 OpenID

OpenID is een gedecentraliseerd authenticatiemechanisme, dat het mogelijk maakt om op verschillende websites te kunnen inloggen zonder je gebruikersnaam en wachtwoord (opnieuw) te hoeven invullen. OpenID is een open standaard met vastgestelde specificaties[6].

Gebruikers op het internet moeten meestal voor elke website zich opnieuw registreren om bijvoorbeeld een reactie te mogen plaatsen op een blog bericht. Het elke keer moeten registreren gaat vaak ten koste van het gebruikersgemak. Voor de eindgebruiker kan dit een obstakel zijn en een te hoge drempel, waardoor mensen misschien afhaken om een reactie te plaatsen. Door het gebruik van OpenID behoeft een gebruiker niet steeds in te loggen. Daardoor wordt de kans vergroot dat een gebruiker een reactie zal plaatsen.

OpenID providers

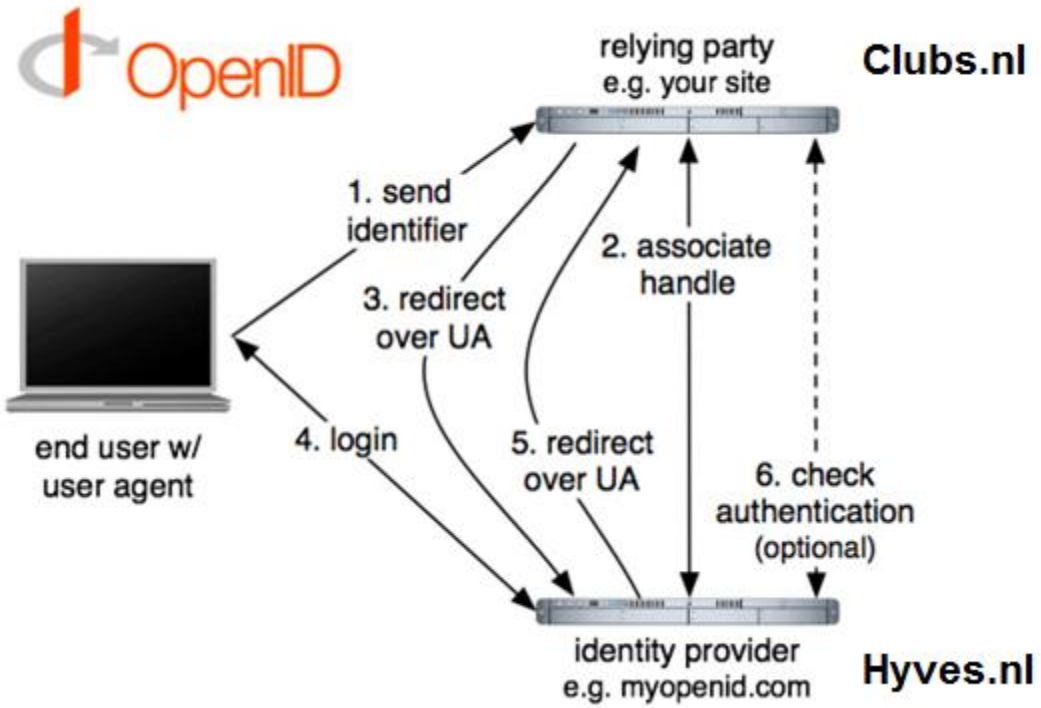
Veel grote websites bieden OpenID al aan[7]. Hieronder een lijst van enkele voorbeelden van grote OpenID aanbieders: Google (Google Account), Yahoo, Hyves, Steam (voor gamers), MySpace, WordPress, Blogger en Orange.

Veel gebruikers die lid zijn van één of meerdere van de bovenstaande websites, zijn er waarschijnlijk/misschien nog niet van bewust wat OpenID is en dat zij een OpenID hebben. Er zijn ook OpenID providers die alleen OpenID aanbieden als dienst, zoals myopenid.com.

[6] http://openid.net/developers/specs/
[7] http://en.wikipedia.org/wiki/List_of_OpenID_providers

Werking van OpenID

Zie Figuur 5 voor een technische weergave van de procedure van OpenID met daaronder stap voor stap een korte uitleg.



Figuur 5 Werking van OpenID

1. Send identifier

Bij de eerste stap stuurt de gebruiker zijn identifier (bijvoorbeeld die van hyves.nl) naar de replying party (bijvoorbeeld clubs.nl).

2. Associate handle

De replying party doet een associate handle (paragraaf 8.2.1 van de specificaties[8]) aanvraag aan de identity provider. De twee partijen stellen een geheime code vast met behulp van de Diffie-Hellman (DH) key exchange[9]. De DH key exchange is ervoor om Man-In-The-Middle attacks[10] tegen te gaan. De identity provider geeft een antwoord terug met de associate handle aan de replying party. De associate handle is een geheime code tussen de replying party en de identity provider voor volgende aanvragen.

3. Redirect over UA

Er wordt een redirect uitgevoerd van de replying party naar de identity provider. Tijdens deze redirect worden de gegevens opgevraagd van de gebruiker aan de identity provider. Er zijn twee type

[8] http://openid.net/specs/openid-authentication-2_0.html
[9] http://en.wikipedia.org/wiki/Diffie_hellman
[10] https://sites.google.com/site/openidreview/issues

modussen om gegevens[11] op te vragen. De eerste heet SREG (simple registration). Hiermee zijn een aantal simpele gegevens mee op te halen om iemand te registreren. De tweede heet AX (attribute exchange). Dit is een uitgebreidere modus waarbij het mogelijk is om meer gegevens op te vragen.

4. Login

De gebruiker moet hier zijn gebruikersnaam en wachtwoord invullen om in te loggen. De gebruiker moet daarnaast ook toestemming hebben gegeven aan de replying party om zijn gegevens te delen.

Indien de gebruiker is ingelogd op de identity provider en al een keer eerder toestemming heeft gegeven aan de replying party en deze keuze heeft opgeslagen, dan wordt de gebruiker direct terugverwezen naar de replying party, zonder hier opnieuw toestemming voor te moeten geven.

5. Redirect over UA

Er wordt een redirect uitgevoerd van de identity provider naar de replying party. Hier worden de gegevens die bij stap 3 zijn opgevraagd terug gegeven, mits hier toestemming voor gegeven is door de gebruiker.

6. Check authentication (optional)

De replying party vraagt aan de identity provider aan de hand van de assoc handle en de signature of de authenticatie valide is. Deze stap is niet verplicht om uit te voeren, maar wel aangeraden als het gaat om betrouwbare data.

Beveiliging

Er zijn een aantal beveiligingen ingebouwd tegen kwaadwilligen:

- Ten eerste de assoc handle die tijdens stap 2 wordt gegenereerd. Dit is de geheime code tussen de replying party en de identity provider. Zonder deze geheime code te weten kan er geen geldig request worden gedaan.
- Ten tweede wordt er een signature gemaakt bij stap 5 over alle attributen en waardes die worden teruggegeven. Met deze signature kan de authenticatie worden geverifieerd bij stap 6. Hierdoor is het niet mogelijk om bijvoorbeeld de identifier te veranderen om in te loggen als iemand anders, omdat dan de signature niet klopt.

Risico's

Er is een aantal risico's met het inloggen met een OpenID. Op het gebied van beveiliging zijn er geen risico's, mits er geen beveiligingslekken zijn aan de kant van de identity provider. Er zijn wel andere risico's te bedenken voor de website van Clubs waar aan gedacht moet worden.

Een voorbeeld van een risico is dat een gebruiker die zich misdraagt, zich snel opnieuw kan aanmelden via een ander identiteit (via een andere OpenID provider). Een oplossing hiervoor kan zijn om alleen een beperkt aantal "trusted" identity providers te accepteren. De gebruiker zou zich dan alsnog volgens de normale manier opnieuw kunnen registreren. Om dit probleem op te lossen zou er een soort van "zwarte-lijst" moeten komen.

[11] http://www.axschema.org/types/

Een ander risico is bijvoorbeeld dat de eindgebruiker afhankelijk wordt van een 3e partij om in te loggen. Wanneer bijvoorbeeld Hyves niet bereikbaar is, kan de gebruiker niet inloggen op Clubs.

7.1.2 Facebook Connect

Facebook maakt gebruik van zijn eigen protocol Facebook Connect[12]. In plaats van het gebruiken van de open standaard OpenID brengt Facebook een nieuw protocol uit. De reden van de ontwikkeling van Facebook connect is omdat Facebook vond dat protocollen als OpenID en oAuth te beperkt waren met het aanbieden van de informatie. Facebook Connect maakt gebruik van zijn Graph API.

