Afstudeerverslag

Het ontwikkelen van een web performance en security analyse applicatie.

Naam: Othmar Hawker
Studentennummer: 10000720
Studie: Informatica

School: De Haagse Hogeschool
Begeleidend examinator: H.G.J. Bechet
Tweede examinator: A.M.J.J. Lousberg

Bedrijf: Protoneers te Den Haag
Bedrijfsbegeleider: Mark Baaijens en Klaas Punselie

Datum: 20-maart-2017
Versie: 1.0

Referaat

Dit is het afstudeerverslag van Othmar Hawker en het beschrijft, het ontwikkelen van een web performance en security analyse applicatie. De afstudeeropdracht is uitgevoerd in de periode augustus 2016 t/m december 2016 bij het bedrijf Protoneers te Den Haag. Dit verslag is een onderdeel van de opleiding Informatica aan De Haagse Hogeschool te Den Haag.

Descriptoren:

- Site crawler
- PHP-Spider
- Laravel
- PHP
- Scrum

Voorwoord

Dit verslag beschrijf ik mijn werkproces bij de afstudeeropdracht “Het ontwikkelen van een web performance en security analyse applicatie.” bij Protoneers in de periode van augustus 2016 tot en met december 2016.

Hierbij ga mijn dank uit naar:

Mark Baaijens (Bedrijfsbegeleider, Protoneers), voor zijn begeleiding en kennis gedurende de afstudeerperiode en ook voor het beschikbaar stellen van de afstudeeropdracht.

Klaas Punselie (Bedrijfsbegeleider, Protoneers), voor zijn begeleiding gedurende de afstudeerperiode.

Mevrouw H.G.J. Bechet (Begeleidend examinator, HHS), voor haar feedback gedurende de contactmomenten.

Mevrouw A.M.J.J. Lousberg (Tweede examinator, HHS), voor haar feedback gedurende de contactmomenten.

Inhoudsopgave

Inleiding

In de periode van augustus 2016 t/m december 2016 heb ik een afstudeerstage bij Protoneer te Den Haag gedaan. De afstudeeropdracht houdt in dat er een web performance en security analyse applicatie ontwikkeld wordt.

Dit verslag is in drie delen verdeeld, in het eerste deel van het verslag wordt de voorbereidend werkzaamheden beschreven inclusief informatie van het bedrijf.

In het tweede deel van het verslag wordt de uitvoering van het project beschreven. Hierin beschrijf ik per sprint wat ik gedaan heb en ook welke keuzes ik heb genomen.

In het laatste deel van het verslag wordt er geëvalueerd op de opgeleverde producten, beroepstaken en doorgelopen proces. Daarnaast bespreek ik ook over de uitbreidingsmogelijkheden van de applicatie.

Deel 1: Context van de opdracht

In dit deel wordt duidelijk hoe er vooraf over de opdracht gedacht werd en hoe de aanpak van de afstudeeropdracht tot stand is gekomen. Om van de afstudeeropdracht een helder beeld te krijgen zijn de volgende onderwerpen van belang: de organisatie, de afstudeeropdracht en het tot stand komen van het plan van aanpak.

1.1 Bedrijf / organisatie

Bedrijf

Protoneers is een netwerk van professionals dat in Den Haag is gevestigd. Dit bedrijf ontwikkelt minimum viable products (MVP) om te bepalen of het product rendabel is voor de markt. Een MVP is een product dat net genoeg functionaliteiten heeft om te bepalen of de klanten er behoefte aan heeft. Als het product goed is, wordt het verkocht aan de klanten van Protoneers. Deze zijn meestal midden- en kleinbedrijven of startups. Het bedrijf bestaat uit vier vaste medewerkers.

De afstudeeropdracht wordt voor Protoneers uitgevoerd en de stakeholders zijn Mark Baaijens, Klaas Punselie en Maurik Hellebrekers. De projectnaam is EM24ME en het doel ervan is in één oplossing alle security en performance aspecten van een website waarnemen.

Begeleiding

Mark Baaijens is mijn technische begeleider en heeft tijdens mijn afstudeerperiode ondersteuning gegeven bij realiseren van de producten.

Klaas Punselie is verantwoordelijk voor de algemene begeleiding binnen het bedrijf.

Maurik Hellebrekers is een stakeholder van het project. De visie van Maurik is om verschillende security en performance aspecten meettool in één platform bieden. Op de huidige markt zijn deze hulpmiddelen te groot en niet gebruikersvriendelijk. Ze geven detailleerde technische informatie en de meeste MKB'ers hebben geen kennis van dat. Het doel van Maurik is een laagdrempelig applicatie aan MKB'ers bieden.

1.2 Opdrachtsomschrijving

Probleembeschrijving

Veel MKB ondernemers maken gebruik van het internet om hun klanten te bereiken. Sommige ondernemers, zoals webwinkels maken de website de hoeksteen van hun product of dienstverlening. In een fysieke winkel kun je als eigenaar inspelen op de zaken die ad hoc gebeuren, maar ter vergelijking in het online kanaal zijn deze zaken op dit moment net wat ingewikkelder maar niet onmogelijk. Op dit moment hebben de MKB ondernemers onvoldoende grip op de security en performance aspecten van hun website.

Er zijn veel tools op de markt die beveiliging en performance meten, maar meeste producten zijn qua gebruik en business model niet gericht op de kleinere ondernemers en er bestaan bijna geen geïntegreerde oplossingen.

Doelstelling van de opdracht

Door middel van een webapplicatie kan de MKB ondernemers meer controle of toezicht krijgen op hun online infrastructuur. Het doel is om de performance, beschikbaarheid, ongebruikelijk gedrag en bestandsintegriteit te meten of bepalen. In de toekomst kunnen hier andere aspecten toegevoegd worden. Hier worden de performance en security aspecten gedefinieerd:

1. Onder beschikbaarheid wordt de uptime van de server en de status van elke webpagina verstaan.
2. Onder performance wordt er gemeten hoe lang het duurt totdat een webpagina volledig laadt.
3. Onder ongebruikelijk gedrag wordt er verstaan of een gebruiker rare tekens in een invoerveld invult en of een gebruiker teveel pogingen maakt op een wachtwoordveld.
4. Als laatst wordt onder bestandsintegriteit verstaan of een bestand op de webserver is gewijzigd.

De eerste en tweede performance aspecten kunnen alleen gemeten worden om door alle links van de website te crawlen. Hiervoor wordt er een site crawler gebruikt. Een site crawler is een bot die een website op een methodische en automatiseerde manier doorbladert. Alle security en performance aspecten worden door meetinstrumenten gemeten. Een meetinstrument is een component die één security of één performance aspect meet of waarneemt. Om de derde en vierde security aspecten vast te leggen, hebben de meetinstrumenten toegang tot de backend van de webserver nodig.

De applicatie wordt als een website aan de gebruiker gepresenteerd waarin ze hun e-mail adres en website moeten aangeven om een scan uit te voeren. De resultaten van de metingen worden naar de gebruiker via e-mail gestuurd. Als de gebruiker meerdere scan heeft gedaan, dan bevatten de resultaten ook trends van de vorige scans. Als laatst moet er rekening met de uitbreidbaarheid van de applicatie houden, want in de toekomst zullen er meer meetinstrumenten erbij komen.

Kortom, de doelstelling van de opdracht is om vier security en performance aspecten van een website te meten of bepalen. Dit wordt gedaan met behulp van een site crawler. De resultaten van de metingen inclusief de trends van vorige scans worden aan de gebruiker gepresenteerd. Als laatst moet er rekening met de uitbreidbaarheid van de applicatie houden.

Deel 2: Uitvoering

Dit afstudeerproject is in acht sprints gedeeld. Er is met sprints gewerkt om de voortgang van het project te kunnen monitoren en ook om de (deel)producten te evalueren. Deze producten kunnen documentatie, functionaliteit en ook ontwerpkeuzes zijn. Elke sprint heeft een tijdsduur van twee weken. In het begin van de sprint wordt er toegelicht wat de doelen voor die sprint zijn. Als het toepasselijk is wordt er ook naar de correspondeerde requirement verwijst als referentie.

Kortom, in dit deel wordt de uitgevoerde werkzaamheden en de verantwoording ervan, per sprint beschreven.

2.1 Sprint 1 – Voorbereidend werkzaamheden

2.1.1 Planning

In de eerste sprint wordt de voorbereidend werkzaamheden uitgevoerd. Deze zijn het plan van aanpak en de requirements opstellen.

De doelen van de eerste sprint zijn:

- Het plan van aanpak met planning opstellen.
- De requirements opstellen.

2.1.2 Plan van aanpak

Scrum

Voor dit project wordt er Scrum gekozen als softwareontwikkelmethode, want het is een bekende methodiek voor mij en Protoneers. Ik heb in mijn derdejaars stage Scrum gebruikt, dus ik heb voldoende ervaring ermee. Er wordt met een aangepaste Scrum methodiek gewerkt, omdat niet alle activiteiten van toepassing binnen dit project zijn en het ontwikkelteam is niet groot genoeg. De activiteiten die wel gebruik worden zijn: Sprints, user stories, Scrum board en Standup Meeting.

Er zijn totaal 8 sprints. De eerste twee sprints zijn als voorbereiding voor de onderzoeksopdracht. Het maken van een plan van aanpak en een onderzoek voeren. Sprint 3 tot en met sprint 7 voer ik de noodzakelijke werkzaamheden uit, zoals ontwikkelen en testen. De laatste sprint is voor de afronding van het project.

User stories worden vanuit interviews met de opdrachtgever verzameld. Hierna worden de user stories ingevoerd in Trello (online Scrum board). Vervolgens worden de user stories ingepland voor een sprint en daarna worden ze in taken verdeeld.

Voor het bijhouden van de user stories wordt er een Scrum board gebruikt. De Scrum board wordt digitaal op Trello.com bijgehouden. De Scrum board is in vier kolommen verdeeld namelijk: product backlog, inbox, doing en done. In de kolom ‘product backlog’ staat de user stories en deze userstories worden in taken verdeeld in de ‘inbox’ kolom. In de kolom ‘doing’ staat de taken waarmee ik bezig ben. In de ‘done’ kolom staat de user stories en taken die uitgevoerd zijn. Dit geeft een duidelijk beeld aan de opdrachtgever over mijn voortgang.

De Standup Meeting wordt drie keer per week gehouden. Twee keer per week met mijn technische begeleider en een keer per week met mijn algemene begeleiding. De Standup Meeting wordt gedaan om de voortgang van het project bij te houden.

Onderzoek naar site crawler

De belangrijkste component van het systeem is de site crawler en het is niet de gewenst dat ik zelf een site crawler ga maken. Het is tijdrovend om één te maken en er zijn verschillende opensource site crawlers op het internet beschikbaar. Daardoor ga ik een onderzoek verrichten om de juiste site crawler

te kiezen. Ik moet eerst de opdrachtgever interviewen om de eisen van de site crawler te achterhalen en daarna een onderzoekplan stellen.

Ontwikkelen

Op dit moment is er geen applicatie voor de klant EM24ME ontwikkeld. Ik moet alle onderdelen van de applicatie zelf realiseren. Van Protoneers moet ik de programmeertaal PHP en de Laravel Framework gebruiken, want daarover kan ik begeleiding krijgen.

Het ontwikkelen van de webapplicatie begint in sprint twee en eindigt in sprint zeven. Als het onderzoek voltooid is, ga ik een globaal ontwerp maken. Dit ontwerp legt het verband tussen de site crawler en het systeem. Daarna kan ik met de werkzaamheden zoals programmeren en testen beginnen. Ik begin eerst met een basissysteem dat alleen de doorzochte links kan tonen. Daarna ga ik via een iteratief proces de meetinstrumenten toevoegen.

De volgende tools zijn nodig voor de ontwikkelomgeving:

Naam van de tool	Functie van de tool
Netbeans met PHP module	Software-ontwikkelomgeving
HeidiSQL	Databasebeheer
Bitbucket	Version control met Git
Composer	Depenendcy manager voor PHP
XAMPP	Lokale HTTP en MySQL server

Tabel 2: Tools voor de ontwikkelomgeving

De programmeertaal van de webapplicatie is PHP, want in dit project wordt het Laravel Framework gebruikt. Ook is mijn technische begeleider is heel bekend met de taal, dus ik kan begeleiding krijgen in deze omgeving. HTML, CSS en javascript horen altijd bij een webapplicatie.

NetBeans is als IDE gekozen, want het is gratis en het bevat een PHP interpreter die de syntactische fouten laten zien. Ook heb verschillende IDEs gebruikt en ik ben meest comfortabel met NetBeans.

Ik gebruik HeidiSQL als database beheer, want het gebruikt niet veel resources vergeleken met andere databasebeheer-applicaties. Met HeidiSQL kan ik een verbinding met de database maken en ook CRUD operaties voeren.

Ik gebruik Git als versie beheer, want Protoneer heeft een Git repository. De Git repository is bij bitbucket.org.

Er wordt een dependency manager gebruikt, zodat de noodzakelijke dependencies automatische wordt toegevoegd en ook geüpdatet. Tijdens het deployment van de applicatie worden alle dependencies automatisch met de applicatie meegenomen.

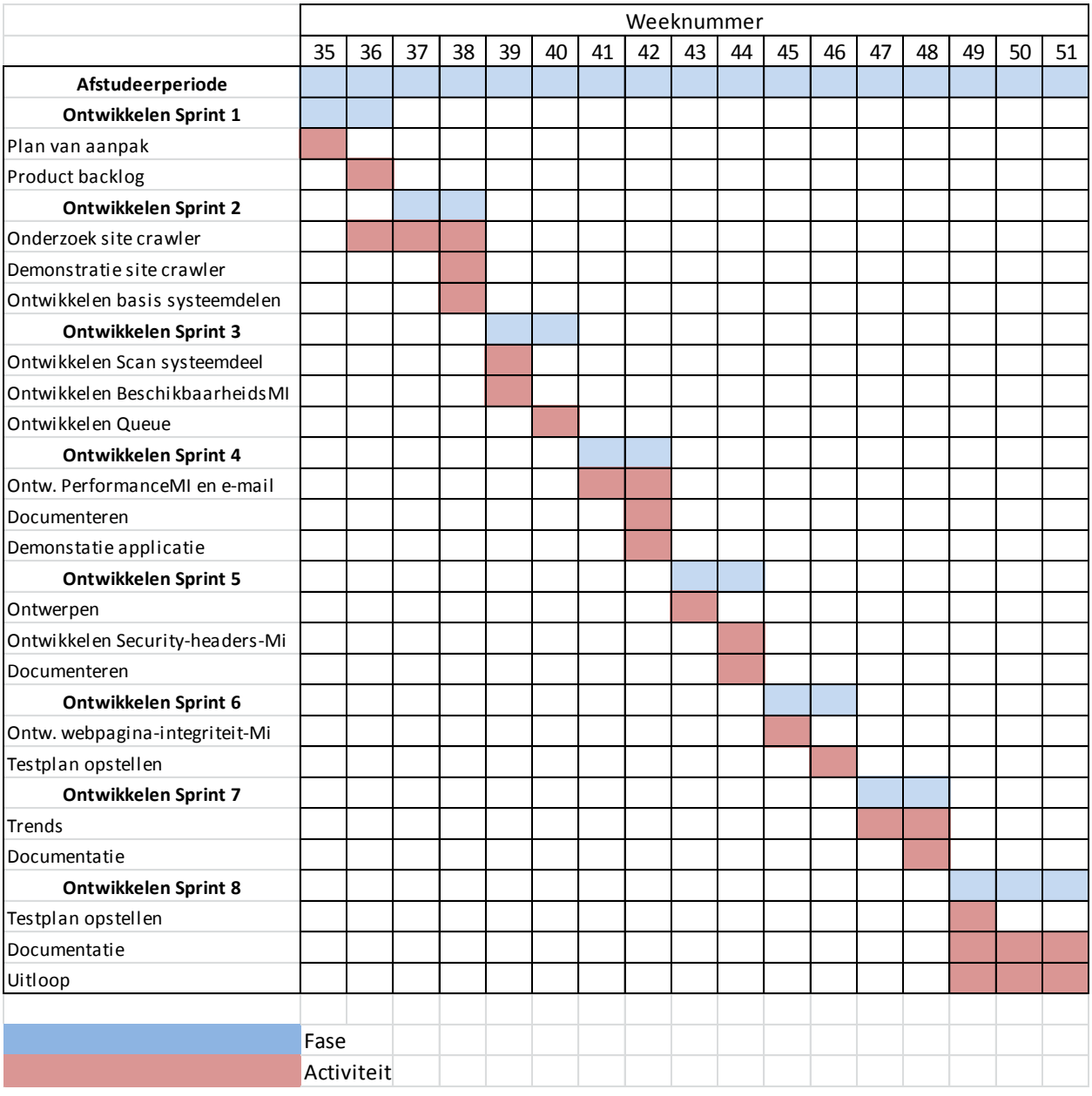
Doordat ik een webapplicatie ontwikkelt, moet het op een webserver draaien en de applicatie is via een webbrowser te bereiken. Ik gebruikt XAMPP want het bevat HTML module, PHP module en MySQL module. Het bevat alles wat ik nodig heb om een webapplicatie lokaal op mijn pc te laten draaien.