7.1.3 Windows Live ID

Windows Live ID maakt ook gebruik van zijn eigen protocol Web Authentication en Delegated Authentication. Windows Live ID maakt gebruik van een REST API. Op 27 oktober 2008 kondigde Microsoft aan[13] dat het de open standaard OpenID zou gaan ondersteunen, waarbij Windows Live ID een OpenID Provider zou worden. Echter is er geen update gekomen[14] van Microsoft over de geplande implementatie sinds augustus 2009.

7.1.4 oAuth

OAuth is een open protocol dat gebruikers in staat stelt om applicaties goedkeuring te geven namens hen te handelen zonder het delen van wachtwoorden[15]. Het protocol OAuth wordt onder andere door Twitter gebruikt.

7.2 Inception fase – eisen vaststellen

Bij de aanvang van het eerste deelproject heb ik eerst een onderzoek gedaan tijdens de inceptionfase. Ik heb hierbij onderzoek gedaan naar de werking en de mogelijkheden van OpenID (zie Bijlage II: Social Media Connectitvity Interface). De stappen van de werking van het inloggen heb ik gevisualiseerd in afbeeldingen en opgenomen in het inception rapport.

7.2.1 Functionele eisen

De functionele eisen heb ik opgesteld na het uitvoeren van het onderzoek naar OpenID. Hier heb ik de initiële functionele eisen opgesteld en opgenomen in het inceptionrapport. Een van de eerste eisen waren de identity providers die Clubs geïmplementeerd wilde hebben. Deze identity providers waren: Hyves, Facebook, Google, Windows Live en Twitter.

De functionele eisen heb ik vervolgens uitgewerkt in use cases. Zie Figuur 6 voor een belangrijke use case van nieuwe gebruikers die nog geen lid zijn.

---

[12] http://developers.facebook.com/blog/post/108
[13] http://winliveid.spaces.live.com/blog/cns!AEE1BB0D86E23AAC!1745.entry
[14] http://blogs.technet.com/b/privacyimperative/archive/2009/08/28/windows-live-id-openid-status-update.aspx
[15] http://thenextweb.com/socialmedia/2010/11/04/facebook-connect-oauth-and-openid-the-differences-and-the-future/

7.2.2 Prototype

Aan het einde van de inception fase heb ik een prototype ontwikkeld, waarin ik een simulatie heb gemaakt van het registreren vanuit mijn test-situatie via Hyves, Google, Facebook, Windows Live en Twitter. Dit heb ik gedaan om de opgestelde functionele eisen te testen. Om dit te testen heb ik van mijn lokale testomgeving een primitieve simulatie omgeving van Clubs gemaakt met enkele buttons van de hierboven genoemde websites, om hiermee in te loggen.

Ik heb naar de identity providers gekeken van welk protocol zij gebruik maakten. Ik heb voor elk protocol een geschikte library gezocht. Ik heb deze protocollen met elkaar vergeleken en gekeken naar de library die mij het meeste aansprak. Voor OpenID had ik voor een library gekozen die veel werd aangeraden op verschillende forums. Voor Facebook, Twitter en Windows Live heb ik een library gebruikt die door hun zelf werd aangeboden. Deze werden ook aangeraden op forums en het leek mij het meest voor de hand liggend om deze te gebruiken.

Libraries

Website:	Protocol:	Library:
Hyves	OpenID	JanRain OpenID Library[16]
Facebook	Facebook Connect	Facebook SDK[17]
Google	OpenID	JanRain OpenID Library
Windows Live	Web Authentication en Delegated Authentication	WebAuth[18]
Twitter	oAuth	Abraham Williams TwitteroAuth[19]

Bij de implementatie van Windows Live had ik veel tijd nodig. Er was van de Windows Live API weinig documentatie en voorbeelden te vinden, waardoor ik erg veel tijd nodig had voor onderzoek. Ik heb toen gezocht op Google Project hosting[20] en een voorbeeld gevonden. Dit voorbeeld heb ik bestudeerd en gebruik van gemaakt.

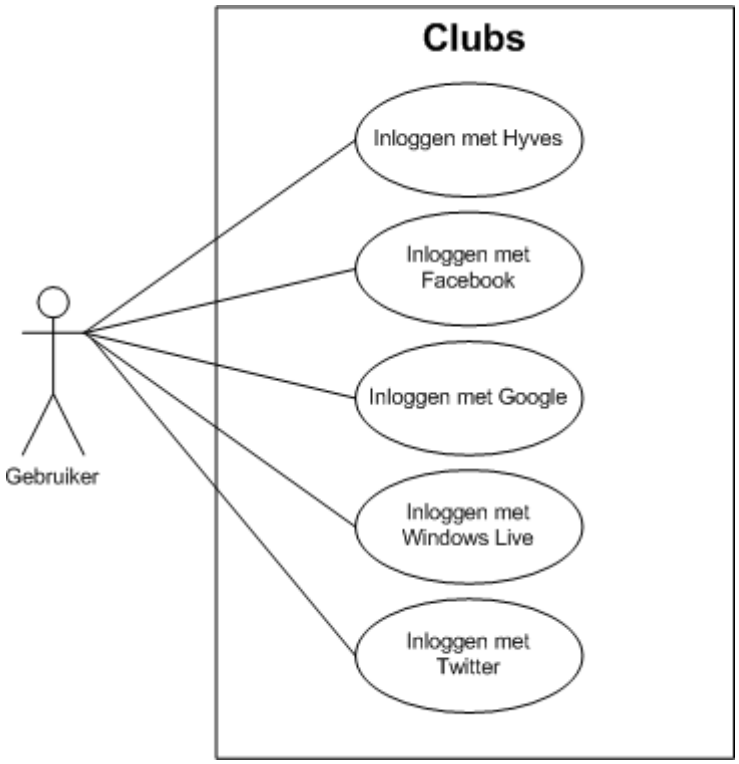
[16] http://www.janrain.com/openid-enabled
[17] https://github.com/facebook/php-sdk/
[18] http://www.microsoft.com/downloads/en/details.aspx?FamilyId=E565FC92-D5F6-4F5F-8713-4DD1C90DE19F
[19] https://github.com/abraham/twitteroauth
[20] http://code.google.com/hosting/

7.3 Elaboration fase – technisch ontwerp uitwerken

Tijdens de elaboration fase heb ik het functioneel ontwerp van de inception fase verder uitgewerkt tot een technisch ontwerp. Ik heb ervoor gekozen om de use cases tijdens de elaboration fase verder uit te breiden, omdat ik bij deze opdracht afhankelijk was van derde partijen en nog niet alles van de technische mogelijkheden afwist.

Hieronder een belangrijke use case waarin wordt beschreven hoe een nieuwe gebruiker die nog geen lid is zich inlogt/registreert op Clubs met een identitity provider. De gebruiker logt in vanuit Clubs.

Nieuwe gebruiker die nog geen lid is (Use case)



Figuur 6 Use case nieuwe gebruikers die nog geen lid zijn

Naam:	**Inloggen op Clubs via een identity provider voor een nieuwe gebruiker**
Samenvatting:	De gebruiker kan zonder het invullen van een registratie forumlier inloggen op Clubs, waarmee hij direct lid wordt.
Actoren:	Gebruiker
Aannamen:	De gebruiker is op de Clubs website en is nog geen lid. De gebruiker wil inloggen met Hyves. De gebruiker is lid van Hyves.
Beschrijving:	1. De gebruiker klikt op een button van inloggen met Hyves. 2. Clubs verwijst de gebruiker naar Hyves met een aanvraag om gegevens te delen van deze gebruikers. De gegevens die worden opgevraagd zijn: naam, geboortedatum, geslacht en het e-mail adres. 3. De gebruiker geeft de website van Clubs toestemming om zijn gegevens te delen met Clubs. **[Gebruiker is niet ingelogd]** 4. De gebruiker komt terug van Hyves op de Clubs website met de gegevens die zijn opgevraagd bij de verwijzing naar Hyves. **[Gebruiker heeft geen toestemming gegeven]** De gebruiker komt terecht op een welkom scherm waar hij kan aangeven of hij een nieuwe gebruiker is op Clubs of al een bestaand account heeft en deze wil koppelen met zijn

	Hyves account. De gebruiker vinkt aan dat hij het eens is met de algemene voorwaarden en drukt op “Ga verder” onder het gedeelte voor de gebruikers die nog geen lid zijn. **[Geen e-mail adres]** 5. Er wordt een account aangemaakt voor de gebruiker, waar hij meteen mee is ingelogd. **[E-mail adres is niet geverifieerd]**
Uitzonderingen:	• **Gebruiker is niet ingelogd:** Wanneer de gebruiker nog niet is ingelogd op Hyves, moet hij eerst zijn gebruikersnaam en wachtwoord invullen om in te loggen op Hyves. • **Gebruiker heeft geen toestemming gegeven:** De gebruiker moet toestemming geven op Hyves om zijn gegevens te delen met Clubs. Wanneer de gebruiker weigert om zijn gegevens te delen zal er een melding komen dat het inloggen met Hyves is mislukt. • **Geen e-mail adres:** Niet alle websites geven een e-mail adres terug. Hiervoor komt een extra invul veld waar de gebruiker alsnog zijn e-mail adres kan invullen. Het invullen van een e-mail adres is verplicht. • **E-mail adres niet geverifieerd:** Wanneer er geen e-mail adres is gegeven en de gebruiker dit alsnog heeft moeten invullen, zal het e-mail adres eerst geverifieerd moeten worden. Een gebruiker mag niet worden ingelogd voordat het e-mail adres is geverifieerd. De gebruiker zal een e-mail moeten krijgen waarmee hij dit alsnog kan doen.
Resultaat:	De gebruiker is lid geworden op Clubs. Het Hyves account is gekoppeld met het Clubs account van de gebruiker.

E-mail verificatie

Een probleem waar ik tegenaan liep was het ophalen van het e-mail adres. Het ophalen van een e-mail aders was niet (zomaar) mogelijk bij elke identity provider. Een voorwaarde dat de mensen mogen inloggen op Clubs was dat het e-mail adres geverifieerd moest zijn. Wanneer het e-mail adres afkomstig is van een identity provider zoals bijvoorbeeld Google, dan is deze al geverifieerd bij Google.

De oplossing voor de identity providers die geen e-mail adres terug geven was dat de gebruiker deze alsnog in moet vullen op de Clubs website. Vervolgens krijgt de gebruiker dan een e-mail om het e-mail adres te verifiëren. Het nadeel hiervan is wel dat dit voor extra handelingen zorgt, waarbij de gebruiker alsnog kan afhaken.

Opvraagbare gegevens per identity provider

Hieronder een overzicht van een gedeelte van de informatie die kan worden opgevraagd per identity provider:

Naam:	✓	✓	✓	✓	✓
E-mail adres:	✓	✓ *	✓	✓	✗
Nickname:	✓	✓	✓	✓	✓
Foto:	✓	✓	✗	✗	✓
Profiel URL:	✓	✓	✗	✓	✓
Geboortedatum:	✓	✓	✗	✓	✗
Geslacht:	✓	✓	✗	✗	✗
Locatie:	✓	✓	✓	✗	✓

* Wanneer de applicatie die geregistreerd staat bij Facebook nog niet is goedgekeurd, wordt er bij het opvragen van een e-mail adres standaard een proxy e-mail adres geselecteerd bij het scherm waar de gebruiker kan kiezen of hij zijn gegevens wil delen met Clubs. Dit is waarschijnlijk om spam te voorkomen en om gebruikers in bescherming te nemen tegen het uitgeven van "*echte*" e-mail adressen aan elke willekeurige applicatie. Een proxy e-mail adres is een e-mail adres van ongeveer 100 tekens lang. Een voorbeeld van een proxy e-mail adres is:

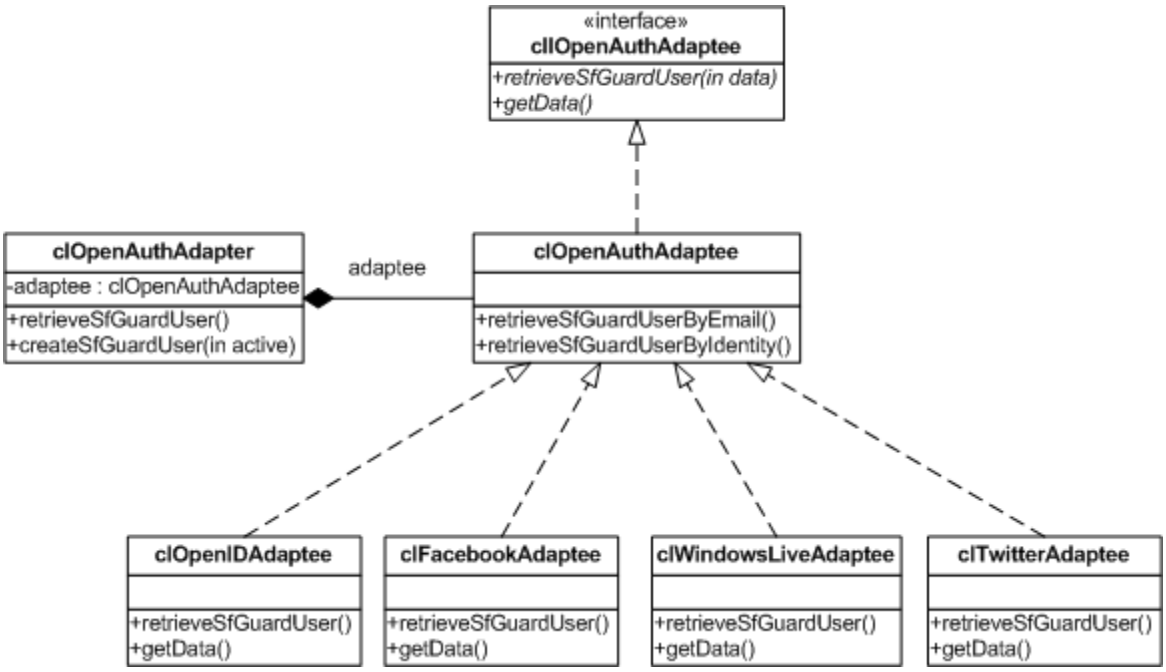
> *apps+161103497255323.100000522804584.2966cd30251e7536f96f1f03291071 54@proxymail.facebook.com*

De gebruiker is wel met een proxy e-mail adres direct te benaderen.

7.3.1 Design patterns

Adapter pattern

Een niet-functionele eis was dat het systeem makkelijk uitbreidbaar moest zijn. Wanneer er een nieuw protocol gekoppeld moet worden aan het Social Media Connectivity Interface, dan moet dit kunnen zonder dat de hele code omgegooid moet worden. Ik heb hiervoor gekeken naar een geschikte oplossing om dit mogelijk te maken. Ik heb toen gekeken naar de design patterns. Ik heb het probleem vergeleken met de design patterns en kwam toen uit op de adapter design pattern.



Figuur 7 Adapter pattern

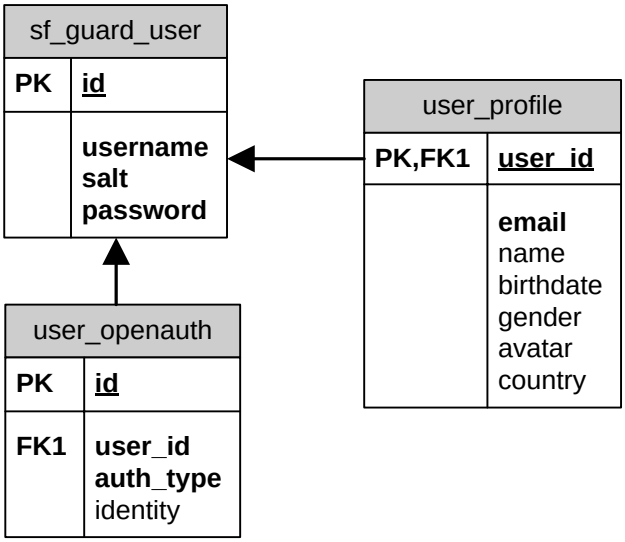
clOpenAuthAdapter	De object adapter heeft twee methodes die de specifieke methodes van de adaptee klasses aanroepen.
clOpenAuthAdaptee	De adaptee klasse die door de adapter klasse moet worden adapt. De adaptee klasse wordt extend door de specifieke identity provider adaptee klasses.
clIOpenAuthAdaptee	Een interface waaraan een adaptee klasse moet voldoen.

Elke identity provider geeft zijn gegevens terug op een andere manier. Voor elke identity provider heb ik een aparte adaptee klasse gemaakt die het interface kan aanspreken.

In eerste instantie was dit meer werk om het te implementeren, maar achteraf bespaart dit veel tijd bij het implementeren van andere identity providers. De reden van het implementeren van de adapter pattern was het bieden van de mogelijkheid om andere identity providers te kunnen implementeren met relatief weinig handelingen zonder de bestaande code te behoeven omgooien.

7.3.2 Databasemodel

Voor de koppeling van een account van een identity provider aan een account van Clubs heb ik een uitbreiding gemaakt op het databasemodel. De uitbreiding is de tabel user_openauth, waarin de identity van een bepaalde identity provider wordt gekoppeld aan een gebruiker van Clubs. Het is mogelijk om meerder identity providers te koppelen aan een gebruiker.