Verwacht resultaat

Aan het einde van de afstudeerperiode is er een security en performance scan applicatie ontwikkeld. De applicatie bevat vier meetinstrumenten, een database, een e-mail module, en een site crawler. Het is mogelijk om via de website een scan in te plannen en de resultaten via e-mail ontvangen.

Planning

In dit project wordt er een sprintduur van twee weken gehanteerd, omdat twee weken genoeg tijd is om een systeemdeel te realiseren. In de afstudeerperiode is er totaal acht sprints. Er is een planning gemaakt waarbij de eerste sprint voorbereidend werkzaamheden worden uitgevoerd. Sprint twee tot en met sprint zeven worden de uitvoerende werkzaamheden gedaan en sprint acht is voor uitloop en afstudeerverslag schrijven.



Tabel 3: Strookplanning

(In de bijlage is het volledige plan van aanpak te vinden.)

2.1.3 Requirements opstellen

In overleg met de opdrachtgever zijn er een aantal eisen veranderd. De opdrachtgever wil dat de gebruiker de applicatie niet voor het gebruiken moet instellen. Hierdoor wordt het doel van twee meetinstrumenten aangepast.

Doordat het ongebruikelijke gedrag meetinstrument en het bestandsintegriteitsmeetinstrument eisen toegang tot de backend van een website, worden ze niet in deze afstudeeropdracht gerealiseerd. In plaats van het ongebruikelijke gedrag meetinstrument, wordt er gecontroleerd of er HTTP-security-headers aanwezig zijn. In plaats van het bestandsintegriteitsmeetinstrument, wordt er de integriteit van een webpagina waargenomen. Deze twee meetinstrumenten hebben geen toegang tot de backend van een website nodig.

In overleg met de opdrachtgever heb ik tien requirements opgesteld. De requirements zijn in deze tabel weergegeven:

ID	Requirement
R.01	De applicatie kan een performance en security scan op iedere(meeste) MKB website uitvoeren.
R.02	De applicatie kan op basis van links door een MKB website zoeken.
R.03	De applicatie kan de laadtijd van een webpagina meten.
R.04	De applicatie kan de beschikbaarheid van een webpagina waarnemen.
R.05	De webapplicatie kan aan hand van HTTP-security-headers de veiligheid van een website bepalen.
R.06	De webapplicatie kan de integriteit van de webpagina's van een website waarnemen.
R.07	De applicatie biedt de gebruiker een Quickscan keuze/interface, waarbij alle meetinstrumenten worden bij de scan gebruikt.
R.08	De resultaten van de scan worden via e-mail naar de gebruiker gestuurd.
R.09	De resultaten van de scan ook gebruikt om met eerdere scans te vergelijken.
R.10	Er moet rekening gehouden worden met de uitbreidbaarheid van de webapplicatie.

Tabel 1: Requirements

User stories opstellen

Na het opstellen van de requirements, worden er gedetailleerde user stories voor elke requirement gemaakt. De user stories bevatten de volgende informatie:

- Een beschrijving van de user story.
- De acties die een gebruiker op het systeem moet doen.
- De taken die uitgevoerd moet worden om de user story af te ronden.

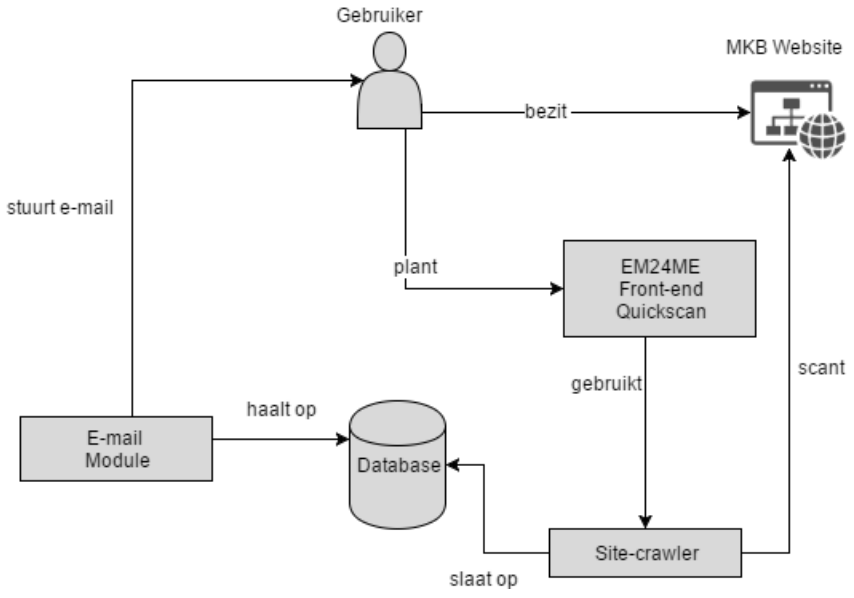
Voorbeeld van een user story:

User story 01 – Quickscan Requirement R.07	
User story	De webapplicatie biedt de gebruiker een Quickscan keuze, waarbij alle meetinstrumenten worden bij de scan gebruikt.
Detail	1. De gebruiker moet hun e-mail adres en website geven om een scan in te plannen. 2. De applicatie gebruikt alle beschikbare meetinstrumenten bij deze scan. 3. Na de scan worden de resultaten naar de gebruiker ge-e-maild.
Taken	1. Maak een front-end interface voor de gebruikers. 2. Maak een queue voor het inplannen van de scans. 3. Maak een Quickscan job die alle beschikbare meetinstrumenten bij de scan gebruikt. 4. Na de scan wordt een e-mail met de resultaten gestuurd.

De requirements worden bij het begin van elke sprint naar een gedetailleerde user story bijgewerkt.

Systeem context

De onderstaande context diagram geeft een eerste beeld van de structuur van het systeem. Het context diagram geeft de relatie tussen het systeem en de externe omgeving aan.



Afbeelding 1: Context Viewpoint van het systeem

Zoals in afbeelding 1 is getoond, heeft een gebruiker een website. De gebruiker plant via de EM24ME Quickscan website een scan in. De scan wordt door de site crawler uitgevoerd, scant de website van de gebruiker en gebruikt alle beschikbare meetinstrumenten. Alle resultaten van de scan worden in de database opgeslagen. Hierna analyseert de e-mail module de resultaten en toont deze in een e-mail die naar de gebruiker wordt gestuurd.

De gebruiker (actor) kan via de EM24ME applicatie een scan inplannen en ontvangt ook van het systeem een e-mail met de resultaten van de uitgevoerde scan.

Het systeem scant een extern systeem voor kwetsbaarheden en performance-problemen. Het externe systeem is altijd de website die de gebruiker aangeeft. Daarnaast gebruikt het systeem een database om de scanresultaten op te slaan.

2.1.4 Resultaat

De risicofactoren in mijn plan van aanpak zijn over de applicatie en ze zijn niet over de afstudeeropdracht, dus ik moet het verbeteren. De planning van het plan van aanpak moet ik gedetailleerder maken, want op dit moment krijgt men niet een goede blik wat er precies gedaan moet worden. Het is beter dat ik de planning in systeem delen ga verdelen.

2.2 Sprint 2 – Onderzoek site crawlers

2.2.1 Planning

Het doel van deze sprint is om een onderzoek te verrichten om een opensource site crawler in het project te gebruiken en het globale ontwerp maken.

User story 02 – Website scannen voor kwetsbaarheden Requirement R.01	
User story	De applicatie kan een performance en security scan op iedere MKB website uitvoeren.
Detail	1. De applicatie biedt de gebruiker een QuickScan keuze, waarbij alle meetinstrumenten bij de scan worden gebruikt. 2. De applicatie zoekt op basis van links door een MKB website.
Taken	1. Voer een onderzoek uit naar een geschikte site crawler. 2. Integreer de site crawler in de applicatie.

De taken voor deze sprint zijn:

- Een onderzoek naar opensource site crawler verrichten.
- De gevonden site crawler in de applicatie toevoegen.
- De applicatie aan de stakeholders presenteren en de volgende functionaliteiten moeten aanwezig zijn:
    - De applicatie moet een front-page met website en email bevatten.
    - De applicatie moet de ingevoerde website crawlen en ook alle gecrawlde links tonen.
    - Na de scan moet de applicatie een email naar de gebruiker sturen; de e-mail mag leeg zijn.

2.2.2 Resultaat onderzoek naar opensource site crawler

Volgens requirements R.01 en R.02 moet de webapplicatie door een MKB website crawlen. Dit wordt met een site crawler gerealiseerd. Doordat het veel tijd kost om een site crawler te maken, ga ik een onderzoek verrichten om de juiste site crawler te kiezen.

Onderzoeksplan

Voordat ik met het onderzoek ben begonnen, heb ik eerst een onderzoeksplan opgesteld. Hiermee wordt het duidelijk voor de onderzoeker en de opdrachtgever naar waar er onderzocht word. Het onderzoeksplan bestaat uit de hoofdvraag met de deelvragen en ook hoe ik informatie ga verzamelen. De belangrijkste deel van het plan is hieronder toegelicht.

Er zijn verschillende opensource site crawlers beschikbaar op het internet en er moet de beste voor dit project gekozen worden. De hoofdvraag luidt als volgt:

Welke opensource site crawler kan Protoneers gebruiken om de EM24ME applicatie te maken en wat moet de tool allemaal doen?

Het doel van dit onderzoek is op een systematische wijze een opensource site crawler softwarepakket kiezen voor het maken van de EM24ME applicatie. Bij het kiezen zijn er verschillende criteriums uitgekozen waaraan de site crawlers eraan moeten voldoen.

Om antwoord te geven op de hoofdvraag moeten er eerste een aantal deelvragen beantwoord worden;

1. Wat is een site crawler?
2. Aan welke eisen moet de site crawler voldoen?
3. Welke site crawlers zijn er gevonden?
4. Welke site crawler ga ik gebruiken?

Samenvatting onderzoek

Een site crawler is een bot die een website op een methodische en automatiseerde manier doorbladert.

Voor het project EM24ME te realiseren hebben ze een specifieke site crawler nodig en deze moet de volgende eisen hebben:

- Moet de hele website doorzoeken inclusief resources zoals HTML, CSS, scripts en bestanden.
- Moet in de programmeertaal PHP zijn.
- Moet configureerbaar zijn.
- Moet de HTTP-response en HTTP-request zien en aanpassen.
- Moet voldoende documentatie hebben.

Er zijn totaal vijftien site crawlers gevonden en na het selectietraject was er één over. Ik heb voor ***PHP-Spider*** gekozen, want het voldoet met alle criteriums. De gekozen site crawler is heel configureerbaar, biedt de mogelijkheid om het httpverkeer te wijzigen. Ook biedt het voldoende documentatie om het te gebruiken en volgens GitHub is de site crawler heel populair.

Deelvraag 1: Wat is een site crawler?

Een site crawler is een bot die een website op een methodische en automatiseerde manier doorbladert. Site crawlers zijn één van de belangrijkste component van een zoekmachine. Ze bladeren door alle beschikbare pagina's van een website en bouwen een kaart van de website.

Deelvraag 2: Aan welke eisen moet de site crawler voldoen?

Er zijn verschillende site crawlers softwarepakket op GitHub. GitHub is een hostingwebsite voor opensourceprojecten. Er wordt op GitHub voor de site crawlers gezocht. De eisen die de site crawlers eraan moeten voldoen heb ik van verschillende interviews gekregen en ze zijn in de onderstaande tabel opgegeven.

Criterium	Reden
De site crawler moet door de hele website zoeken inclusief resources. De site crawler moet op basis van links door de website	De site crawler moet de HTML pagina's, CSS scripts, javascript en ook bestanden doorzoeken, want die bestanden en scripts worden waargenomen.

doorzoeken.	
De programmeertaal van de site crawler moet PHP zijn.	De framework van het systeem is in PHP geschreven en als de site crawler ook in PHP is, maakt het eenvoudig om de site crawler daarin te gebruiken.
Het moet gemakkelijk zijn om de site crawler te configureren.	Er moet instellen welke resources (HTML, CSS of bestanden) de crawler gaat doorzoeken, ook of de site crawler door de subdomains gaat zoeken en of de crawler links gaat filtreren.
Het moet mogelijk zijn om de HTTP-request en de HTTP-response aan te passen.	Door de HTTP-request aan te passen kan de site crawler op bepaalde webpagina's in te loggen.
De site crawler moet voldoende documentatie aangeven en wat zegt de GitHub metrics over de populariteit van de site crawler?	Voldoende documentatie maakt het duidelijk wat de site crawler doet en hoe het gebruikt moet worden. Met behulp van de Github metrics weten we hoe populair of vertrouwd de site crawler is. De metrics zijn: aantal stars, aantal watches en aantal forks.

Tabel 2: Eisen van de site crawler

Deelvraag 3: Welke site crawlers zijn er gevonden?

Er zijn totaal vijftien opensource site crawlers gevonden. Ik heb twee knock-out criteriums gebruikt om tot een shortlist te komen namelijk: de programmeertaal moet PHP zijn en wijzigen van HTTP headers moet mogelijk zijn. Na het toepassen van de knock-out criteriums zijn er vijf site crawlers overgebleven. Hierna had ik alle vijf site crawler uitgeprobeerd en daarna één eruit gekozen, de beschrijving van de vijf site crawler zijn als volgt:

PHP-Crawl, deze site crawler is heel configureerbaar, het is mogelijke te kiezen welke resources het doorzoekt en hoe het de resources opslaat. Wat er bij deze site crawler ontbreekt, is dat het is niet mogelijk om een HTTP-request te wijzigen of inzien.

PHP-Spider, deze site crawler maakt het heel simpel om het httpverkeer te wijzigen en inzien. Dit is heel belangrijk voor het analyseren van een website. Ook zijn de meeste functionaliteiten van deze site crawler configureerbaar. Deze site crawler biedt ook voldoende documentatie om er extra functionaliteiten erbij te implementeren.

Crawl Anywhere, de eerste indruk die ik bij deze site crawler kreeg is dat het meer op een zoekmachine lijkt. Crawl Anywhere is een groot softwarepakket, het bevat: web administratie, crawler, documentverwerking pipeline, Solr zoekmachine en zoekmachine front-end. De crawler kan alleen de inkomende links analyseren en niet de uitgaande links.

FCC/Crawler, deze site crawler heeft bijna geen documentatie en is niet compleet. Er is ook geen mogelijkheid om HTTP-request te wijzigen.

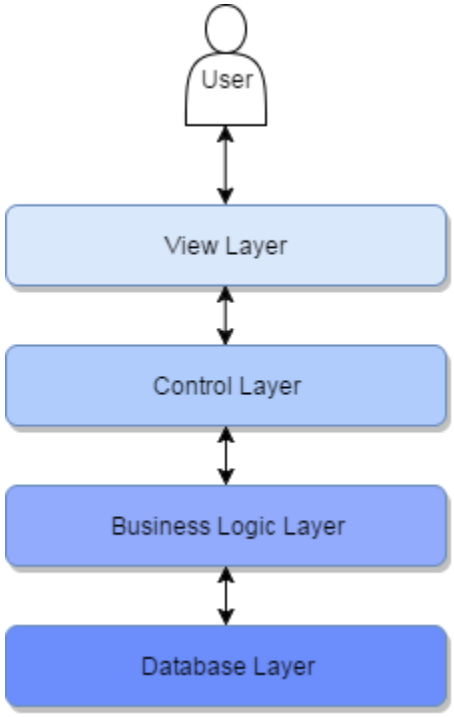
Spatie/Crawler, deze site crawler kan alle links en resources op een website doorzoeken, biedt voldoende documentatie en kan de HTTP-request en HTTP-response inzien. Als laatst is er geen gemakkelijk manier nieuwe functionaliteiten bij deze site crawler toe te voegen.

Deelvraag 4: Welke site crawler ga ik gebruiken?
Ik heb voor PHP-spider gekozen, want het biedt voldoende documentatie, is volgens GitHub heel populair, bevatte mogelijkheid om het HTTP verkeer te wijzigen en is gemakkelijk om wijzigingen in te brengen.

(In de bijlage is het volledige onderzoek te vinden.)

2.2.3 Architectuur van de applicatie

De applicatie wordt in vier lagen ontworpen. Elke laag heeft een verantwoordelijkheid en communiceert alleen met de onderliggende of bovenliggende laag.



Afbeelding 2: Lagen structuur

De vier lagen zijn:

- **View laag:** Deze laag bevat de grafische user interface waarbij de gebruiker met de applicatie communiceert en ook de e-mail templates.
- **Control laag:** Deze laag is verantwoordelijk voor de communicatie tussen de view laag en de business logic laag.
- **Business logic laag:** Deze laag voert uit de scan met behulp van de meetinstrumenten.
- **Database laag:** Deze laag bevat de model-objecten en is ook verantwoordelijk voor het communiceren met de database.

Keuze Laravel Framework

Laravel is een PHP web framework en het is bedoeld voor het ontwikkelen van webapplicaties volgens MVC architectural pattern. Laravel biedt verschillende functionaliteiten zoals:

- **Object-relational mapping (ORM):** Hiermee wordt er een verbinding tussen de database en de model-objecten gemaakt. Met behulp van deze ORM is het mogelijk om de database te bevragen.
- **Routing**: Hiermee worden de HTML-calls naar de correcte controller doorverwezen.
- **Queue**: Biedt de mogelijkheid om tijdrovende taken in te plannen en uit te voeren. Tijdrovende taken worden in een aparte thread uitgevoerd.
- **Mail**: Hiermee is het mogelijk om e-mail te sturen.
- **Testen**: Laravel biedt verschillende testmethodieken zoals: Unit testen, testen van HTTP-requests en webbrowser testen en database testen.
- **Errors en Logging**: Laravel heeft de mogelijkheid om de meeste acties te loggen en de fout meldingen in volle details weergeven.

In dit project wordt Laravel gebruikt voor het maken van de applicatie, want dit is de framework die Protoneers hanteert. Ook is Laravel passend voor de lagen structuur in afbeelding 2, want de structuur is ook volgende de model, view en controller zoals Laravel.

HTML5 Boilerplate front-end keuze

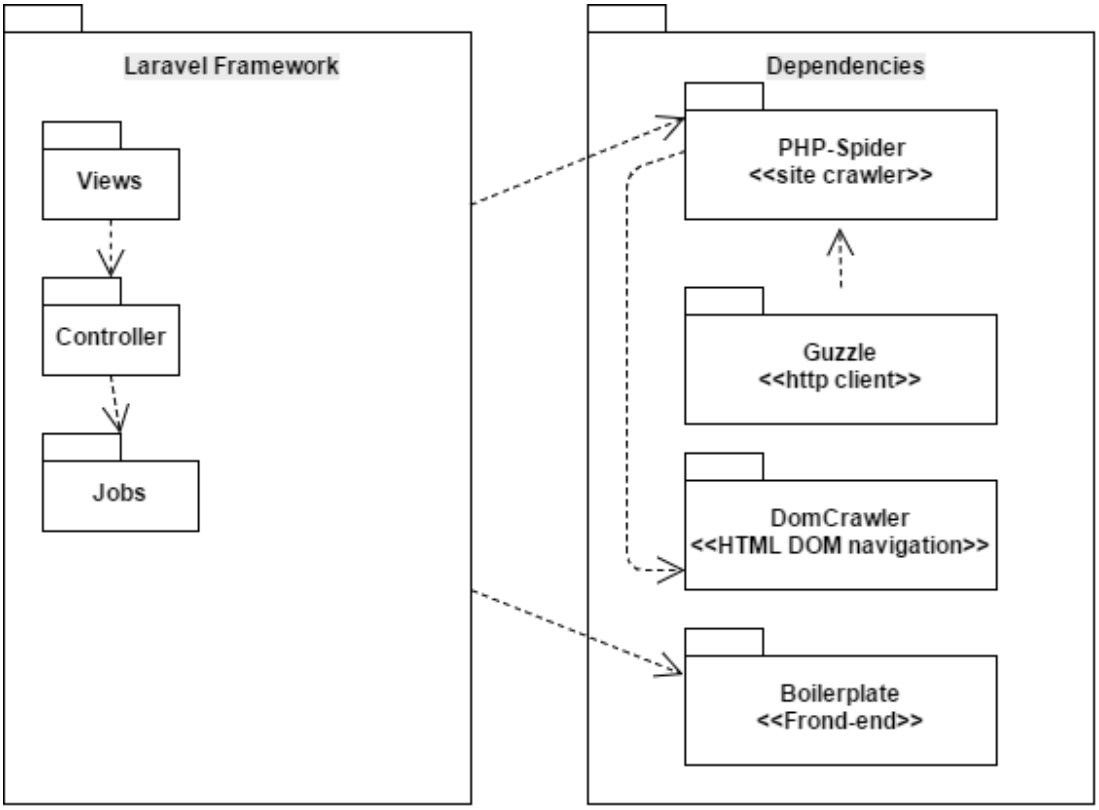
HTML5 Boilerplate is de front-end van het systeem. Deze component bevat verschillende front-end templates om websites snel, robuust en responsive te maken. Deze omgeving is belangrijk voor de eerste indruk van de applicatie. Het biedt verschillende klaargemaakte templates om snel een front-end te ontwikkelen. Hierdoor kan ik als ontwikkelaar meer tijd op het ontwikkelen van de applicatie besteden en zich niet bezig moet houden met de design van de applicatie.

Database keuze

Op dit moment ondersteunt Laravel ORM alleen vier databases, namelijk: MySQL, PostgresSQL, SQLite en Microsoft SQL Server. Er is voor MySQL gekozen, want de meeste webhostingdiensten ondersteunen MySQL en ook ik ben het meest bekend met MySQL.

Globaal systeemontwerp

Bij het einde van deze sprint moet ik een demonstreerbare versie van deze applicatie aan de opdrachtgever presenteren. Hiervoor wordt de volgende onderdelen van de applicatie gemaakt.



Afbeelding 3: Package diagram, Laravel met dependencies

De packages zijn:

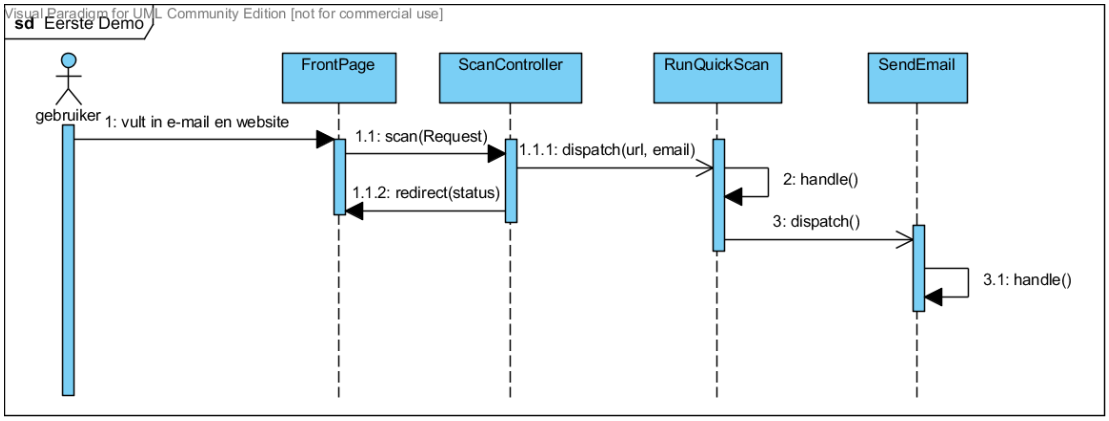
- **View**: Hier komen de HTML-pagina's en formulieren tevoorschijn. Er moet hier een formulier komen waarbij de gebruikers hun website en e-mail moet invullen.
- **Controller**: Hier wordt de HTTP-aanvraag naar de correcte actie doorverwezen.
- **Jobs**: Hier worden de tijdintensief taken zoals een website scannen en e-mailen uitgevoerd.

De dependencies die bij het Laravel project zijn:

- **PHP-Spider**, de gekozen web crawler.
- **HTML5 Boilerplate**, de front-end van het systeem.
- **MySQL,** de DBMS bij het systeem.

PHP-spider is de gekozen site crawler uit het onderzoek en het bevat twee dependencies die een grote rol spelen bij het crawlen van een website. Ze zijn:

- **Guzzle**, is een PHP HTTP-cliënt voor het sturen van HTTP-request. Hiermee is het mogelijk om de HTTP-request en response headers aan te passen en inzien.
- **Symfony DOM Crawler**, is een HTML object navigator. Hiermee is het mogelijk om met een query alle links, CSS bestanden of javascript scripts selecteren. Dit is nodig om door een hele website te crawlen.



Afbeelding 4: Sequence diagram van de demo

1. De gebruiker vult in hun website en e-mailadres en druk de 'Start scan' knop.
2. Er wordt een 'ScanController' object gemaakt en de methode 'scan' ontvangt de aanvraag. De aanvraag bevat de website link en het e-mailadres van de gebruiker.
3. Daarna wordt er een asynchrone aanvraag naar het object RunQuickScan gedaan.
4. De ScanController stuurt de status naar de FrontPage. De status geeft aan dat er is een scan ingeplant.
5. Vervolgens wordt de handle methode van RunQuickScan uitgevoerd, deze methode voert de scan uit.
6. Na de scan wordt er een asynchrone aanvraag naar het object SendEmail gedaan, waarna de handle methode stuurt een e-mail.

2.2.4 Testen

Functionaliteit site crawler

Deze unittesten zijn gemaakt om de functionaliteiten van de front-end te testen. De unittesten zijn met PHPUnit gerealiseerd, het is een testing framwork van Laravel.

Testcase	E-mail	Website	Te verwachten resultaat	Werkelijk resultaat
1	abc@abc.com	http://merlinq.net/	Er verschijnt een melding dat er een scan is ingeplant en binnen 15 minuten heeft de e-mail adres een e-mail ontvangen.	V
2	tom@gmail.com	merlinq.net	Er verschijnt een melding dat er een scan is ingeplant en binnen 15 minuten heeft de e-mail adres een e-mail ontvangen.	X
3	<leeg>	<leeg>	Er verschijnt een melding dat de e-mail en website ontbreken.	V
4	tom@	www.merlinq.net	Er verschijnt een melding dat de e-mail niet geldig is.	V

Tabel 3: Front-end testen

Testgeval 2 is mislukt en dit testgeval wordt in de volgende sprint hersteld.

2.2.5 Resultaat

De applicatie kan een scan op een website uitvoeren en op dit moment toont het alle gecrawlde links in de command-line-interface. De gecrawlde links worden niet opgeslagen en ook niet verwerkt. Na het demonstreren van de demo applicatie waren de stakeholders heel tevreden ermee. Ze vonden dat het een goed begin voor het project was.

Bij het testen van de functionaliteiten van de applicatie blijkt er, dat de site crawler niet werkt zonder de 'HTTP' of 'HTTPS' protocol in de URL. Dit is bij testcase 2 te zien en het wordt in de volgende Sprint opgepakt.

De website geeft geen feedback als er een scan is gepland, dus dat moet ook in de volgende sprint toegevoegd worden.

2.3 Sprint 3 - Beschikbaarheidsmeetinstrument ontwikkelen

2.3.1 Planning

Het doel van deze sprint is om het beschikbaarheidsmeetinstrument te ontwikkelen. Een meetinstrument is een component die één security of één performance aspect meet of waarneemt.

User story 03 – Beschikbaarheidsmeetinstrument Requirement R.04	
User story	De applicatie kan de beschikbaarheid van een webpagina waarnemen.
Detail	1. Tijdens een scan wordt de HTTP-statuscode van elke link waargenomen. De HTTP-status is altijd een cijfer tussen 100 en 600. 2. De HTTP-statuscode wordt opgeslagen met de correspondeerde links. 3. Het resultaat wordt aan de gebruiker in een e-mail aan hand van cards gepresenteerd.
Taken	1. Een generiek meetinstrument klasse die als template dient voor de meetinstrumenten. 2. De 'model' objecten maken. 3. Beschikbaarheidsmeetinstrument ontwikkelen.

De taken voor deze sprint zijn:

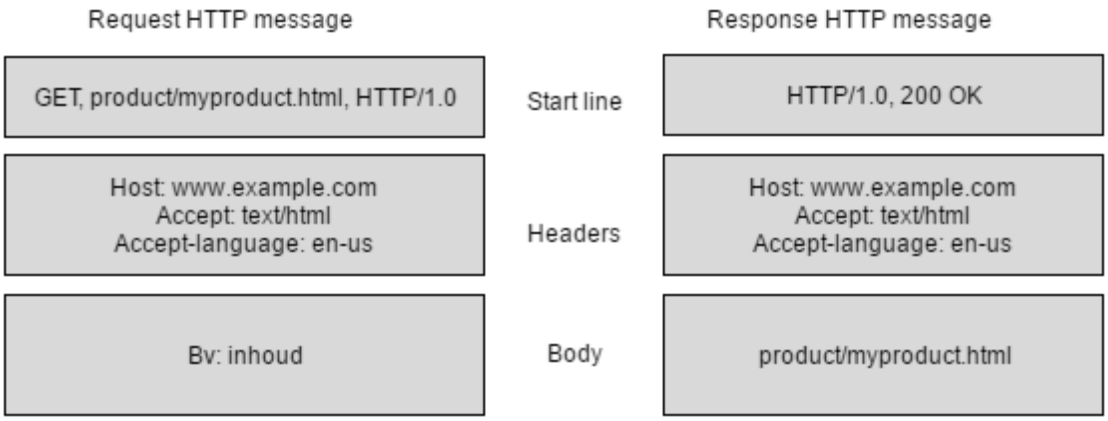
- De structuur van de meetinstrumenten ontwerpen.
- De 'model' objecten maken en ook de database tabellen aanmaken om de metingen op te slagen.
- Het beschikbaarheidsmeetinstrument ontwikkelen. De metingen moeten in de database opgeslagen worden.
- Meetinstrument configureren zodat het alleen doelwit website doorzoekt.

2.3.2 Het HTTP-protocol

Voordat ik de meetinstrumenten beschrijf, moet ik het HTTP-protocol kort uitleggen.

Volgens de Internet Engineering Task Force (IETF)[1] is HTTP een protocol voor de communicatie tussen een webcliënt en een webserver. Dit protocol wordt veel op het internet gebruikt, want het wordt gebruikt om verschillend datatype te verzenden zoals: afbeeldingen, documenten, video's, audio, etc. Het HTTP-protocol werkt op basis van een request en response cyclus. Een cliënt doet een HTTP-request (aanvraag) aan een server en krijgt op een HTTP-response (antwoord) van de webserver.

[1] https://www.w3.org/Protocols/HTTP/1.0/spec.html



Afbeelding 3: HTTP-request en response diagram

Request bericht
In afbeelding 3 is te zien dat een HTTP-request bericht uit drie delen bevat en deze zijn:

- Start line en die bevat de request method (GET, POST, etc), welke resource is geroepen (“product/myproduct.html”) en HTTP-versie.
- Request headers, deze zijn key-value pair en ze zijn de parameters bij de HTTP-request
- Facultatief body.

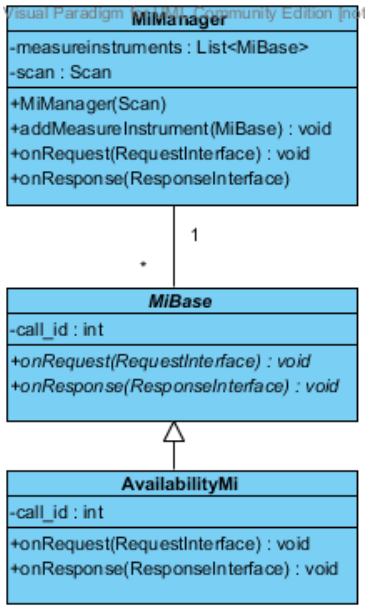
Response bericht
Een response bericht bevat ook uit drie delen en deze zijn:

- Start line en deze bevat de HTTP-status en HTTP-versie, bijvoorbeeld ‘*HTTP/1.0 200 OK*’ betekend dat de request is geslaagd.
- Response headers, deze zijn key-value pair en ze zijn de parameters die bij de HTTP-response horen.
- Gevraagde resource.

Webservers gebruiken het HTTP-protocol om met webbrowsers te communiceren en e-mailservers gebruiken het IMAP-protocol om met e-mailclients te communiceren. Om de veiligheid en performance van een website waar te nemen, moet deze applicatie de HTTP-request en HTTP-response inlezen. Websites communiceren via het HTTP-protocol en hierdoor heeft deze applicatie voldoende informatie om te bepalen of het website veilig is en goed presteert.

2.3.3 Structuur van de meetinstrumenten

Voor de afstudeeropdracht moet er vier meetinstrumenten ontwikkeld worden en in de toekomst zullen er nog meer erbij komen. Op dit moment wordt de scan in de ‘RunQuickScan’ klasse uitgevoerd en die klasse bevindt zich in het package ‘Jobs’. Om rekening met de uitbreidbaarheid van de applicatie te houden, maak ik een nieuw package die ‘MeasureInstruments’ heet. Hierin komen de klassen die verantwoordelijk zijn voor het analyseren van het HTTP-verkeer. De ‘Jobs’ package is verantwoordelijk voor het uitvoeren van scan.