Figuur 8 Databasemodel

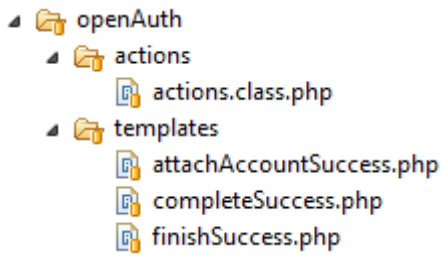
Bij het inloggen van de gebruiker via een identity provider wordt in de database opgevraagd aan de hand van de identifier of deze is gekoppeld aan een gebruiker van Clubs.

7.4 Construction fase – bouwen

Het uitgangspunt bij de start van de construction fase was het prototype wat ik al eerder tijdens het project had gemaakt. Voor het project heb ik een nieuw module gemaakt wat ik de naam “openauth” heb gegeven. Het zijn “open” protocollen die met **aut**hentiecatie te maken hebben.

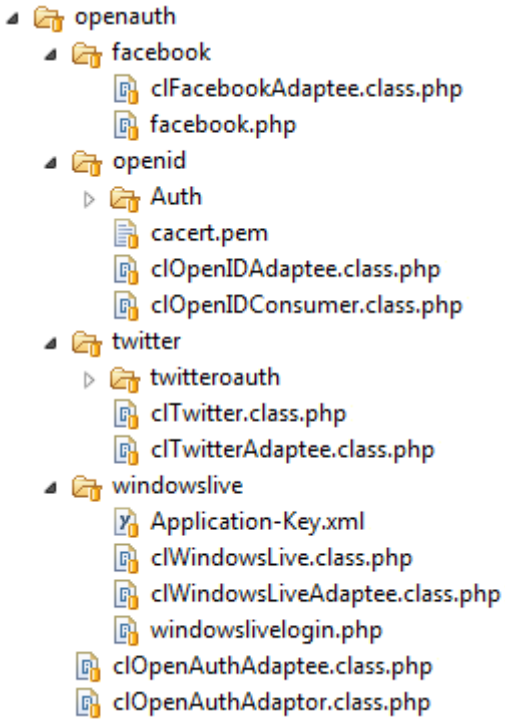
Hieronder de mappenstructuur met de bestanden die erbij gekomen zijn met het interface.

/apps/frontend/modules/openAuth



Figuur 9

/lib/openauth



Figuur 10

7.4.1 Resultaat

Hieronder een voorbeeld van het resultaat op de voorpagina. Rechts onder “Welkom bij Clubs” en rechts bovenaan naast “Wachtwoord vergeten?” staan nu 5 buttons van identity providers waarmee ingelogd kan worden.

Figuur 11 Resultaat Social Media Connectifity Interface

7.5 Transition fase – project afronden

In de transition fase heb ik het project afgerond. Het oplossen van fouten was niet nodig. Er waren geen meldingen binnen gekomen van fouten. Verder heb ik instructies gegeven van de stappen die nodig zijn voor de deployment.

8 Deelproject II: Centrale login

De tweede opdracht was het realiseren van de centrale login. De centrale login is ervoor om Federated identity mogelijk te maken. Federated identity[21] maakt authenticatie van de gebruiker mogelijk die over meerdere systemen gaat.

Bij het overschakelen van Clubs-website A naar Clubs-website B moet er nu opnieuw worden ingelogd. Door de nieuwe samenwerkingsverbanden in de nabije toekomst zal het aantal domeinen fors toenemen. Met de centrale login moet het mogelijk zijn dat de gebruiker zich slechts één keer hoeft in te loggen (Single sign-on) en dat hij bij het overschakelen naar een andere Clubs-website op een andere domeinnaam ingelogd blijft en niet opnieuw hoeft in te loggen. Daarnaast moet het ook mogelijk zijn om op elke willekeurige Clubs website te kunnen uitloggen (Single sign-out).

8.1 Single sign-on

De definitie van single sign-on is dat een gebruiker met één dezelfde identiteit, met één wachtwoord, dat hij eenmaal moet invoeren, toegang heeft om te schakelen van website A naar website B zonder een onderbreking.

8.2 Single Sign-out

De definitie van signle sign-out is dat een gebruiker zich maar op één plek hoeft uit te loggen, waarna de gebruiker op andere systemen automatisch ook is uitgelogd.

8.3 Functionele eisen

Single sign-on

- De gebruiker hoeft zich maar één keer te authentificeren met een gebruikersnaam en wachtwoord. Bij het schakelen tussen website A en website B mag de gebruiker niet worden onderbroken om zich opnieuw te moeten authentificeren, maar moet hierbij meteen ingelogd zijn zonder hiervan iets te merken.
- De gebruiker moet via meer dan één systeem ingelogd kunnen zijn. Bijvoorbeeld de ene sessie vanaf zijn PC en de andere sessie vanaf zijn laptop. Wanneer de gebruiker Clubs vanaf een ander systeem (pc, laptop) benadert, moet ervoor een nieuwe sessie voor worden aangemaakt.
- In sommige gevallen moet er een uitzondering worden gemaakt bij het kijken of de gebruiker al is ingelogd. De uitzonderingssituaties doen zich voor wanneer:
    - de gebruiker een bot is (bijvoorbeeld: crawlers).
    - het om push berichten gaat van bijvoorbeeld de payment service provider.

Single sign-out

- Bij het uitloggen van de gebruiker op website B, moet de gebruiker ook uitgelogd zijn op website A.
- Bij het uitloggen van de gebruiker moet er rekening mee worden gehouden dat de gebruiker op een andere locatie nog ingelogd kan zijn en dat de gebruiker alleen op de juiste systemen wordt uitgelogd.
- Het moet mogelijk zijn voor de gebruiker om uit te kunnen loggen op andere locaties.

[21] http://en.wikipedia.org/wiki/Federated_identity

8.4 Bestaande oplossingen

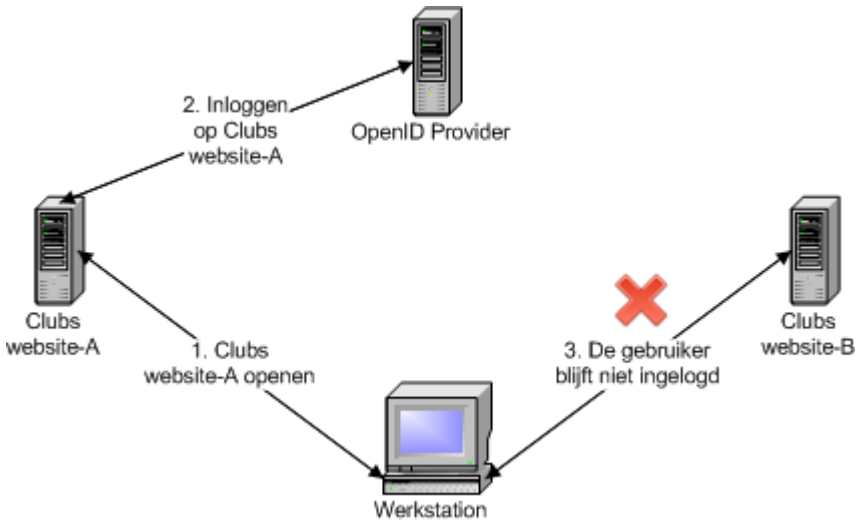
Ik ben eerst gaan kijken naar bestaande oplossingen die Federated identity mogelijk maken. Hierbij kwam ik drie oplossingen tegen:

- OpenID
- oAuth
- Central Authentication Service (CAS)

8.4.1 OpenID

De eerste oplossing die ik tegen ben gekomen was OpenID. Bij de eerste opdracht ben ik ook bezig geweest met OpenID, maar dan aan de kant van de replying party. Deze opdracht zal vooral gaan over de kant van de provider.

Een OpenID provider is een gedecentraliseerd authenticatiesysteem. Het is hierdoor niet mogelijk om ingelogd te blijven over verschillende domeinen. De reden waarom dit niet mogelijk is, is omdat bij een gedecentraliseerd authenticatiesysteem het voor website A niet mogelijk is om bij website B op te vragen wat de gebruiker doet. In Figuur 12 is te zien waar het fout gaat.



Figuur 12 OpenID Provider

Ook is het niet mogelijk om in één keer uit te loggen op meerdere sites (Single sign-out).

Met de hierboven genoemde punten voldeed OpenID dus niet aan de functionele eisen.

8.4.2 oAuth

De tweede oplossing die ik ben tegen gekomen was oAuth. Bij het onderzoeken van oAuth kwam ik erachter dat oAuth geen alternatief is van OpenID, maar voor iets anders bedoeld is[22] en zelfs samen in combinatie gebruikt wordt. Het verschil met OpenID is dat oAuth vooral met autorisatie[23] (rechten – toestemming geven voor het delen van data tussen applicaties zonder het uitwisselen van wachtwoorden) te maken heeft en OpenID met authenticatie (inloggen). De keuze voor het protocol oAuth viel meteen af, omdat het niet nodig was om data uit te wisselen tussen andere applicaties.