Afbeelding 6: Klassendiagram van het 'MeasureInstruments' package.

In afbeelding 6 is het klassendiagram van de meetinstrumenten te zien. Deze klassen behoren in het 'MeasurementInsturments' package. Alle meetinstrumenten erven van de klasse 'MiBase' over en deze klasse heeft twee belangrijke methodes. Deze zijn: 'onRequest' en 'onResponse'. Hieronder is er een toelichting van deze twee methodes doen.

onRequest

De "onRequest" methode wordt geroepen voordat de site crawler een link gaat crawlen. De methode heeft ook een parameter 'RequestInterface' en hiermee kan het meetinstrument de HTTP-request headers bewerken voor de gegeven link. Het is mogelijk om alle HTTP-request headers hier te wijzigen. Een voorbeeld is om cookie met de aanvraag sturen.

onResponse

De "onResponse" methode wordt geroepen wanneer de site crawler een link heeft gecrawld. Deze methode heeft twee parameters 'ResponseInterface. Met 'ResponseInterface' is het mogelijk om de inhoud van de HTTP-response te zien.

MiManager

De 'MiManager' klasse beheert alle meetinstrumenten die bij de scan behoort. Het 'RunQuickScan' object die de scan gaat uitvoeren, voegt via de 'addMeasurement' methode de benodigde meetinstrumenten toe. Deze klasse heeft ook de 'onRequest' en 'onResponse' methodes en hier wordt de data naar alle toegevoegd meetinstrument gestuurd.

Door dit ontwerp keuze is het mogelijk om nieuwe meetinstrumenten toevoegen zonder wijzigingen in de structuur te brengen. Doordat alle websites het HTTP-protocol gebruiken en met een request en response cyclus werken, werkt deze oplossing.

2.3.4 Beschikbaarheidsmeetinstrument

Het beschikbaarheidsmeetinstrument neemt de HTTP-statuscodes[2] van de response waar. Hiermee is het mogelijk om een linkrot op te sporen. Linkrot is een link op een webpagina die naar een niet-bestaande webpagina verwijst. Ook is het gewenst om de herkomst van de gebroken link aan de gebruiker te tonen.

Webservers gebruiken de HTTP-statuscodes om met elkaar te communiceren. HTTP-statuscode is een cijfer en ze beteken:

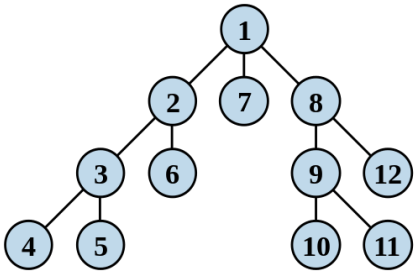
- 1xx: Mededelend
- 2xx: Succes
- 3xx: Omleiding
- 4xx: Aanvraagfout
- 5xx: Serverfout

HTTP-statuscode tussen 100 en 400 betekent meestal dat alles goed is en statuscode boven 399 betekent dat er in de cliënt(bezoeker van de website) of server een fout is. Dit meetinstrument neemt de HTTP-statuscode waar en slaat het in de database op.

2.3.5 Structuur van de metingen

De metingen moeten opgeslagen worden, want ze worden aan de gebruiker gepresenteerd en ook worden ze bij toekomstige scans gebruikt. Eerst wordt er uitgelegd hoe de site crawler werkt en daarna wordt er de structuur van de metingen toegelicht.

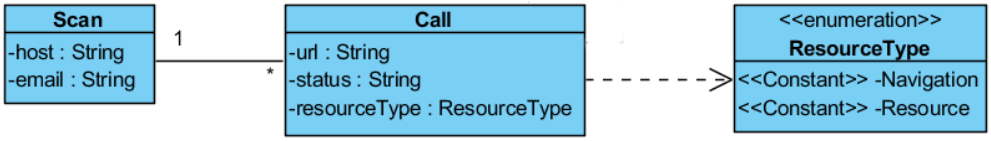
De site crawler die gebruikt wordt, zoekt door een website op basis van links. De site crawler doet dit met een depth-first search (DFS) zoekalgoritme. In afbeelding 7 is de boomstructuur van een scan waarin knooppunt 1 is de hoofdpagina van de gecrawlde website en alle andere knooppunten zijn pagina's en ze zijn als een hiërarchie gestructureerd.



Afbeelding 7: Volgorde waarin de knooppunten van de boomstructuur bekeken worden.

De site crawler levert de mogelijkheid om de HTTP-request en HTTP-response te zien. Hiermee heb ik het volgende ontwerp opgesteld:

[2] http://httpstatus.es



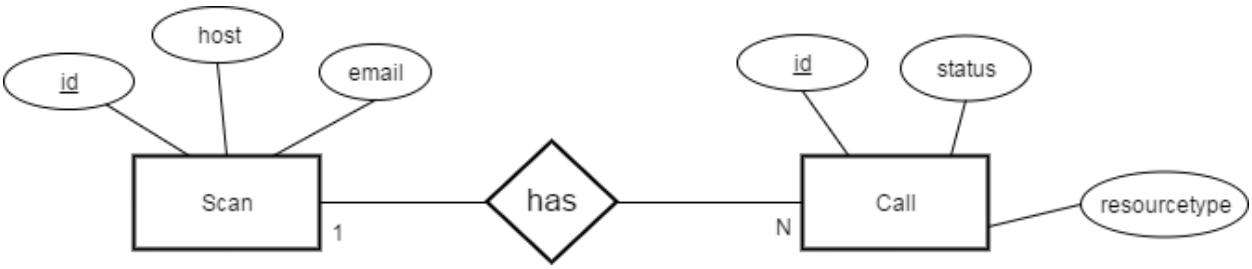
Afbeelding 8: Klassendiagram eerste ontwerp model

Allereerst als de gebruiker een scan inplant, wordt er een 'Scan' object met de volgende attributen gemaakt: website en e-mail adres van de gebruiker.

Vervolgens wordt deze scan door de site crawler uitgevoerd en voor elke link (die de site crawler doorzoekt) wordt er een 'Call' object gemaakt. Het 'Call' object houdt bij de gecrawlde link, status van de pagina en bij welke Scan het hoort. De enumeratie 'ResourceType' houdt bij het type van het 'Call' object. Een 'Call' object heeft twee types, namelijk:

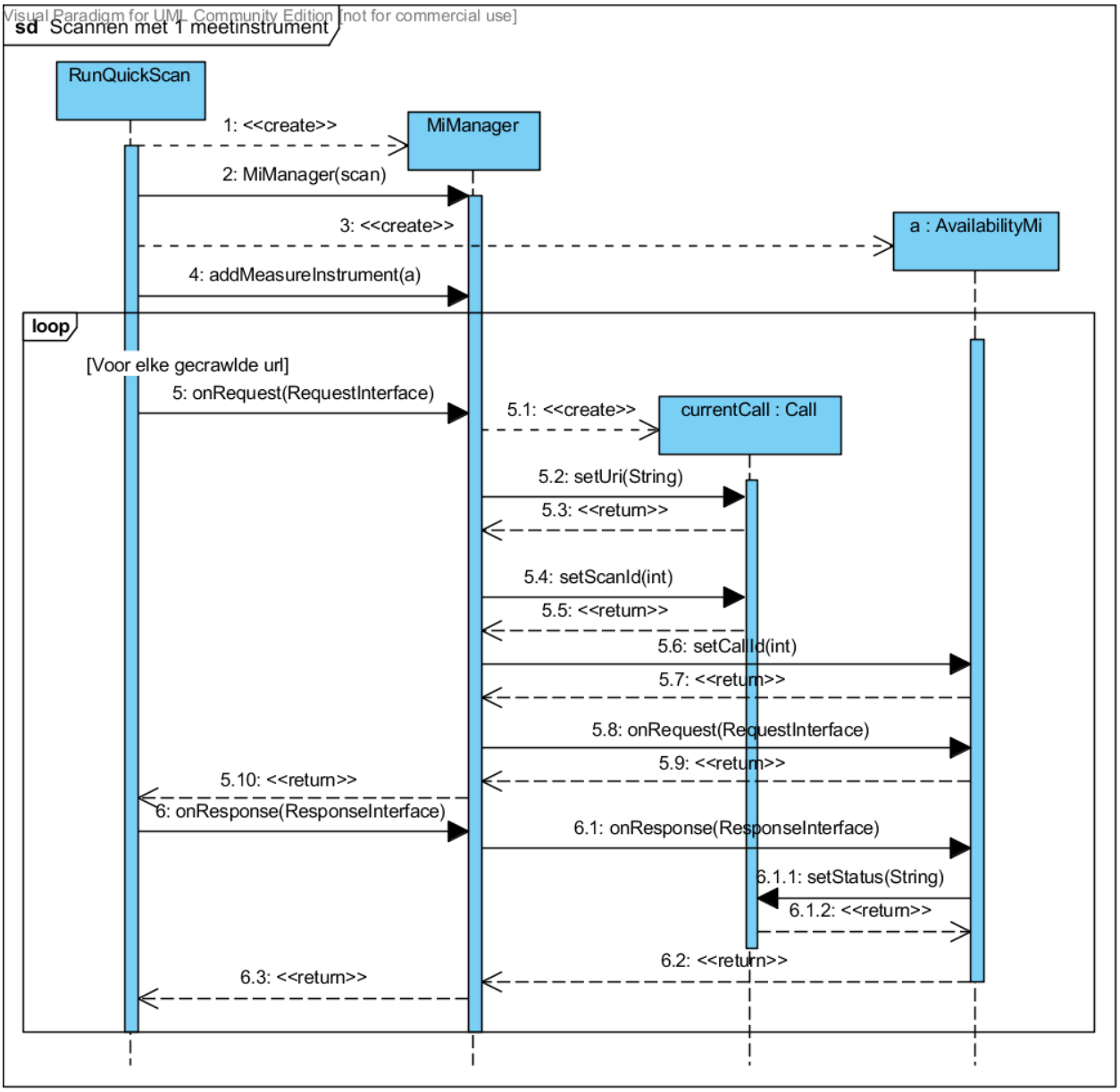
- Navigatie, een HTML-pagina.
- Bron, niet een HTML-pagina zoals afbeelding, scripts of bestanden.

Hieronder is het databaseontwerp van de metingen:



Afbeelding 9: ER-diagram van de metingen

Het database-ontwerp is dezelfde als het klassendiagram in afbeelding 8, want dat is van de Laravel framework vereist. Laravel framework voegt automatisch een timestamp attribuut bij elke tabel toe.



Afbeelding 10: Sequence diagram 'Scannen met één meetinstrument'

In afbeelding 10 is er een sequence diagram van het scannen met het beschikbaarheidsmeetinstrument te zien. Dit sequence-diagram toont hoe het scannen met één meetinstrument gaat.

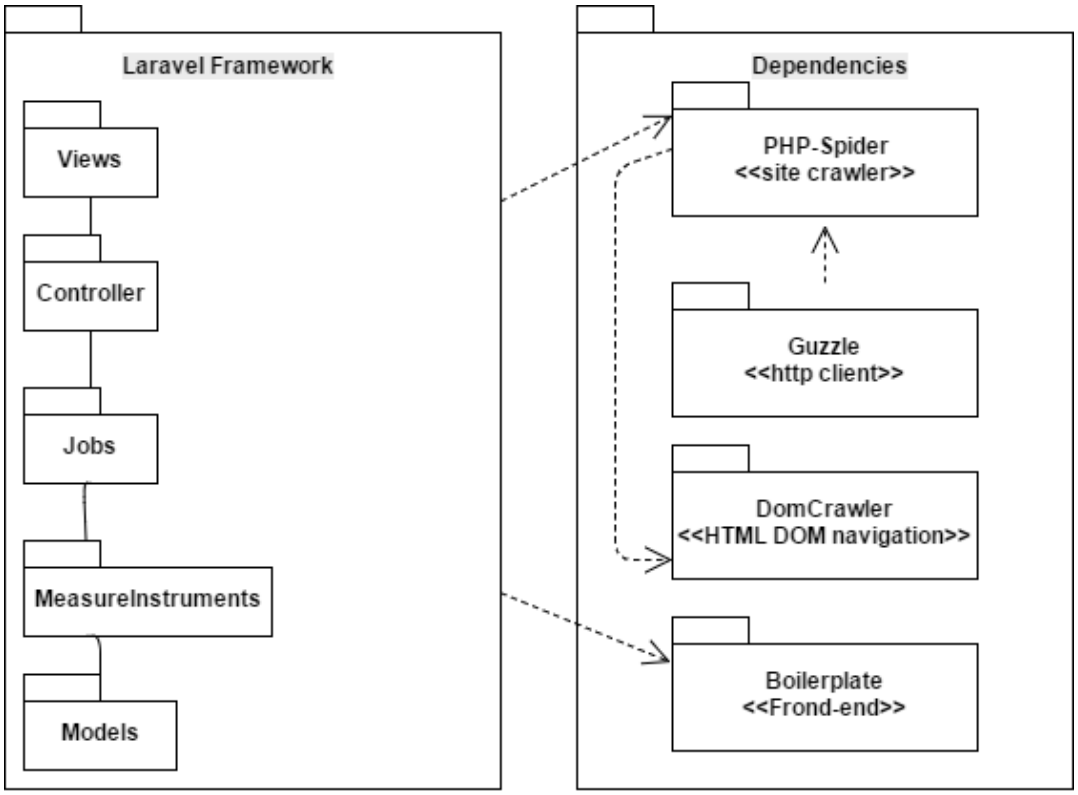
1. Ten eerste wordt er een 'MiManager' object door 'RunQuickScan' gemaakt.
2. Er wordt ook een scan object gestuurd.
3. Daarna wordt het beschikbaarheidsmeetinstrument geïnitialiseerd.
4. Dat meetinstrument wordt bij de 'MiManager' klasse toegevoegd.
5. Voor elke link die de site crawler doorzoekt wordt er de HTTP-request aan de 'MiManager' gestuurd. Voor elke link wordt er een 'Call' object gemaakt met de URL van de link. Het meetinstrument krijgt de id van de gemaakte 'Call' object. Als laatst krijgt het meetinstrument de HTTP-request.

6. Daarna wordt de HTTP-response naar de ‘MiManager’ gestuurd en het wordt ook naar het meetinstrument gestuurd. Het beschikbaarheidsmeetinstrument leest de status van de HTTP-request en voegt dit aan het ‘Call’ object toe.

2.3.6 Resultaat

Na een scan worden de HTTP-statuscodes in de database opgeslagen. Ze worden in de in de ‘Call’ tabel opgeslagen. Ook was er niet genoeg tijd voor het ontwikkelen van de card van dit meetinstrument, dus die taak wordt in de volgende sprint opgepakt.

Tijdens het scannen van een website gaat de site crawler ook een externe website volledig scannen en dat niet gewensts. Ik moet de site crawler in de volgende sprint configureren zodat het alleen de eerste link van een externe website crawlt.



Afbeelding 11: Package diagram

In deze sprint zijn er twee packages toegevoegd, ze zijn “Models” en “MeasureInstruments”. Models bevat de “Scan”, “Call” en “ResourceType” klassen. Het “MeasureInstruments” package bevat alle klassen die verantwoordelijk zijn voor het uitvoeren van een scan. De klassen zijn bij afbeelding 6 te zien.

2.4 Sprint 4 – Performance-meetinstrument ontwikkelen

2.4.1 Planning

In deze sprint wordt het performance-meetinstrument ontwikkeld en het presenteren van de metingen in de e-mail. Hieronder zijn de gedetailleerde user stories voor deze sprint.

User story 03 – Performance-meetinstrument Requirement R.05	
User story	De applicatie kan de laadtijd van een webpagina meten.
Detail	1. Tijdens een scan wordt de laadtijd van elke webpagina gemeten. Dit wordt in milliseconde gemeten. 2. De laadtijd wordt opgeslagen bij de correspondeerde links. 3. Het resultaat wordt aan de gebruiker in een e-mail aan hand van cards gepresenteerd.
Taken	1. Performance-meetinstrument klasse aanmaken 2. De 'Call' klasse wijzigen zodat het ook de metingen van het performancemeetinstrument kan opslaan. 3. Performancemeetinstrument card ontwikkelen.

User story 04 – Cards in e-mail presenteren Requirement R.08	
User story	De resultaten van de scan worden via e-mail naar de gebruiker gestuurd.
Detail	1. Een card is een GUI element die alle bevindingen van één meetinstrument bevat. 2. Een card bestaat uit drie delen, deze zijn: een header, body en footer. 3. De header bevat de titel, het body bevat de resultaten en de footer bevat meestal een overzicht van de resultaten.
Taken	1. Resultaten van de database halen. 2. Opgehaalde resultaten in een HTML-template toevoegen. 3. E-mail naar de gebruiker verzenden.

De taken voor deze sprint zijn:

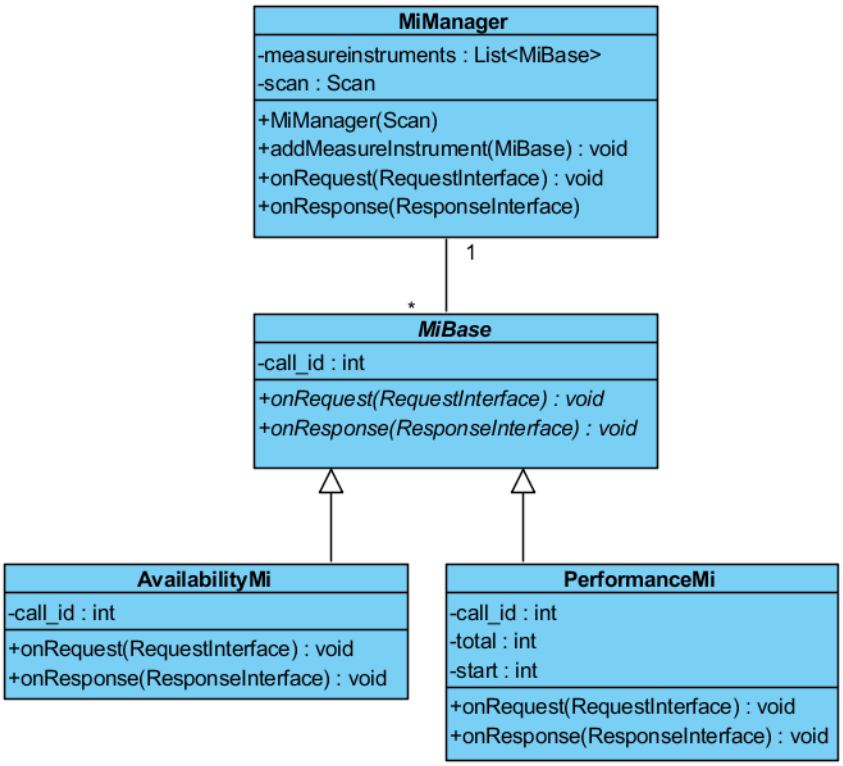
- Performance-meetinstrument aanmaken.
- Het 'Call' object wijzigen zodat de nieuwe metingen opgeslagen kunnen worden.
- Child-parent relatie tussen de 'Call' objecten aanmaken, zodat het beschikbaarheidsmeetinstrument de herkomst van de linkrot kan aanwijzen.
- Site crawler configureren zodat het op één domain blijft.
- Meetinstrument cards ontwikkelen.

2.4.2 Performance-meetinstrument

Het performance-meetinstrument meet hoe lang het tussen de HTTP-request en response duurt en hiermee is het mogelijk om de prestatie van alle pagina's en resources zien. Dit meetinstrument neemt ook de grootte van de webpagina waar, want hiermee wordt de downloadsnelheid berekend. De

paginagrootte staat bij de “Content-Length” HTTP-header veld. De waarde is een geheel getal en de eenheid ervan is byte.

Dit meetinstrumenten slaat op de pagina-grootte in milliseconde en de laadtijd in bytes op.



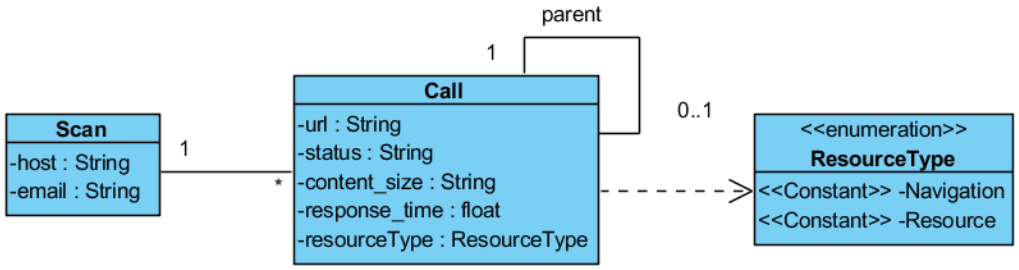
Afbeelding 12: Klassendiagram meetinstrumenten

Het performancemeetinstrument heeft twee attributen nodig om de laadtijd te bepalen. Bij de ‘onRequest’ methode wordt de tijd gemeten en bij de ‘start’ attribuut toegevoegd. Daarna bij de ‘onResponse’ methode wordt de tijd weer gemeten en het verschil tussen de gemeten tijd en de waarde bij de ‘start’ attribuut wordt de laadtijd. De tijd wordt met de ‘microtime()’[3] functie van PHP opgehaald; de functie haalt aan de actuele Unix timestamp in microseconde.

2.4.3 Ontwerp bijwerken

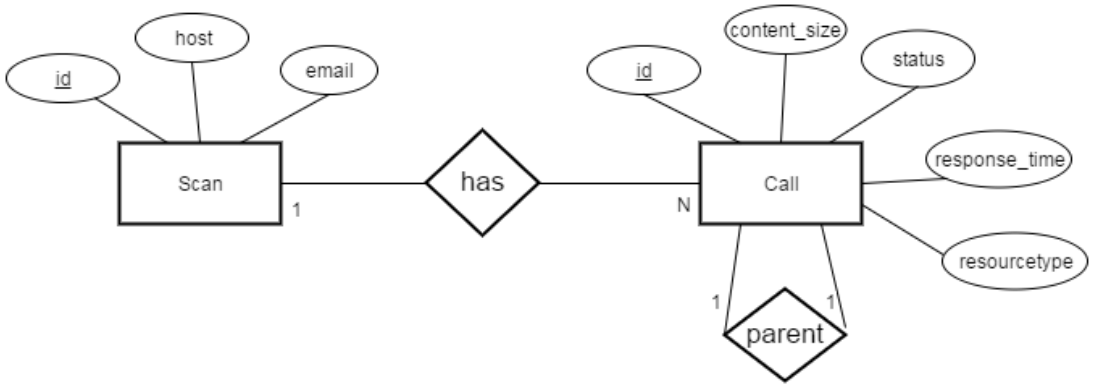
Het huidige ontwerp moet bijgewerkt worden om de metingen van het nieuwe meetinstrument te gebruiken.

[3] http://php.net/manual/en/function.microtime.php



Afbeelding 13: Klassendiagram performancemeetinstrument model

Bij de 'Call' klasse zijn er twee attributen toegevoegd, namelijk 'content_size' en 'response_time'. Het 'content_size' attribuut houdt bij de paginagrootte in bytes en 'response_time' attribuut houdt bij de laadtijd in milliseconde.



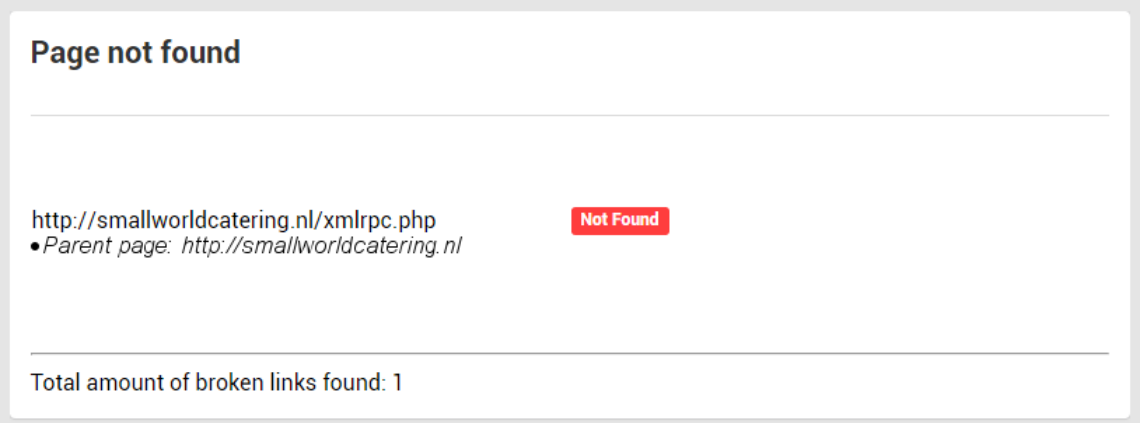
Afbeelding 14: ER-diagram van het vorige klassendiagram

Child-parent relatie

De 'Call' klasse heeft een zelf-associatie die 'parent' is genoemd. Deze relatie in combinatie met het 'resourcetype' attribuut, houdt bij de relatie tussen de links. Bijvoorbeeld het is mogelijk te zien dat een webpagina bestaat uit vier afbeeldingen, twee CSS bestanden en Javascript-bestand. Dit is nootzakelijk bij het beschikbaarheidsmeetinstrument, want daarin wordt deze relatie gebruikt om de afkomst van de linkrot te tonen.

2.4.4 Resultaat in de e-mail presenteren

Na een scan worden de resultaten naar de gebruiker ge-e-maild en daarin bevat verschillende kaarten, ofwel cards. Een card is een GUI element die alle bevindingen van één meetinstrument bevat. Elk meetinstrument dat tijdens de scan was gebruikt, kan zijn bevindingen in één card tonen. Een voorbeeld van een card is in afbeelding 15 te zien.



Afbeelding 15: Performance-meetinstrument-card

Lay-out card

Bij de card is er ook rekening met uitbreidbaarheid van de applicatie gehouden. Als er een nieuw meetinstrument erbij komt, is het niet gewenst om opnieuw een card te maken. In de applicatie is een card HTML-code en het is in drie delen verdeeld. De drie delen zijn: een header, body en footer, dit is ook in afbeelding 15 te zien. Elk deel is los van elkaar zodat er gemakkelijk een card gegenereerd kan worden. Bij elk deel hoort één of meer HTML-bestand en dat bestand bevat een sjabloon voor de card.

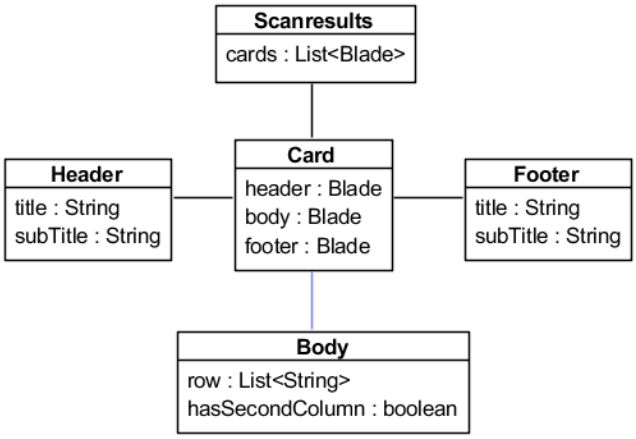
De header en footer bevatten één sjabloon voor een grote vetgedrukte titel met een optionele titel. Het body bevat drie soorten sjablonen en ze zijn:

1. Een lijst met één kolom en elke rij heeft een titel en een verkleinde subtitel.
2. Een lijst met twee kolommen waarbij de eerste kolom een titel bevat en de tweede kolom een rood of groene label bevat.
3. Een afbeelding.

Ontwerp card

Alle card-onderdelen zijn Blade Template[4] bestanden en ze bevinden zich in het 'Views' package. Blade Template is een template engine van Laravel Framework. Het is een syntax die de HTML-bestanden leesbaarder maakt en het presenteren van data gemakkelijker maakt. Er zijn verschillende PHP template engine en ik gebruikt die van Laravel, want het sluit goed aan met de rest van de applicatie.

[4] https://laravel.com/docs/5.4/blade



Afbeelding 16: Onderdelen van een card

Bij afbeelding 16 is er een diagram van de componenten van een card weergegeven. De objecten in het diagram zijn de Blade templates die zijn de componenten van een card. Het diagram is gemodelleerd volgens UML, maar de objecten daarin zijn geen klassen. Ondanks deze reden heb ik toch UML gebruikt, want door Laravel Blade zijn de objecten bijna hetzelfde als UML-klassen. De objecten kunnen variabelen hebben, maar ze zijn een weakly-typed variabele. Dit betekend dat er geen foutmelding wordt gegeven als het variabele die doorgegeven met het verwachte variabele type niet klopt.

Een 'Card' object heeft drie Blade variabels, namelijk header, body en footer. Bij de drie variabels wordt het correcte Blade-object verwacht, bijvoorbeeld het 'body' variabele verwacht het 'Body' object of het 'Chart body' object. De 'Header' en 'Footer' objecten hebben een verplichte titel variabele en een facultatief subtitel variabele.

Het 'Body' object verwacht een lijst van String en een boolean om de tweede kolom te tonen. Het 'Scanresults' object bevat alle cards die uiteindelijk naar de gebruiker wordt gestuurd.

Ik heb voor dit ontwerp gekozen, want het is gemakkelijk om onderdelen te wijzigen en ook erbij toevoegen. Het zorgt ervoor dit deel van de applicatie ook uitbreidbaar is. Een alternatief van deze keuze is om 1 e-mail template maken waarin alle meetinstrumenten hun uitvindingen kunnen plaatsen. Dit alternatief bevat veel herhalende code, want alle cards zijn dezelfde en mijn ontwerp keuze vermijdt dit probleem.

Beschikbaarheidsmeetinstrument card inhoud

Het beschikbaarheidsmeetinstrument card toont de gebroken links inclusief de herkomst pagina. Alleen de gebroken links aangeven is niet genoeg voor de gebruiker, want de herkomst van de gebroken link geeft de gebruiker een specifieke pagina om de gebroken link te herstellen. De query voor het halen van alle gebroken links van de eerste scan ziet er als volgt uit:

"**SELECT** * **FROM** call **WHERE** call.scan_id = 1 **AND WHERE** status >= 400;".

2.4.5 Site crawler configureren

Om ervoor te zorgen dat de site crawler goed werkt, wordt die eerst geconfigureerd. PHP-Spider biedt verschillende configuratie-opties die de site crawler kan beperken.

De eerste configuratie-optie is de URL-ontdekker. Dit geeft aan welk HTML-element de site crawler mag doorzoeken. In dit geval wordt er deze waarde gebruikt "//a|//link|//script". Dit is XPath code en het betekent: selecteer alle a-nodes, alle link-nodes en alle script-nodes. Hierdoor kan de site crawler door alle gevonden links en scripts doorzoeken.

De tweede configuratie-optie zorgt ervoor dat er altijd binnen één website wordt gecrawld. Als de site crawler een externe link ontdekt, wordt deze link niet verder gecrawld.

De derde configuratie-optie kan een aantal limieten aan de site crawler zetten. Er zijn twee limieten, maximaal aantal downloaden en hoe diep er mag gecrawld. Voor de Quickscan module moet er door de hele website gecrawld worden, dus er wordt een download limiet van 1000 gebruikt en geen crawl-diepte limiet.

Bij laatste configuratie-optie wordt er een middleware[5] bij de site crawler toegevoegd. Volgens GuzzleHTTP is middleware een techniek om HTTP-request eerst naar een middleware doorsturen om te bewerken. Voor dit project wordt er een CurlHandler middleware gebruikt. Deze middleware kan de HTTP-request en response inzien en wijzigen.

2.4.5 Testen

Deze testgevallen zijn unittest van het performancemeetinstrument, het beschikbaarheidsmeetinstrument, het ontwerp en de cards. De unittesten zijn met PHPUnit gerealiseerd, het is een testing framework die Laravel framework biedt.

Testcase	Test beschrijving	Verwachte resultaat	Werkelijk resultaat
5	Beschikbaarheidsmeetinstrument waarneemt een niet bestaande link	Status moet een integer tussen 400 en 600 zijn.	V
6	Beschikbaarheidsmeetinstrument waarneemt een bestaande link	Status moet een integer tussen 100 en 400 zijn.	V
7	Performance-meetinstrument "onRequest" meet 1489680794.2765 bij timestamp en "onResponse" meet 1489680795.3129.	De laadtijd is 1034 milliseconde.	V
8	Gecrawlde link is een afbeelding	De waarde van de "resourcetype" constante is "Resource".	V

Tabel 3: Testgevallen meetinstrumenten

Regressietesten bij het testgeval 2 toont dat de fout is weggehaald.

[5] http://docs.guzzlephp.org/en/latest/handlers-and-middleware.html#middleware

2.4.6 Resultaat

Een probleem met mijn huidig ontwerp is dat als er een nieuw meetinstrument erbij moet komen, moeten er wijzigingen in de database komen. Het volgende meetinstrument kan een willekeurig aantal metingen opslaan en daarom moet er een verbetering in het ontwerp komen.