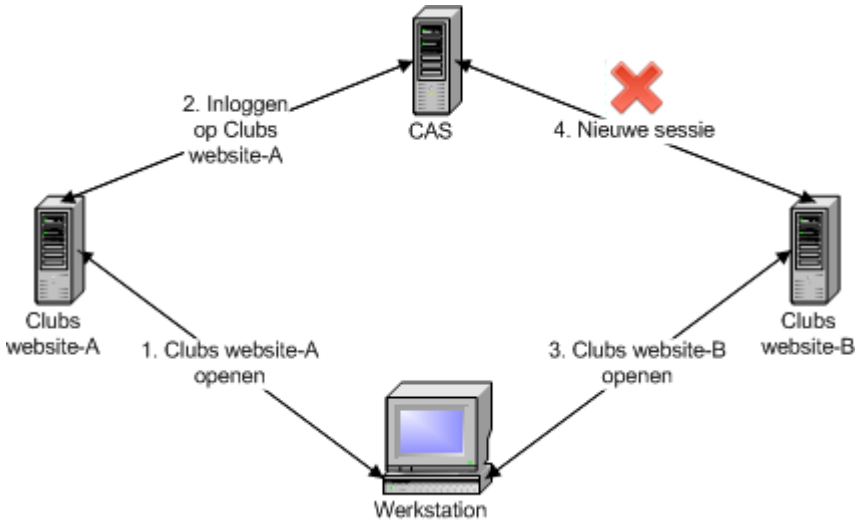
[22] http://cakebaker.42dh.com/2008/04/01/openid-versus-oauth-from-the-users-perspective/
[23] http://softwareas.com/oauth-openid-youre-barking-up-the-wrong-tree-if-you-think-theyre-the-same-thing

8.4.3 Central Authentication Service

De derde oplossing die ik tegen ben gekomen was Central Authentication Service. Dit is een authenticatie systeem gemaakt door de universiteit Yale en wat nu een project is van Jasig . CAS is gemaakt voor authenticatie over het web. CAS is een open protocol met een open-source Java server component.

In tegenstelling tot de eerder genoemde oplossingen voldoet CAS wel aan de functionele eisen. Een groot nadeel van CAS is echter dat het geen gebruik maakt van gedeelde sessies. Dit betekent dat een gebruiker voor elk bezoek aan een andere Clubs-website een nieuwe sessie krijgt. Dit is niet optimaal voor de performance van de Clubs applicatie. Hieronder een korte uitleg waarom dat zo is:

- Het is niet mogelijk om bepaalde gegevens, zoals rechten informatie en pagina's die je recentelijk bezocht hebt te delen bij het overschakelen van Clubs-website A naar Clubs-website B, omdat de gegevens niet gedeeld zijn. Hierdoor moet een niet-ingelogde gebruiker bij elk request van een pagina aan de CAS server vragen of hij in de tussentijd is ingelogd geweest op één van de andere Clubs-websites.
- Voor de Signle sign-out[24] is een niet gedeelde sessie ook minder handig, omdat hierdoor gewerkt moet worden met een time-out of er moet op elke site apart worden bijgehouden dat de gebruiker moet worden uitgelogd.



Figuur 13 CAS

Een ander nadeel van CAS is dat het server component een Java applicatie is die geïnstalleerd (en onderhouden) moet worden. Hier brengt de applicatie beheerder IC&S erg veel kosten voor in rekening.

[24] https://wiki.jasig.org/display/CAS/Single+Sign+Out

8.5 Inception fase – eisen vaststellen

Bij de inception fase van het tweede deelproject ben ik begonnen met het zoeken naar bestaande oplossingen die Federated identity over het web mogelijk maken (zie Bijlage III: Centrale Login).

Functionele eisen

In het inception rapport heb ik eerst de functionele eisen opgesteld. Een van de belangrijkste functionele eisen was het mogelijk maken van Single sign-on op de Clubs websites. Ik heb op basis van deze functionele eisen enkele use case diagrammen gemaakt. Er waren niet veel use case diagrammen nodig, aangezien het werk van de centrale login bijna alleen maar op de achtergrond gebeurt en er weinig interactie met de gebruiker is.

Oplossing

Na het onderzoek heb ik de op internet gevonden oplossingen met elkaar vergeleken en de voor- en nadelen van die oplossingen onder elkaar gezet. Ik heb het resultaat met de lead developer besproken.

In overleg met de opdrachtgever en lead developer is er besloten om een eigen implementatie te ontwikkelen die Federated identity mogelijk maakt op Clubs. De hoofdreden om niet voor CAS te kiezen was vooral omdat het niet gebruik maakt van gedeelde sessies, wat dus nadelig is voor de performance. Voor de eigen implementatie heb ik mij laten inspireren door andere oplossingen[25][26] van mensen op het internet.

[25] http://stackoverflow.com/questions/342378/cross-domain-login-how-to-login-a-user-automatically-when-transfered-from-one-d/342498#342498

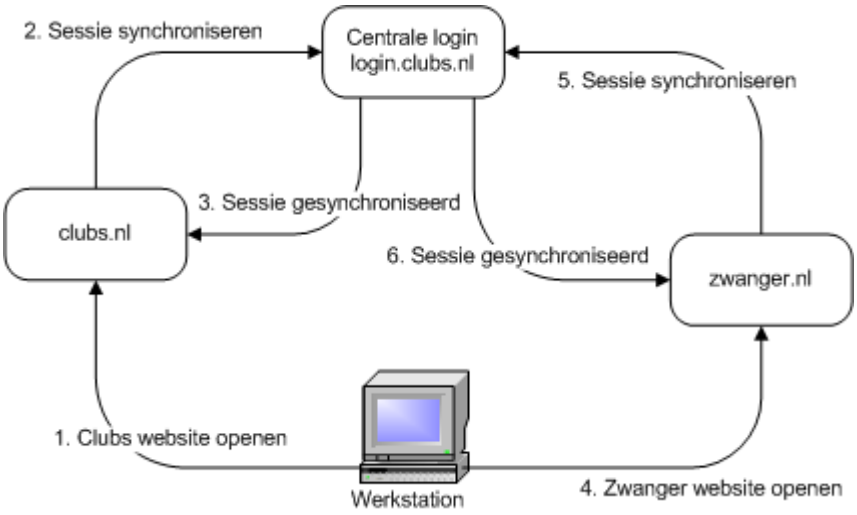
[26] http://www.jasny.net/articles/simple-single-sign-on-for-php/

Prototype

Aan het einde van de inception fase heb ik een prototype ontwikkeld die de werking van de centrale login demonstreerde. Met de proof of concept waren de volgende functionaliteiten getest:

- Schakelen tussen website A en website B met verschillende domeinnamen, zonder opnieuw in te hoeven loggen. (Single sign-on)
- Bij het uitloggen op website B wordt de gebruiker ook uitlogt op website A indien de gebruiker daar aanwezig (d.w.z. ingelogd) is geweest. (Single sign-out)

Met de proof of concept is een eigen implementatie van Federated identity ontwikkeld. De proof of concept bestond uit een op zichzelf staand prototype dat niet in de Clubs omgeving is geïmplementeerd. Hiermee is aangetoond dat het mogelijk was om in te loggen op website A en dat wanneer de gebruiker dan verder door ging naar website B dat het dan voor de gebruiker niet nodig was om opnieuw in te loggen. Dit wordt gedaan door het synchroniseren van de sessie. Zie het "Elaboration rapport" van "Bijlage III: Centrale Login" voor een overzicht van de werking van de centrale login.