2.5 Sprint 5 – Verbetering in het ontwerp

2.5.1 Planning

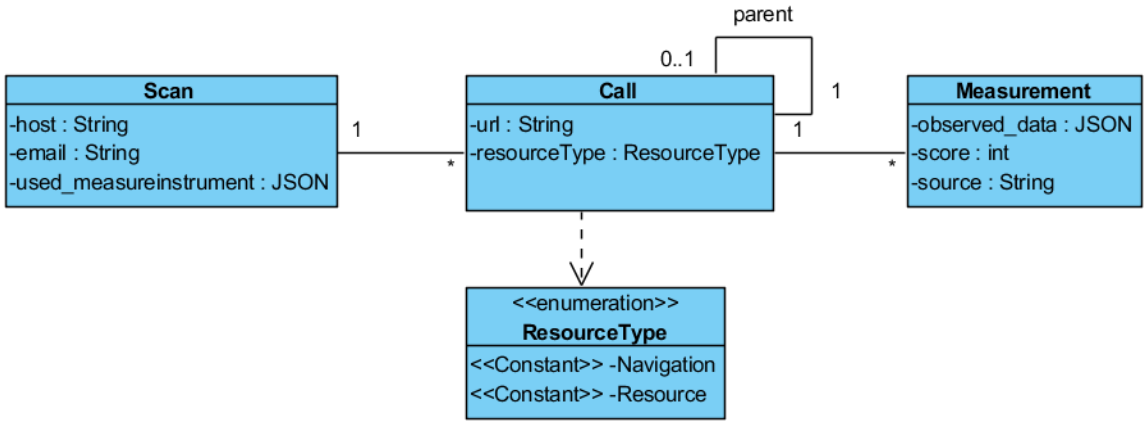
Het ontwerp verbeteren zodat er geen wijzigingen in de database moet komen als er een nieuw meetinstrument wordt toegevoegd.

Taken voor deze Sprint:

- Een nieuwe tabel aanmaken voor de metingen.
- Performancemeetinstrument card maken.

2.5.2 Verbetering in het ontwerp

Een probleem met het huidige ontwerp is dat als er een nieuw meetinstrument erbij moet komen, moeten er wijzigingen in de database komen. Het volgende meetinstrument kan een willekeurig aantal metingen opslaan en daarom moet er een verbetering in het ontwerp komen. In afbeelding 17 is het verbeterde ontwerp te zien.



Afbeelding 17 Klassendiagram metingen.

Het 'Scan' object houdt bij welke meetinstrumenten voor de scan zijn gebruikt, want die meetinstrumenten moeten de resultaten ophalen en verwerken.

Het 'Call' object gaat niet meer de metingen bijhouden en is vanaf nu alleen verantwoordelijk voor de links en de relatie tussen de links. De metingen worden bij het 'Measurement' object bijgehouden. Bij elke 'Call' wordt er een aantal 'Measurement' object gemaakt voor elke meetinstrument dat bij de scan hoort. Als er bij de scan vier meetinstrumenten hoort, worden er vier 'Measurement' objecten per 'Call' object gemaakt. 'Measurement' objecten houden de volgende attributen bij: de waargenomen metingen, de bron van de meting, score en een verwijzing naar het 'Call' object. De attributen status, content_size en response_time in het vorige ontwerp worden nu in observed_data opgenomen. Dit is in afbeelding 16 te zien. Het attribuut 'score' van 'Measurement' geeft een beoordeling aan de meting. Deze functionaliteit wordt later in het project geïmplementeerd.

id	call_id	source	observed_data	score
41	13	App\MIs\AvailabilityMi	{"Status": 200}	0
42	13	App\MIs\PerformanceMi	{"response_time": 2.5334680080414, "Content-Length": ["290476"]}	0
43	13	App\MIs\SecurityHeaderMi	{"X-Frame-Options": ["SAMEORIGIN"], "X-XSS-Protection": ["1; mode=block"], "X-Content-Type-Options": ["nosniff"], "Strict-Transport-Security": ["max-age=631138519"]}	0
44	13	App\MIs\PageIntegrityMi	{"Hash": "80253f8cbf952679ee7988f8876b4322f9bddc2e"}	0

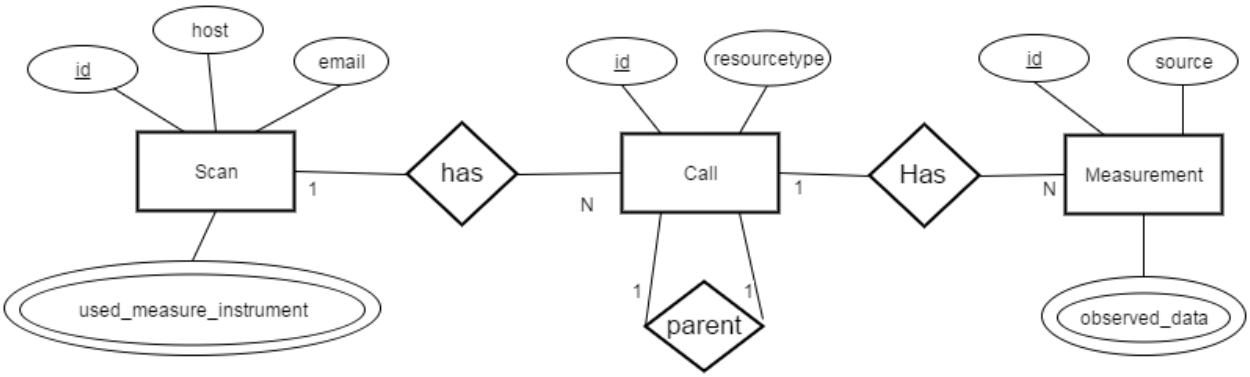
Afbeelding 18: Metingen van één call object ofwel een link.

Dit is een verbetering want het attribuut 'observed_data' in 'Measurement' is een JSON object en voordelen hiermee zijn:

- Meer dan een waarde per kolom, hierdoor moet er geen wijzigingen komen als er een nieuw meetinstrument erbij komt.
- MySQL ondersteunt JSON object, dus het is mogelijk om query's in het JSON object uit te voeren.

Een nadeel van JSON object datatype is dat de SQL-query's complexer wordt.

Hieronder is het databaseontwerp van het huidige ontwerp:

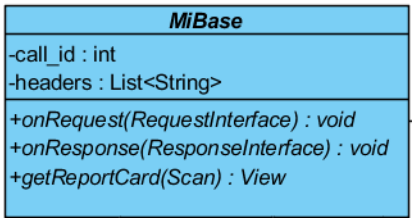


Afbeelding 19: ER-diagram van het huidige ontwerp

2.5.3 Performancemeetinstrument card

Ontwerp cards

De meetinstrumenten veranderen van verantwoordelijkheid geduurde de flow van de applicatie. Bij het begin van de scan worden ze geselecteerd voor de scan. Daarna tijdens de scan slaan ze de metingen in de database op. Als laatst halen de meetinstrumenten de metingen van de database op en de metingen worden naar de juiste card-sjabloon gestuurd. Dit is te zien in de volgende afbeelding.



Afbeelding 20: MiBase klasse

Alle meetinstrumenten hebben de nieuwe methode 'getReportCard(Scan)' gekregen en hierin worden de metingen door middel van SQL-queries opgehaald. Daarna worden de metingen naar de juiste view gestuurd.

De meetinstrumenten zijn ook verantwoordelijk voor het presenteren van het resultaat. Er is besloten om de code en query die de resultaten ophaalt, in dezelfde klasse van het meetinstrument houden. Want alles van een bepaald meetinstrument is in één plaats en dit vermindert het aantal klassen in de applicatie. Als er wijzigingen bij een bepaald meetinstrument moet komen, gebeurt dit in één plek.

Een nadeel van dit ontwerp is dat een meetinstrument klasse groot kan worden.

Card design

Het performance-meetinstrument card toont de drie traagste pagina's en de gemiddelde laadtijd van de gecrawlde website. Om te bepalen welke webpagina's de traagste zijn, worden de laadtijd van alle navigatie links met de bijbehorende bronlinks bij elkaar opgeteld. De SQL-query ziet er zo uit:

```
"SELECT c.uri, ROUND(SUM(json_extract(m2.observed_data, '$.response_time')),6)
+ json_extract(m.observed_data, '$.response_time') AS totalTime,
COUNT(c2.id) AS amountResources
FROM call AS c
INNER JOIN call AS c2 ON c.id = c2.parent_id
INNER JOIN measurement AS m2 ON c2.id = m2.call_id
INNER JOIN measurement AS m ON c.id = m.call_id
WHERE c.content_type_id = '1'
AND c2.content_type_id = '2'
AND m.source = 'App\MIs\PerformanceMi'
AND m2.source = 'App\MIs\PerformanceMi'
AND c.scan_id = '1'
AND c2.scan_id = '1'
GROUP BY c2.parent_id
ORDER BY `totalTime`
DESC LIMIT 3".
```

Deze SQL-query toont de drie traagste pagina's en geeft er ook bij uit hoeveel bronnen de webpagina bevat; dit is in afbeelding 10 te zien. Het 'json_extract' gedeelte in de query haalt de waarde van

‘response_time’ uit de JSON kolom ‘observed_data’ zodat die waarde vergelijkbaar wordt. Hierna wordt er met het resultaat van de query een card gemaakt.

2.5.4 Resultaat

Na het brengen van de wijzigingen in het ontwerp, wordt het gemakkelijker om nieuwe meetinstrumenten toe te voegen bij de applicatie. Dit omdat de tabel ‘Measurement’ heel flexibel is. Ook na de wijzigingen zijn alle testen geslaagd. De SQL-query van het beschikbaarheidsmeetinstrument moest ook gewijzigd worden, want de metingen zijn naar de ‘Measurements’ tabel verplaatst.

2.6 Sprint 6 – Security-headers-meetinstrument ontwikkelen

2.6.1 Planning

User story 05 – Security-headers meetinstrument Requirement R.05	
User story	De webapplicatie kan aan hand van HTTP-security-headers de veiligheid van een website bepalen.
Detail	1. Tijdens de scan wordt de zeven HTTP-security-headers waargenomen. 2. De volgende HTTP-security-headers worden waargenomen: a. HTTP Strict Transport Security (HSTS) b. HTTP Public Key Pinning Extension (HPKP) c. X-Frame-Options d. X-XSS-Protection e. X-Content-Type-Options f. Content-Security-Policy g. X-Permitted-Cross-Domain-Policies 3. Het resultaat wordt aan de gebruiker in een e-mail aan hand van cards gepresenteerd.
Taken	1. Security-headers-meetinstrument klasse aanmaken. 2. Security-headers-meetinstrument card ontwikkelen.

User story 06 – Webpagina-integriteitsmeetinstrument Requirement R.06	
User story	De webapplicatie kan de integriteit van de webpagina's van een website waarnemen.
Detail	1. Tijdens de scan wordt de integriteit van de webpagina opgeslagen om veranderingen op de webpagina waar te nemen. 2. De integriteit wordt met een cryptografische hashfunctie bepaald. 3. Het resultaat wordt aan de gebruiker in een e-mail aan hand van cards gepresenteerd.
Taken	1. Webpagina-integriteitsmeetinstrument klasse aanmaken. 2. Webpagina-integriteitsmeetinstrument card ontwikkelen.

De taken voor deze Sprint zijn:

- Security-headers-meetinstrument met card implementen.
- Webpagina-integriteitsmeetinstrument met card implementen.

2.6.2 Security-headers-meetinstrument

Het security-headers-meetinstrument neemt de HTTP-headers in verband met security waar. Als de security headers aanwezig zijn, kunnen ze de moderne webbrowser begrenzen, zodat ze bekende kwetsbaarheden gemakkelijk kan voorkomen.

Volgens de Open Web Application Security Project (OWASP)[6] zijn er op dit moment zeven security-headers, namelijk:

- **HTTP Strict Transport Security (HSTS)**, als deze header aanwezig is moet de webbrowser alleen met een HTTPS verbinding met deze website communiceren en nooit met een het onveilige HTTP-protocol.
- **HTTP Public Key Pinning Extension (HPKP)**, vertelt aan de webbrowser precies welke certificaten bij deze website hoort en dat de webbrowser alleen deze certificaten in de toekomst moet accepteren. Voorkomt een man-in-the-middle-aanval. Dit is een aanval waarij de informatie tussen twee communicerende partijen onderschept wordt zonder dat beide partijen ervan bewust zijn.
- **X-Frame-Options.** Als deze header goed ingesteld is, zorgt het ervoor dat de herkomst van de webpagina niet in een frame van een andere webpagina voorkomt. Voorkomt een clickjacking-aanval. Clickjacking is wanneer een aanvaller maak gebruik van een transparante of doorzichtige knop of link om een gebruiker te verleiden tot het klikken op een knop of link op een andere pagina.
- **X-XSS-Protection**, de juiste configuratie van deze header voorkomt een cross-site scripting(XSS) aanval. Een XSS is wanneer een ingevoerde tekst van de gebruiker niet juist wordt verwerkt en wordt door de webbrowser uitgevoerd.
- **X-Content-Type-Options**, deze header zorgt ervoor dat de webbrowser niet een MIMI-sniffing gaat uitvoeren. Dit is wanneer de webbrowser de bestanden gaat interpreteren als iets anders dan door het type die in de HTTP-headers is verklaard.
- **Content-Security-Policy(CSP)** vereist een zorgvuldige configuratie zodat het een grote invloed heeft hoe de webbrowser de webpagina laadt. CPS voorkomt verschillende web aanvallen.
- **X-Permitted-Cross-Domain-Policies** is een XML-bestand dat geeft een webclient, zoals Adobe Flash Player of Java Runtime Environment(niet beperkt tot deze voorbeelden) toestemming om data van een andere domain te halen.

Het security-headers-meetinstrument controleert of deze zeven security-headers aanwezig zijn. De zeven headers bevinden zich in de HTTP-response en dit meetinstrument noteert alleen de aanwezige headers.

Security-headers-meetinstrument card

Op dit moment toont het security-headers-meetinstrument card de aanwezigheid van de HTTP-security-headers. Het doel van dit meetinstrument is om de gebruiker te informeren dat belangrijke HTTP-security-headers ontbreken of aanwezig zijn. Later in de toekomst wordt er een beoordeling gegeven op de waarde van de securty-header.

Op dit moment wordt er een lijst met alle zeven HTTP-security-headers met hun status gepresenteerd. De niet gevonden HTTP-security-headers krijgen de status “Not found” en de anderen krijgen de status “Found”. Deze informatie wordt in de “DualColumnList” weergegeven.

[6] https://www.owasp.org/index.php/OWASP_Secure_Headers_Project

2.6.3 Webpagina-integriteitsmeetinstrument

Het webpagina-integriteitsmeetinstrument houdt bij de integriteit van het body alle HTTP-response. Hiermee is het mogelijk te zien welke webpagina's zijn gewijzigd. Het bepalen van bestandsintegriteit wordt met een cryptografische hashfunctie gerealiseerd. Er wordt de volgende hashfuncties voor integriteit gebruikt:

- CRC ("Cyclic redundancy check"), is een van de eerste data-integriteit controle standard en het is nu verouderd.
- MD5 ("Message Digest #5"), is de meest voorkomende data-integriteit controle standard, maar volgens best practices is het niet aanbevolen om deze hashfunctie te gebruiken.
- SHA ("Secure Hash Algoritm")[7] is de tweede meest voorkomende data-integriteit controle standaard in gebruik. Er zijn verschillende soorten SHA hashfuncties en volgens best practices zijn SHA-1 en SHA-2 het meest geschikt voor het bepalen van data-integriteit controle.

Voor dit project is de cryptografische hashfunctie SHA-2 gekozen. Ik heb voor SHA-2 gekozen, want het een best practice is bij het bepalen van bestandsintegriteit. Er wordt de variant SHA-224 en deze heeft een hash-lengte van 224 karakters.

Webpagina-integriteitsmeetinstument card

Het webpagina-integriteitsmeetinstument geeft aan welke pagina's veranderd zijn. Dit is alleen mogelijk als er een voorgaande scan is, want er wordt een vergelijking tussen de huidige scan en de voorgaande scan gedaan.

Het meetinstrument haalt eerst alle links van de huidige en de voorgaande scan op. Daarna wordt elke link van de huidige scan met de identieke link van de voorgaande scan vergeleken. Als de hash-waarde niet identiek is, wordt deze link aan de gebruiker met de status "Page content changed" gepresenteerd.

2.6.4 Resultaat

Het toevoegen van de twee meetinstrumenten ging gemakkelijk, want er moest geen wijziging in het ontwerp komen.

[7] http://www.filetransferglossary.com/category/integrity/

2.7 Sprint 7 – Tonen van Trends

2.7.1 Planning

User story 07 – Trends Requirement R.09	
User story	De resultaten van de scan ook gebruikt om met eerdere scans te vergelijken.
Detail	1. De resultaten van de huidige scan wordt met de voortgaande scans vergelijken. 2. De vergelijking wordt als een grafiek of staafdiagram gepresenteerd.
Taken	1. Vastleggen welk meetinstrument wel of niet een trend card gaat tonen. 2. Chart-tool bij de applicatie toevoegen, zodat de tool grafieken gaat genereren.

Taken voor deze sprint zijn:

- Ontwerp van de trends vastleggen.
- Chart-tool zoeken.
- Meetinstrumenten-ontwerp bijwerken zodat er ook trends kunnen gemaakt worden.

2.7.2 Tonen van trends

Een eis van de opdracht is ook om de resultaten van de voorgaande scans in de volgende reportages (requirement R.09) utiliseren. Trend is de verzameling van resultaten van meer dan één meting van dezelfde website. Het toevoegen van voortgaande scans in de rapportage laat de gebruikers trends van de security en performance aspecten zien, bijvoorbeeld verbetering in performance. Niet alle meetinstrumenten hoeven de trends te tonen. De volgende tabel geeft aan wanneer de trends in de e-mail zal verschijnen.

Meetinstrument	Presentatie-vorm	Eerste scan	Tweede scan of meer
Beschikbaarheidsmeetinstrument	Resultaat	V	V
	Trends	X	V
Performance-meetinstrument	Resultaat	V	V
	Trends	X	V
Security-headers-meetinstrument	Resultaat	V	V
	Trends	X	X
Webpagina-integriteitsmeetinstrument	Resultaat	X	V
	Trends	X	X

Tabel 4: Meetinstrument met hun presentatie-vorm.

Het beschikbaarheidsmeetinstrument toont bij de eerste scan alleen de resultaten van die scan. Als er weer een scan op die website wordt gedaan, dan verschijnt in de e-mail de resultaten van die scan en ook de trends van twee scannen. Het webpagina-integriteitsmeetinstrument kan alleen trends-card maken, want het heeft resultaten van meer dan 1 scan nodig.

De trends worden in grafieken gepresenteerd. De gebruiker krijgt de scanresultaten via e-mail en daarin mag er geen Javascript code bevatten. Met behulp van Javascript kan er gemakkelijk een grafiek genereren, maar Javascript code wordt automatisch door de e-mailcliënt geblokkeerd, want het kan onveilig zijn. Een oplossing hiervoor is om de grafiek aan de serverkant genereren en een afbeelding in de e-mail toevoegen.

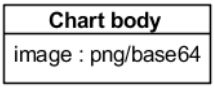
C-PChart

Er zijn weinig opensource en gratis serverside chart tool op Packagist. Packagist is de PHP online repository voor opensource PHP library. Er is voor PChart gekozen, want het is de meest populairste PHP serverside chart library op Packagist.

Na het uitproberen van deze tool blijkt dat het een geschikte tool is met verschillende functionaliteiten. C-PChart maakt het eenvoudig om grafieken te genereren en deze grafieken worden in de e-mail toegevoegd. De tool levert de grafieken als een afbeelding en die afbeelding wordt in een card toegevoegd.

Ontwerp cards bijwerken

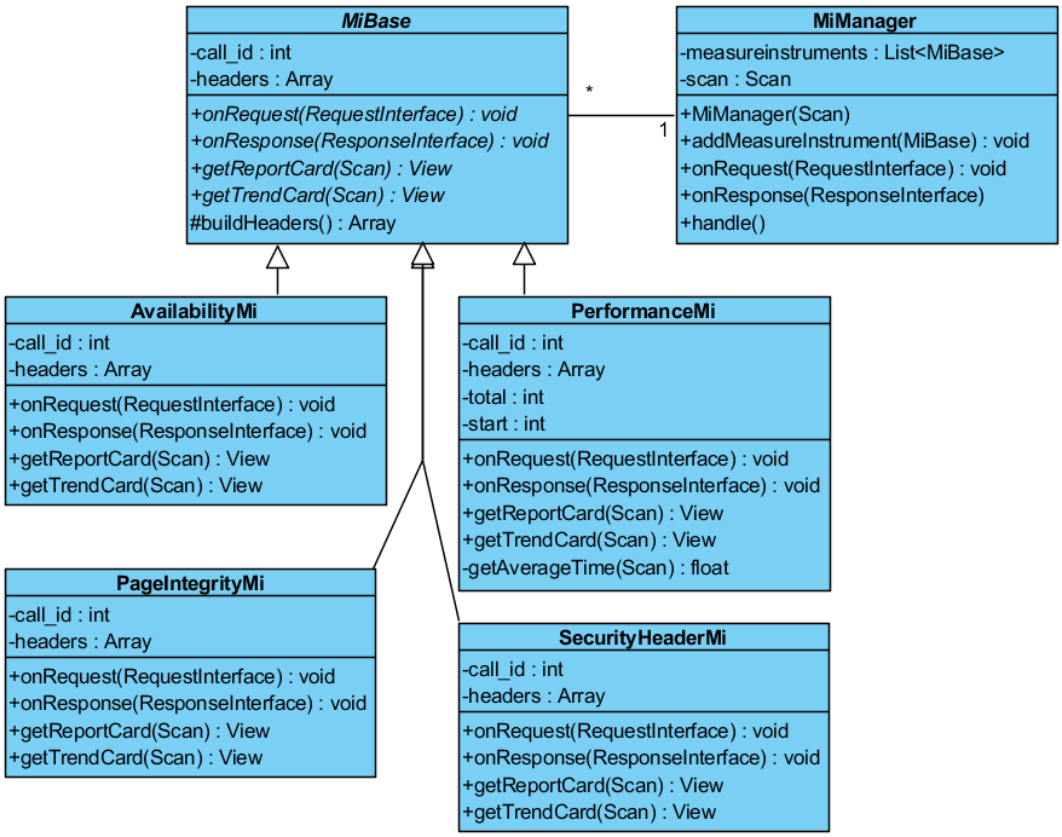
Er moet een nieuwe card-sjabloon voor de afbeeldingen.



Afbeelding 21: Card-onderdeel voor het tonen van afbeeldingen

Het ‘Chart Body’ object verwacht een afbeelding en die moet in Base64 gecodeerd zijn. Een alternatief was om een link(verwijzing) naar de afbeelding geven, maar sommige e- mailcliënten kunnen de link blokkeren. Dit object hoort bij de body variabele in het “Card” object die in paragraaf 2.4.4 is geïllustreerd.

2.7.3 Ontwerp trends



Afbeelding 21: Klassendiagram meetinstrumenten

Alle meetinstrumenten hebben de mogelijkheid om de methode 'getTrendCard(scan)' te gebruiken om een trend card te maken. Bij het maken van een e-mail, worden van alle meetinstrumenten die tijdens de scan gebruikt de 'getReportCard' en de 'getTrendCard' methode geroepen. Als een meetinstrument een card kan aanmaken krijgt de e-mail de card ervan, anders wordt er geen card geretourneerd.

Bij het aanmaken van een trend card, wordt er maximaal de tien laatste voortgaande scan met dezelfde url opgehaald. Hiermee wordt de trend bepaald.

3.7.4 Resultaat

De trend card hergebruikt verschillende principes van de report card. Er wordt dezelfde card onderdelen (afbeelding 16) gebruikt, ze bevinding zich in dezelfde klasse en het is optioneel om een te maken.

2.8 Sprint 8 – Tonen van Trends

2.8.1 Planning

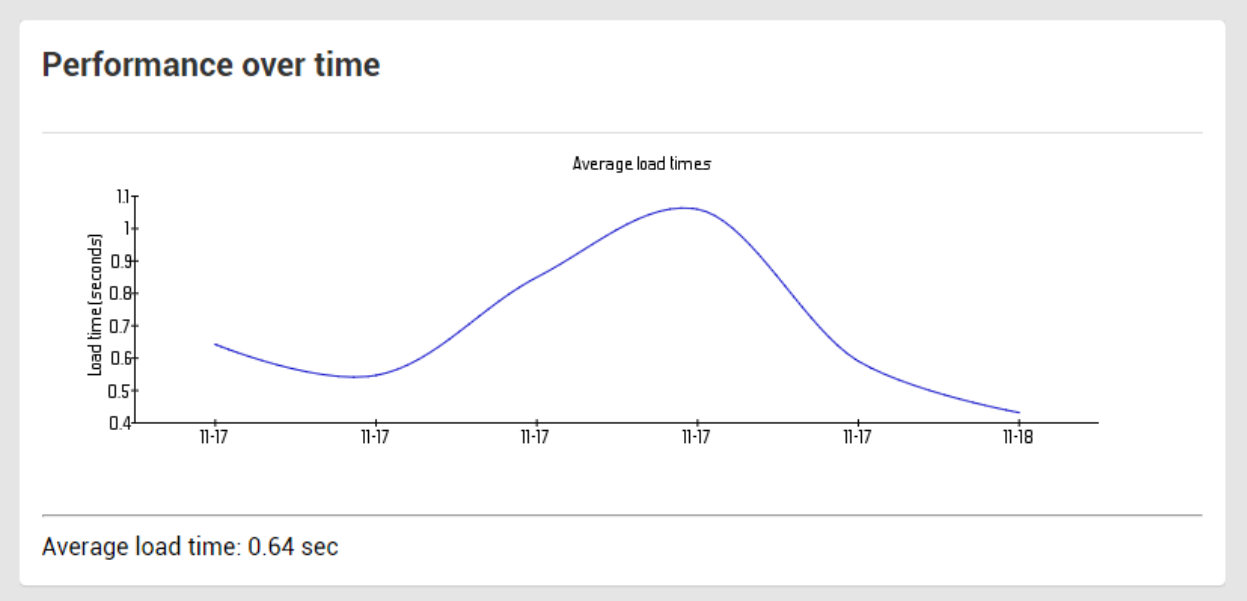
Taken voor deze Sprint zijn:

- Performance-meetinstrument card ontwikkelen.
- Beschikbaarheidsmeetinstrument ontwikkelen.
- Security-headers card ontwikkelen.

2.8.2 Trend cards

Performance-meetinstrument trend card

Het performance-meetinstrument trend card toont de gemiddelde prestatie van de website over tijd. Deze informatie wordt in een grafiek gepresenteerd en er wordt maximaal tien voortgaande uitgevoerde scan gebruikt. Dit is in de volgende afbeelding te zien.



Afbeelding 22: Performance-meetinstrument trends card

Om deze grafiek te maken wordt de laatste zes scans die dezelfde URL hebben als de huidige scan opgehaald. Daarna wordt per scan de gemiddelde laadtijd berekend. De volgende SQL query berekent de gemiddelde laadtijd van de eerste scan:

> *"**SELECT AVG(json_extract**(m.observed_data, '.response_time')) **AS** average_time **FROM** measurement m **WHERE** m.scan_id **IN** (**SELECT** id **FROM** call c **WHERE** c.scan_id = 1)**AND** m.source = 'App//Mi//PerformanceMi' ;"*

Hierna worden de resultaten in een grafiek gepresenteerd. De datum van de meting staat in de x-as en de gemiddelde laadtijd staat in de y-as van de grafiek.

Beschikbaarheidsmeetinstrument trend card
Het beschikbaarheidsmeetinstrument trend card toont het aantal linkrot van de laatste tien scans. Dit wordt in een staafdiagram voor de gebruiker gepresenteerd. Er wordt maximaal tien voorgaande scan voor het staafdiagram gebruikt. De query om het aantal linkrot bij een scan is:

> *"**SELECT COUNT**(m.id) AS amount_of_bad_pages **FROM** measurements **AS** m **JOIN** call **AS** c **ON** m.call_id = c.id **WHERE** c.scan_id = 1 **AND** m.source = 'App//Mi//AvailabilityMi'**AND** json_extract(m.observed_data, '$status') >= 400;"*

De datum van de meting staat in de x-as en het aantal linkrot staat in de y-as van de grafiek.

Security-headers card
Het Security-headers-meetinstrument trend card toont of er een verandering in de HTTP-security-headers is. Er wordt alleen met de voortgaande scan vergelijkt of er een verandering is.

Eerst wordt er gecontroleerd of de waarde van een HTTP-header is gewijzigd. Als het gewijzigd is, wordt dat met de status 'Changed' getoond. Vervolgens wordt er gecontroleerd of een HTTP-header is verwijderd of er een nieuwe is bijgekomen. Dan wordt de status 'New HTTP header' of 'HTTP header removed'.

2.8.3 Testen
Naast de unittesten worden er ook exploratory testen gedaan. Exploratory testing is de gekozen testmethodiek, omdat hiermee kan er de kwaliteit van de applicatie bepalen. Het doel van deze testen is dat het systeem onder normale omstandigheden correct werkt.

Exploratory testen - Logische testgevallen

Testcase	Actie	Te verwachte resultaat
1	Vul een website en geen e-mail adres in het formulier om een scan te starten.	Er verschijnt een melding dat de e-mail ontbreekt en er is geen scan gestart.
2	Vul een website en een e-mail adres in het formulier om een scan te starten.	Er verschijnt een melding dat er een scan is ingeplant en binnen 15 minuten heeft de e-mail adres een e-mail ontvangen.
3	Run een scan op een website die nooit is gescand.	In de e-mail verschijnt er drie cards.
4	Run een scan op een website die eerder is gescand.	In de e-mail verschijnt er zes cards.
5	Maak een grafiek met de volgende parameters: -X-as: 1-11-16, 2-11-16, 3-11-16, 4-11-16, 5-11-16 -Y-as: 0.5, 1.0, 1.5, 2.0, 2.5	Een stijgende lineaire lijn in de grafiek, de x-as bevat vijf datums en de y-as gaat van 0 t/m 2.5.
6	Er zijn vier meetinstrumenten voor een scan gebruikt.	Elke "Call" object bevat vier "Measurement" objecten.

Tabel 5: Exploratory testgevallen

2.8.4 Resultaat

Na het uitvoeren van de exploratory testen en de unittesten zijn ze allemaal geslaagd. Naast het uitvoeren van de testen, ziet de front-end en de e-mail opmaak van de applicatie goed uit. De applicatie functioneert naar behoren en kan in gebruik genomen worden.

Deel 3: Evaluatie

In dit hoofdstuk worden de producten, beroepstaken en het proces geëvalueerd.

3.1 Evaluatie producten

Aan het eind van de afstudeerperiode heb ik een prototype applicatie te kunnen opleveren aan de wensen van Protoneers. Dit product bestaat uit: een site crawler, vier meetinstrumenten, een e-mail module, een database, een front-end en het is gemakkelijke om uit te breiden. Deze componenten worden allemaal in de Laravel framework gerealiseerd. De site crawler kan een scan uitvoeren op een MKB website en de resultaten worden in de database opgeslagen. Hierna worden de resultaten geanalyseerd en naar de gebruiker ge-e-maild.

Ik ben heel tevreden met het product, want het is een prachtig applicatie die een website kan scannen voor security en performance aspecten. Ik heb veel geleerd tijdens het realiseren van dit product.

3.2 Procesevaluatie

Ik ben redelijk tevreden over het verloop van het proces. Ik heb deze afstudeeropdracht gekozen, want ik wilde iets doen die ik nooit heb gedaan waardoor ik veel ervan kan leren. Dit betekent dat ik veel moeilijkheden heel vroeg in de afstudeerperiode tegenkwam. Het gevolg hiervan is dat ik soms achter de planning staat, maar bij het einde van de afstudeerperiode heb ik de planning ingehaald.

Planning en uitvoering

Ik duurde langer over het onderzoek naar site crawler, het gebruik van de site crawler in Laravel en het tonen van trends in e-mail dan er was gepland.

Het onderzoek naar site crawler duurde langer, want ik onderschatte de complexiteit van het kiezen en uitproberen van de site crawlers. Ik zou voor de volgende keer meer tijd voor deze soort taken inplannen.

Het gebruik van de site crawler in Laravel duurde langer gepland, want ik moest een wijziging in het ontwerp doen om de applicatie uitbreidbaar te maken en dat was niet gepland. Ik had niet goed rekening met de uitbreidbaarheid van de applicatie tijdens het eerste ontwerp gehouden.

Het tonen van trends in de e-mail was ook lastig, want het betreft heel veel informatie. Sommige trend-cards moeten maximaal tien voorgaande scans gebruiken en hierdoor kunnen de SQL-query's ingewikkeld worden.

Integendeel, het implementeren van de laatste twee meetinstrumenten had ik over korter tijd dan er was gepland gedaan. Na de verbetering in het ontwerp is het eenvoudige en snel om een nieuw meetinstrument te implementeren.

Methodiek

Ik ben tevreden over de gekozen Scrum methodiek, want hierdoor was het gemakkelijk om te gaan met de veranderingen en de vertraging in het project. De Standup Meeting zorgt ervoor dat er altijd contactmomenten geweest met de opdrachtgever en begeleiders om over het project te bespreken.

Voor de volgende keer zou ik op de dezelfde manier te werk gaan, omdat dit project is goed gelopen. Ook is de opdrachtgever tevreden met de opgeleverde applicatie.