Figuur 14 Prototype centrale login

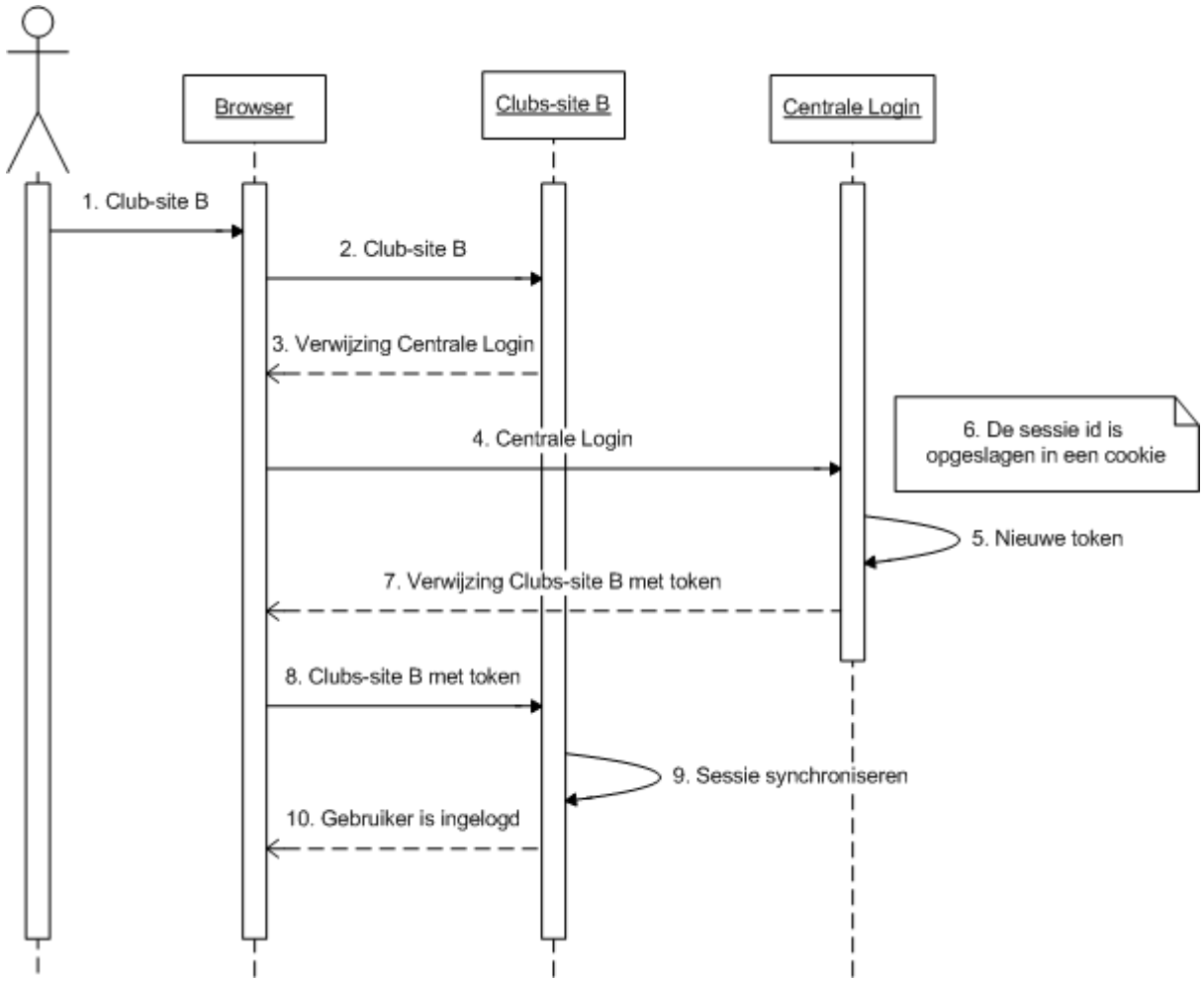
Doordat de sessie gesynchroniseerd wordt met de centrale login zal de gebruiker ingelogd blijven.

8.6 Elaboration fase – technisch ontwerp uitwerken

Tijdens de elaboration fase heb ik het functioneel ontwerp van de inception fase verder uitgewerkt tot een technisch ontwerp. In dit hoofdstuk worden een aantal technische ontwerpen weergegeven en uitgelegd.

Sessie synchronisatie (sequentiediagram)

In dit sequentiediagram worden de stappen weergegeven die worden uitgevoerd tijdens het synchroniseren van de sessie. In de volgende drie sequentiediagrammen zal de techniek van deze stappen worden beschreven.



Figuur 15

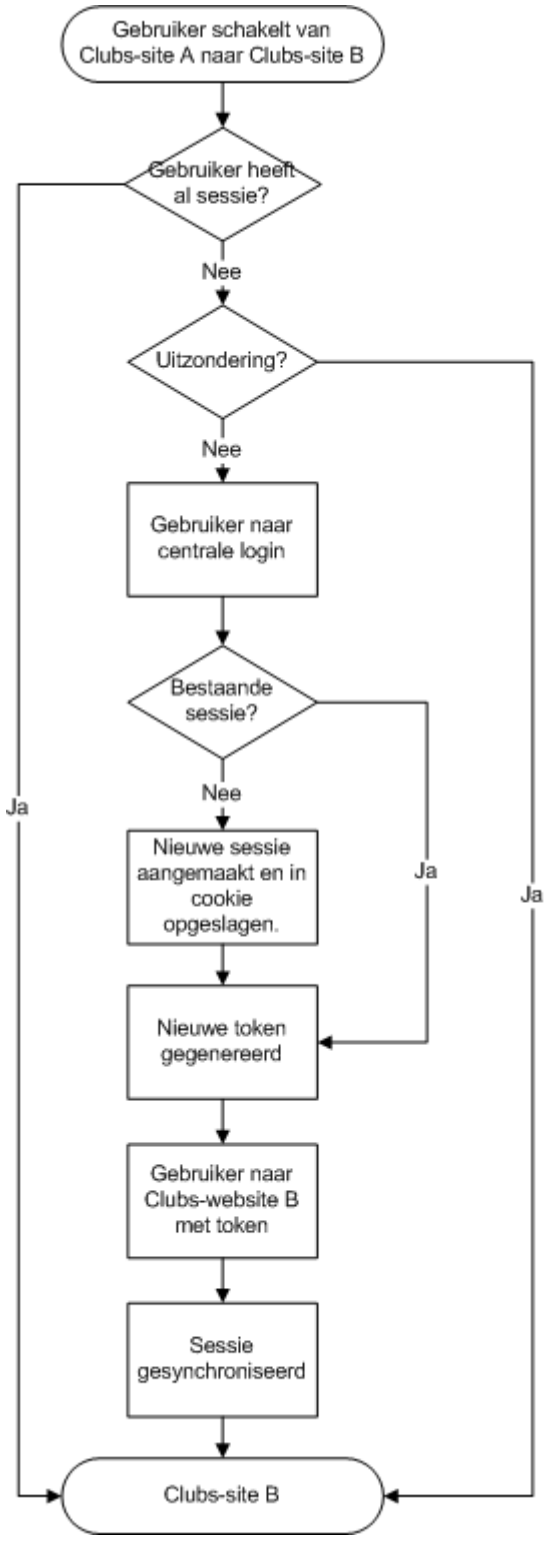
1. De gebruiker opent Clubs-site B in de browser.
2. De browser opent Clubs-site B.
3. Clubs-website B stuurt een verwijzing naar de browser richting de Centrale login.
4. De browser verwijst de gebruiker door naar de Centrale Login.
5. De centrale login maakt nieuwe sessie aan met een token die verwijst naar de sessie id. De token wordt aangemaakt in verband met beveiligingsredenen, zodat de mogelijkheid van het onderscheppen en het overnemen van de sessie id beperkt blijft. Deze risicofactoor wordt beperkt, doordat de token maar één keer gebruikt kan worden en maximaal voor één minuut geldig is. Bij een volgende keer wordt er een nieuwe token gegenereerd die verwijst naar de

sessie id, die is opgeslagen in een cookie. Wanneer de sessie niet meer bestaat zal er een nieuwe sessie worden aangemaakt.

6. De sessie id is opgeslagen in een cookie voor het synchroniseren van de sessie op andere websites. Met dit sessie id kan een token worden gegenereerd die naar dit sessie id verwijst.
7. De centrale Login stuurt een verwijzing naar de browser naar Clubs-website A met de token.
8. De browser gaat naar Clubs-site B met de sessie token.
9. De sessie wordt gesynchroniseerd op Clubs-site A, door de sessie id op te zoeken m.b.v. de token. De token zal vervolgens worden verwijderd en niet meer geldig zijn.
10. Gebruiker is ingelogd op Club-site B.
11. Hieronder volgt een flowchart met de stappen en beslissingen die moeten worden uitgevoerd bij het synchroniseren van de sessie.

Sessie synchronisatie (flowchart)

Zie Figuur 16 voor een flowchart met de stappen en beslissingen die moeten worden uitgevoerd bij het synchroniseren van de sessie.