3.3 Beroepstaken evaluatie

1.3 Selecteren van standaardsoftware (Niveau 2)

Voor dit project heb ik een onderzoek uitgevoerd om het juiste open source softwarepakket te kiezen. Ik heb een stakeholder geïnterviewd voor het verzamelen van de selectiecriteriums. Daarna heb ik een longlist opgesteld en aan hand van de selectiecriteriums is de longlist tot een shortlist verkort. Vervolgens heb ik alle open source softwarepakketten uitgeprobeerd en er bleef één over.

Ik heb deze beroepstaak op niveau 2 gehaald, omdat de gekozen softwarepakket uit verschillende componenten bestaat. Tijdens het selectietraject was er eerst een longlist die naar een shortlist verkort moest worden.

2.1 Opstellen gegevensmodel voor een database (Niveau 3)

Voor dit project heb ik een ER-diagram voor het gegevensmodel ontworpen. Het gegevensmodel bevat verschillende één op veel relatie, n-aire verbanden en ook verschillende meerwaardige attributen.

Deze beroepstaak is op niveau 3 gehaald, omdat het gegevensmodel heeft verschillende complexe structurele verbanden.

3.2 Ontwerpen systeemdeel (Niveau 3)

Voor dit project is er de ontwerptechniek UML gebruikt om verschillende ontwerpen van de applicatie te modeleren. Het betreft de objectgeoriënteerde applicatie met een front-end user interface en de architectuur van de applicatie.

Deze beroepstaak is gedeeltelijk op niveau 3 gehaald, omdat het betreft niet het ontwerpen van een applicatie met een redelijk complexe algoritmiek.

3.4 Bouwen applicatie (Niveau 4)

Dit project is gerealiseerd met de Laravel Framework en er is tijd besteed om Laravel Framework te leren kennen, want deze framework wordt binnen Protoneers gebruikt. Er is ook met een objectgeoriënteerde programmeertaal gewerkt, namelijk PHP. Vervolgens is er ook met een geavanceerde programmeertool gewerkt, deze is NetBeans. De gemaakte codebase wordt door een versiebeheertool beheerd, namelijk Bitbucket en deze gebruikt met het Git-protocol. Als laatst is er veel rekeningen gehouden met de uitbreidbaarheid van de applicatie. De applicatie is eenvoudig uit te breiden en vergt geen wijziging in de structuur van de applicatie.

Ik heb deze beroepstaak op niveau 4 gehaald, omdat het EM24ME project is in een complext omgeving ontwikkeld en is afhankelijk van verschillende opensource softwarepakketten.

3.4 Uitbreidingsmogelijkheden voor de applicatie

Meer meetinstrumenten toevoegen
Een mogelijk meetinstrument is om de Nikto2[8] webserver scanner in dit project te integreren. Nikto is een webserver scanner die uitgebreide security testen uitvoert. Nikto scant voor meer dan 6000 potentieel gevaarlijke bestanden of programma's, controleert voor verouderde versie webserver en versie specifieke problemen op meer dan 270 webservers.

Meetinstrumenten kiezen voor een scan
Een bruikbaar functionaliteit is om de gebruikers de mogelijkheid om specifieke meetinstrumenten te gebruiken voor een scan geven. Er zijn gevallen waar een gebruiker alleen een performance scan wil uitvoeren. Hiervoor is er ook een interface nodig voor de meetinstrumentselectie.

Periodiek scannen
Een website periodiek scannen is handig in sommige gevallen. Gebruikers krijgen de mogelijkheid om hun website om elke uur t/m elke dag te scannen. Hiervoor krijgen de gebruikers een interface om de frequentie te kiezen en ook welke meetinstrumenten er bij de periodieke scan hoort.

Scenario's doorlopen
Het kan zijn dat website-eigenaren niet hun hele website willen crawlen en alleen een aantal links willen crawlen. Hiervoor kan de gebruiker via een interface het gewenste scan-scenario aanmaken. Een scenario is een lijst van links dat de site crawler moet doorlopen. Bij het aanmaken van scenario's kunnen de gebruikers ook kiezen welke meetinstrument bij de scan hoort. Gebruikers kunnen ook meerdere scenario's hebben.

Naar PHP 7 upgraden
Op dit moment draait de applicatie op PHP 5.6 en upgraden naar PHP 7 heeft een aantal voordelen. Volgens de Zend-website [9] is PHP 7 minstens twee keer sneller dan PHP 5.6 en gebruikt ongeveer twee keer minder systeemresources. Laravel versie 5 ondersteunt PHP 7, dus er moeten geen wijzigingen in de huidige code komen.

[8] https://cirt.net/nikto2
[9] http://www.zend.com/en/resources/php-6-need-to-know-about-php-7

Bijlagen

Bijlage A – Literatuurlijst

Boeken

Tré, G. D. (2007). Principes van databases. Amsterdam: Pearson Education.

Rozanski, N., & Woods, E. (n.d.). Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Edition 2.

Jason Gilmore, W. (n.d.). Easy Laravel 5: A hands on introduction using real-world project.

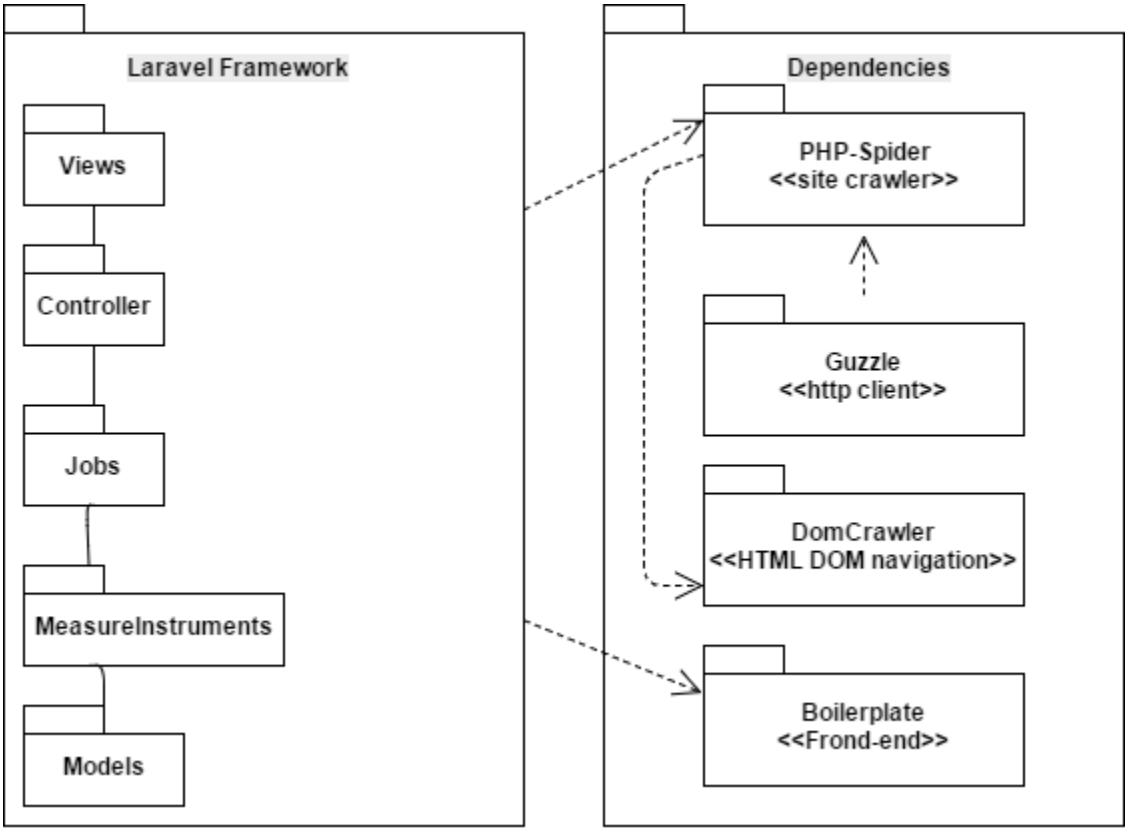
Websites

HTTP/1.1: Status Code Definitions. (n.d.). Retrieved November 25, 2016, from https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

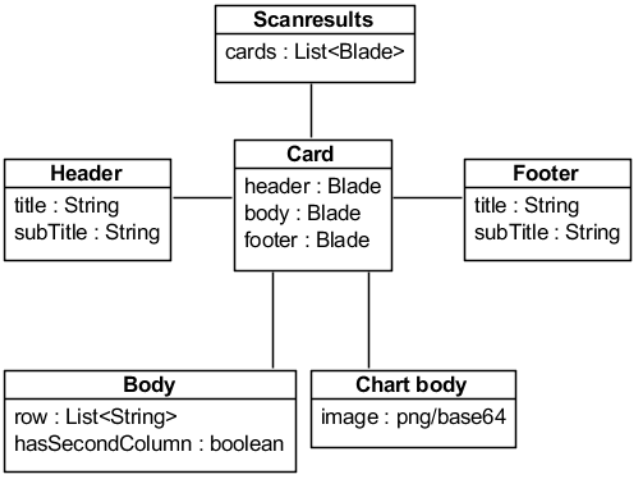
Bijlage B - Woordenlijst

Naam	Beschrijving
Site crawler	Een site crawler is een bot die een website op een methodische en automatiseerde manier doorbladert.
HTTP request	Met een HTTP request kan een webbrowser een aanvraag op een webserver doen. (Voorbeeld in bijlage E)
HTTP response	Via de HTTP response krijgt de webbrowser het antwoord op de HTTP request. (Voorbeeld in bijlage E)
Laravel Framework	Laravel is een PHP web framework en is bedoeld voor het ontwikkelen van web applicatie volgens MVC architectural pattern.
Meetinstrument	Een meetinstrument is een module die één security of één performance aspect meet of waarneemt.
Quickscan	Gebruiker vult in e-mail en website en binnen 15 ontvangt de gebruiker een e-mail met de resultaten.
Trends	Verzameling de resultaten van meer dan één meting, die informatie geeft over de voortgang van de website.
Composer	Dependency manager voor PHP.
Packagist	Grootste repository voor Composer.

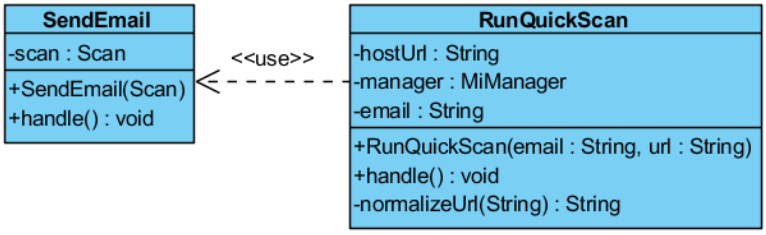
Bijlage C – UML diagrammen



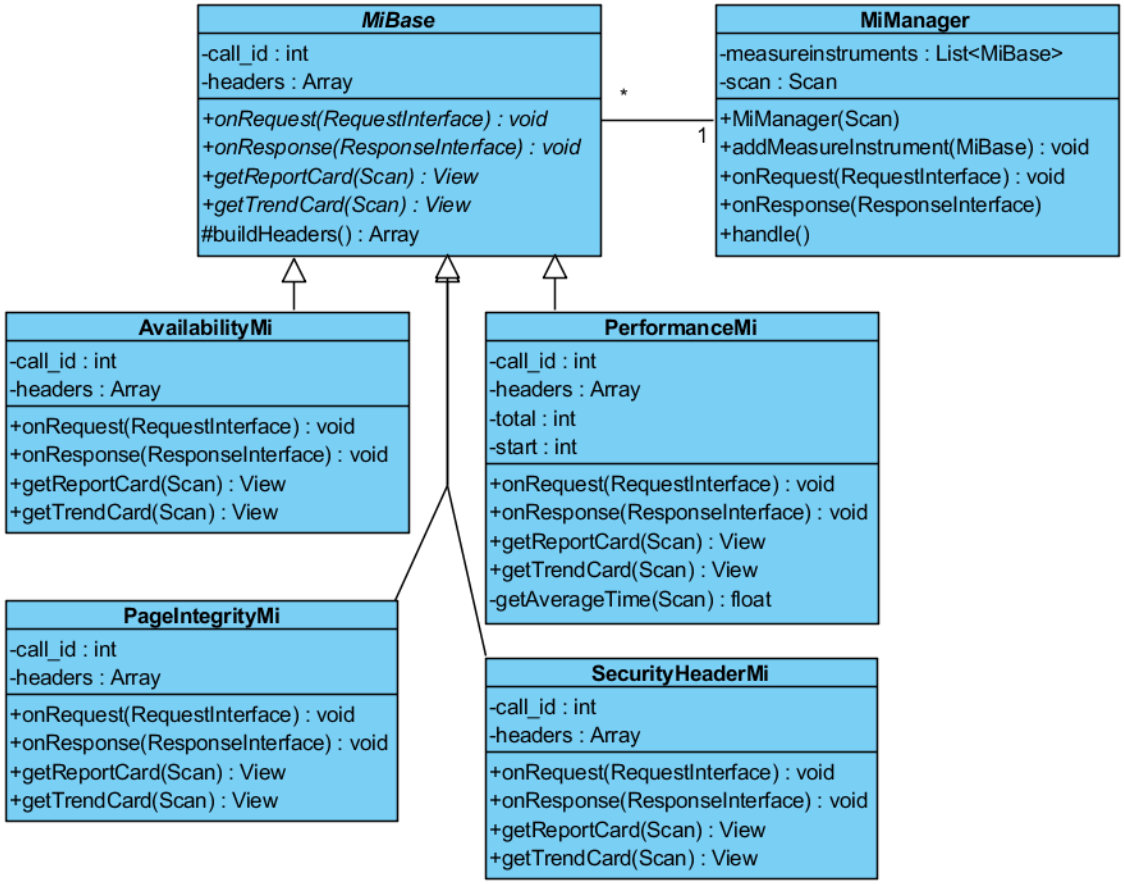
Afbeelding C.1: Package diagram; systeem met dependencies.



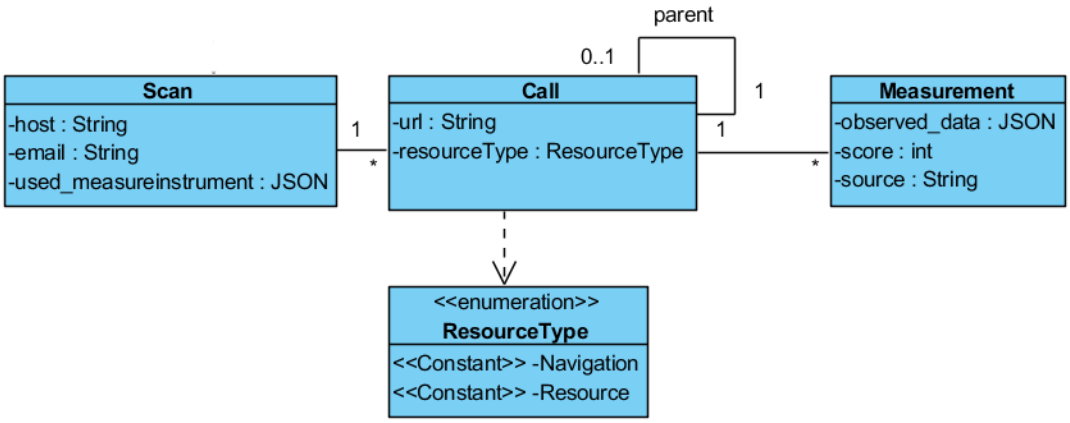
Afbeelding C.2: Klassen in het 'Views' package



Afbeelding C.3 Klassen in het 'Jobs' package



Afbeelding C.4 Klassen in het 'MeasureInstruments' package



C.5 Klassen in het 'Models' package

Bijlage D – HTTP request en HTTP response

▼ General

Request URL: https://www.google.nl/
Request Method: GET
Status Code: 200
Remote Address: 216.58.212.163:443

▼ Response Headers

alt-svc: quic=":443"; ma=2592000; v="36,35,34"
cache-control: private, max-age=0
content-encoding: gzip
content-type: text/html; charset=UTF-8
date: Fri, 25 Nov 2016 13:29:35 GMT
expires: -1
server: gws
status: 200
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block

▼ Request Headers

:authority: www.google.nl
:method: GET
:path: /
:scheme: https
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
accept-encoding: gzip, deflate, sdch, br
accept-language: en-US,en;q=0.8,nl;q=0.6
cache-control: max-age=0
cookie: SID=ewOSo5a4gxCEQ0wUWLVs1jfy6gGdhVAFX-xid3sNIOV3cayNG8Dk3KcrnVeD1h-L9kZrRQ.; HSID=AxnjVOP1187n0D3e-; SSID=ABxn0ZEwEy4cvYiElxQsUnuFA; DV=wo2G6NLr850aBMhGFNhXdO0no1A3sQI
dnt: 1
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.87 Safari/537.36
x-client-data: CJa2yQEIorbJAQjEtskBCIqSygEIqZ3KAQ==

Afbeelding D.1: HTTP-request en HTTP-response van google.nl