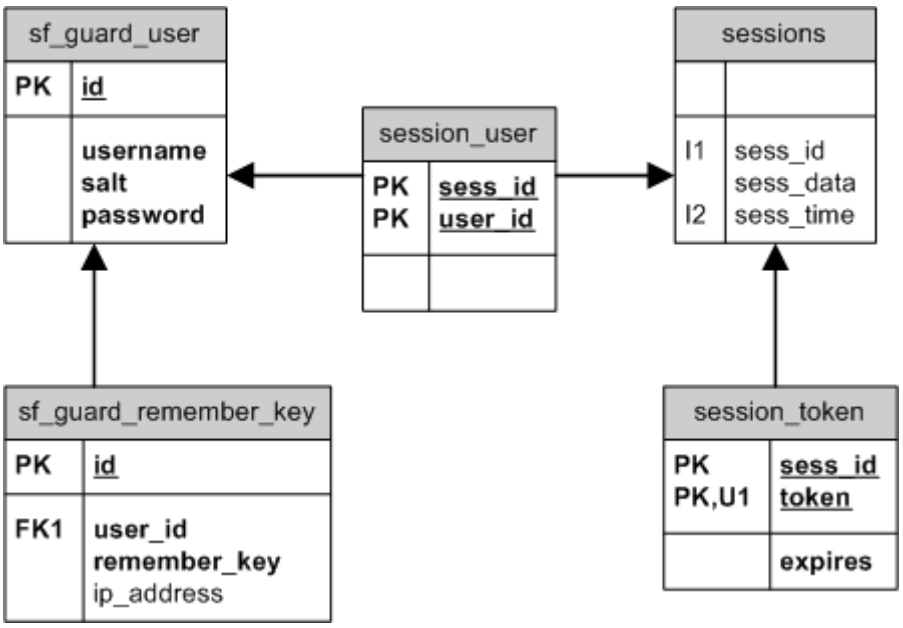
Uitzondering:

- *De gebruiker al gesynchroniseerd is.*
- *De request afkomstig is van een bot (bijv. Google Crawler bot). Dit is niet mogelijk omdat bots geen cookie ondersteuning hebben.*
- *De request een niet-GET request is (redirects zijn met bijv. een POST request niet mogelijk).*
- *De request een AJAX call is (redirects niet mogelijk).*
- *Cookies ondersteuning niet aanstaat.*
- *De request wordt uitgevoerd op hetzelfde domein als van de centrale login.*

Figuur 16 Flowchart sessie synchronisatie

8.6.1 Databasemodel

Hieronder worden de database tabellen weergegeven die betrekking hebben op de opdracht. De tabellen sf_guard_user, sf_guard_remember_key en sessions zijn tabellen die al bestonden. Voor de tabel sf_guard_remember_key was een wijziging nodig. In de oude situatie was de user_id de primaire sleutel. In de nieuwe situatie is hiervoor een extra colom gekomen met een auto increment. Hierdoor is het voor een gebruiker mogelijk om ingelogd te blijven op meerdere locaties.



session_token

De tabel session_token is er nieuw bijgekomen. Dit is een tabel met tokens die verwijzen naar de sessie id. De token bestaat uit een string van 32 tekens lang. De token is een unieke waarde die random gegenereerd wordt. De expires colom houd bij tot wanneer de sessie token geldig is.

session_user

De tabel session_user is er nieuw bijgekomen. In deze tabel wordt bijgehouden welke session id bij welke ingelogde gebruiker hoort. Door deze gegevens op te slaan kan de gebruiker zich op andere locaties uitloggen.

Object-Relational Mapping

In Symfony zit standaard een Object-Relational Mapping (ORM) geïntegreerd. De ORM die Clubs gebruikt is Propel[27] versie 1.5. Deze ORM maakt gebruik van het active record pattern. De ORM kan op basis van tabellen in de database automatisch objecten genereren die manipulaties kunnen uitvoeren op de database. Eerst moesten de tabellen gedefinieerd worden in *schema.yml*.

/config/schema.yml

```
session_token:
  _attributes: { phpName: SessionToken }
  sess_id: { phpName: SessId, type: VARCHAR, size: '64', primaryKey: true, required: true,
foreignTable: sessions, foreignReference: sess_id, onDelete: CASCADE, onUpdate: CASCADE }
  token: { phpName: Token, type: VARCHAR, size: '40', primaryKey: true, required: true }
  expires: { phpName: Expires, type: TIMESTAMP, required: true }
  _uniques: { token: [token] }
session_user:
  _attributes: { phpName: SessionUser }
  sess_id: { phpName: SessId, type: VARCHAR, size: '64', primaryKey: true, required: true,
foreignTable: sessions, foreignReference: sess_id, onDelete: CASCADE, onUpdate: CASCADE }
  user_id: { phpName: UserId, type: SMALLINT, size: '8', primaryKey: true, required: true,
foreignTable: sf_guard_user, foreignReference: id, onDelete: CASCADE, onUpdate: CASCADE }
  _indexes: { user_id: [user_id] }
```

Op basis hiervan kunnen de modellen worden gegenereerd met het volgende commando:

php symfony propel:build-model

Na het uitvoeren van het commando worden een aantal files in */lib/model/* aangemaakt:

- SessionToken.php – model klasse
- SessionTokenPeer.php – peer klasse met methodes voor het ophalen van gegevens uit de database
- SessionTokenQuery.php – query klasse voor het toevoegen van criteria bij het ophalen van objecten uit de database
- SessionUser.php
- SessionUserPeer.php
- SessionUserQuery.php

[27] http://www.propelorm.org/

Hieronder enkele voorbeelden van CRUD statements die op database worden uitgevoerd met de gegenereerde objecten van het Propel ORM:

```php
<?php

// Een nieuwe session token schrijven naar de database.
$sessionToken = new SessionToken();
$sessionToken->setSessId(session_id());
$sessionToken->setExpires(date('Y-m-d H:i:s', strtotime('+1 min')));
$sessionToken->setToken($sessionToken->generateToken());
$sessionToken->save();

// Een session token uitlezen uit de database met de peer klasse
$sessionToken = SessionTokenPeer::retrieveBySessId(session_id());

// Een session token uitlezen uit de database met de query klasse
$sessionToken = SessionTokenQuery::create()
  ->filterBySessId(session_id())
  ->findOne();

// Een session token wijzigen
$sessionToken = SessionTokenPeer::retrieveBySessId(session_id());
$sessionToken->setExpires(date('Y-m-d H:i:s', strtotime('+1 min')));
$sessionToken->save();

// Een session token verwijderen met de peer klasse
$sessionToken = SessionTokenPeer::retrieveBySessId(session_id());
$sessionToken->delete();

// Een session token verwijderen met de query klasse
SessionTokenQuery::create()
  ->filterBySessId(session_id())
  ->delete();

// Custom query met prepared statement (verwijderen van sessies op
andere locaties)
$db = Propel::getConnection();
$stmt = $db->prepare('DELETE FROM sessions
  WHERE sess_id != ? AND sess_id IN(
    SELECT sess_id FROM session_user WHERE user_id = ?)');
$stmt->bindValue(1, $sessId);
$stmt->bindValue(2, $userId);
$stmt->execute();
```

8.6.2 Design patterns

In dit hoofdstuk worden de design patterns intercepting filter en context object beschreven, die ik gebruikt heb voor dit project.

Intercepting filter

De intercepting filter is een pattern verantwoordelijk van het injecteren van gedrag voor en/of na de verwerking van een verzoek. Hierdoor is het mogelijk om voor en na het renderen van de pagina bepaalde acties uit te voeren. Dit was nodig om voordat de pagina geladen wordt te controleren of de sessie is gesynchroniseerd (zie 8.7 Construction fase). Ik heb voor de intercepting filter pattern van Symfony gekozen. Het is aantrekkelijk om zoveel mogelijk gebruik te maken van functies die standaard in Symfony aanwezig zijn.

Context object

De context object pattern biedt een efficiënte en applicatie-transparante manier van het delen van informatie tussen de verschillende lagen in een software systeem. Zoals in de code van de *clSessionSynchronizationFilter* is te zien, wordt hier gebruik gemaakt om de controller, configuration, request en user object mee op te halen (zie 8.7 Construction fase).

8.7 Construction fase – bouwen

Tijdens de construction fase ben ik begonnen om Single sign-on in de Clubs applicatie te implementeren. Hiervoor kon ik al enkele stukken code gebruiken die ik tijdens het maken van het prototype had ontwikkeld.

Een van de belangrijkste onderdelen van de centrale login was het synchroniseren van de sessie. Hiervoor heb ik gebruik gemaakt van de *intercepting filter* en de *context object* (hieronder in het geel) pattern. Hieronder een voorbeeld van de code:

/apps/frontend/lib/clSessionSynchronizationFilter.class.php

```php
<?php

class clSessionSynchronizationFilter extends sfFilter
{
  public function execute($filterChain)
  {
    if ($this->isFirstCall())
    {
      $context = $this->getContext();
      $request = $context->getRequest();
      $user = $context->getUser();
      /*
       * Do not synchronize for:
       * - bots, spiders
       * - non-GET requests (because redirect won't work)
       * - AJAX requests (no redirect possible)
       * - when cookies are disabled
       */
      if ($request->isMethod(sfRequest::GET)
        && !$request->isXmlHttpRequest()
        && $user->isSynchronized() === false
        && !$user->isBot()
        && !preg_match('/'.str_replace('.', '\.',
              strstr(sfConfig::get('app_login_host'), '.')).'/',
            $request->getUri())
        && !in_array($context->getActionName(),
            array('sessionSynchronization',
                  'sessionSynchronizationCallback'))
        && $request->getParameter('cookie') !== 'false')
      {
        $context->getConfiguration()->loadHelpers(array('Url'));
        $context->getController()->redirect(
          '@session_synchronization?callback='
          .urlencode(url_for('@session_synchronization_callback',
          true)).'&return='.urlencode($request->getUri()));
      }
    }
    $filterChain->execute();
  }
}
```

De filter heb ik vervolgens toegevoegd aan de filter chain in het bestand *filters.yml*.

/apps/frontend/config/filters.yml

```
rendering: ~

clubHostnameFilter:
  class: clClubHostnameFilter

remember:
  class: clGuardRememberMeFilter

security:  ~

sessionSynchronizationFilter:
  class: clSessionSynchronizationFilter

credentialFilter:
  class: clCredentialFilter


cache:     ~

common:
  class: sfCommonFilter
```

Tijdens het project heb ik ook gebruik gemaakt van het *active record* pattern m.b.v. het Propel ORM. Hieronder is een stuk code te zien waar stap 5 van Figuur 15 is uitgewerkt:

/apps/frontend/modules/sfGuardAuth/actions/actions.class.php

```
<?php

public function executeSessionSynchronization(sfWebRequest $request)
{
  $con = Propel::getConnection(SessionTokenPeer::DATABASE_NAME);
  $con->beginTransaction();

  $sessionToken = new SessionToken();
  $sessionToken->setSessId(session_id());
  $sessionToken->setExpires(date('Y-m-d H:i:s', strtotime('+1 min')));
  do
  {
    $token = $sessionToken->generateToken();
    $sessionToken->setToken($token);
  }
  while (!$sessionToken->validate('token'));
  $sessionToken->save($con);
  $con->commit();

  if ($callback = $request->getParameter('callback'))
  {
    $this->redirect(urldecode($callback)
      .(strpos('?', $callback) !== false ? '&' : '?')
      .'token='.$token
      .'&return='.urlencode($request->getParameter('return')));
  }
  else
  {
    $this->redirect('@homepage');
  }
}
```

Hierboven is het SessionToken object te zien. Dit is een model klasse voor de database tabel *session_token*. De klasse SessionToken is gegenereerd door Propel. Door het aanroepen van de *save* method wordt de data die met de setter methodes zijn gedefinieerd naar de database weggeschreven. Ik heb hier gekozen voor een database *transaction,* om te voorkomen dat er dubbele session tokens naar de database kunnen worden geschreven (zie hierboven in het geel).

8.8 Transition fase – project afronden

In de transition fase heb ik het project afgerond. Het oplossen van fouten was niet nodig. Er waren geen meldingen binnen gekomen van fouten. Verder heb ik instructies gegeven van de stappen die nodig zijn voor de deployment.

9 Beroepstaken

9.1 Ontwerpen, bouwen en bevragen van een database

Tijdens de elaboration fase van "*Deelproject I: Social Media Connectivity* Interface" is er een stuk van de database ontworpen. Tijdens de construction fase zijn deze toevoegingen aan de database gebouwd en zijn er gegevens bevraagd aan de database.

9.2 Ontwerpen systeemdeel

Tijdens de elaboration fase van "*Deelproject I: Social Media Connectivity* Interface" is er gewerkt aan het ontwerpen van een interface en de implementatie in de huidige code.

9.3 Bouwen applicatie

Tijdens de construction fases van "Deelproject I: Social Media Connectivity Interface" en "Deelproject II: Centrale login" is er code geschreven bij het bouwen van nieuwe features aan de applicatie.

10 Evaluatie

In dit hoofdstuk komen mijn persoonlijke ervaringen aan bod over de opgeleverde producten en hoe het proces van het project is verlopen.

10.1 Productevaluatie

De doelstelling van het project bestond uit het opleveren van twee producten. Als ik naar het product kijk dan ben ik over het algemeen best tevreden. Over een aantal punten ben ik iets minder tevreden.

Opdracht 1

De eerste opdracht bestond uit het opleveren van een interface die de koppeling kon maken tussen verschillende identity providers, met als doel om de drempel te verlagen voor het registreren op Clubs.

Wanneer ik terugkijk op de doelstelling van deze opdracht dan vind ik dat ik de opdracht goed heb uitgevoerd. Door de vermindering in het aantal handelingen is de drempel lager geworden om lid te worden op Clubs. Een 0-meting vlak voor de release gaf aan dat er +/- 40 aanmeldingen per dag waren. Een 5 tal weken daarna was het opvallend om te zien dat het aantal aanmeldingen naar +/- 60 aanmeldingen per dag was gestegen. Of de stijging alleen te maken had met deze opdracht is niet zeker, omdat er ook andere updates in de diverse software updates zaten die er ook op gericht waren om het aantal leden te vergroten. Ook was het opvallend om te zien dat mensen direct na het beschikbaar komen van de nieuwe functies er nagenoeg direct gebruik van gingen maken.

Een punt waar ik bij de deze opdracht vooral tevreden over ben is het kiezen voor de adapter design pattern. In het begin wist ik nog niet hoe de adapter pattern werkte en dacht ik dat ik daar nog wel even mee bezig zou zijn. Maar tijdens het ontwikkelen van de code bleek dat het toch minder gecompliceerd was dan dat het in het begin leek. Ik kwam er achter dat deze techniek juist helpt om de code sneller te ontwerpen. Ik vind dat de implementatie erg goed gelukt is.

Opdracht 2

Over het uitvoeren van de tweede opdracht ben ik tevreden. De Single sign-on waar het voornamelijk over ging werkt volledig zonder dat de gebruiker daar iets van merkt.

Het ontwikkelen van de centrale login verliep soepel. Door eerst goed vooronderzoek te hebben gedaan naar bestaande oplossingen had ik veel kennis opgedaan van de werking van deze systemen. Daarnaast had ik veel kennis opgedaan van het Symfony framework met het uitvoeren van opdracht één, waardoor ik een uitgebreid ontwerp kon maken. Hierdoor kon ik sommige technische beslissingen die betrekking hadden op het Symfony framework vroegtijdig voorzien. Het gebruik maken van design patterns vond ik erg leuk. Dit maakt het product naar mijn idee ook professioneler.

Ontwikkelen

Op het programmeer gedeelte ben ik erg gegroeid qua kennis in de programmeertaal PHP. Dit is goed voor mijn ontwikkeling, omdat ik mij in de toekomst graag wil specialiseren in deze richting. Ik ben voor het eerst veel met Symfony framework bezig geweest. Ik heb hierdoor veel ervaring opgedaan met het framework wat goed is voor mijn ontwikkeling.

Testen

Het testen heb ik vooral gedaan door functionaliteiten zelf uit te proberen in mijn eigen testomgeving. Het had beter geweest als ik met unit tests had gewerkt, wat meer zekerheid zou geven of de code werkt. Voor in de toekomst zouden hier nog unit tests voor geschreven kunnen worden, om de functionaliteiten geautomatiseerd te kunnen testen. Verder is er getest met een acceptatietest-omgeving. Tijdens de construction fase heb ik een klein testplan opgesteld van testsituaties die de mensen moesten doorlopen om de functionaliteit te testen. De resultaten van de tests waren op één fout na positief.

Rapportage

Over de rapportage ben ik tevreden. Het beschrijft wat ik gedaan heb en hoe ik tot stand ben gekomen tot mijn beslissingen. De rapportage gaf een goed beeld van het product waardoor de opdrachtgever ook een duidelijk beeld kreeg waar ik mee bezig was.

10.2 Procesevaluatie

Planning

Het uitvoeren van het project aan de hand van de planning verliep wel goed. De planning ging niet altijd zoals ik verwacht had. Ik wist aan het begin van mijn afstudeerperiode niet precies hoelang ik per opdracht bezig zou zijn. Daarom heb ik een globale planning aan het begin gemaakt en me hier zoveel mogelijk aan gehouden.

Ik heb de voortgangsbewaking uitgevoerd door een gesprekje met de lead developer te houden om de één tot twee weken. De elaboration fase liep bij het tweede project iets uit door het uitgebreide ontwerp. Hierdoor bespaarde ik wel tijd tijdens de construction fase.

RUP

Het gebruiken van RUP heb ik als positief ervaren. Ik ben echter niet iteratief te werk gegaan, wat wel een bekend kenmerk is van RUP. De faseringen van RUP waarmee het project doorlopen wordt vond ik wel een prettige manier van werken. Ik vond de keuze om de twee opdrachten op te splitsen in aparte RUP projecten een goed idee. Hierdoor zijn de rapporten van de projecten overzichtelijker geworden.

Bijlage I: Plan van aanpak

Bijlage II: Social Media Connectitvity Interface

1. Inception rapport
2. Elaboration rapport
3. Contruction rapport
4. Transition rapport

Bijlage III: Centrale Login

1. Inception rapport
2. Elaboration rapport
3. Contruction rapport
4. Transition rapport