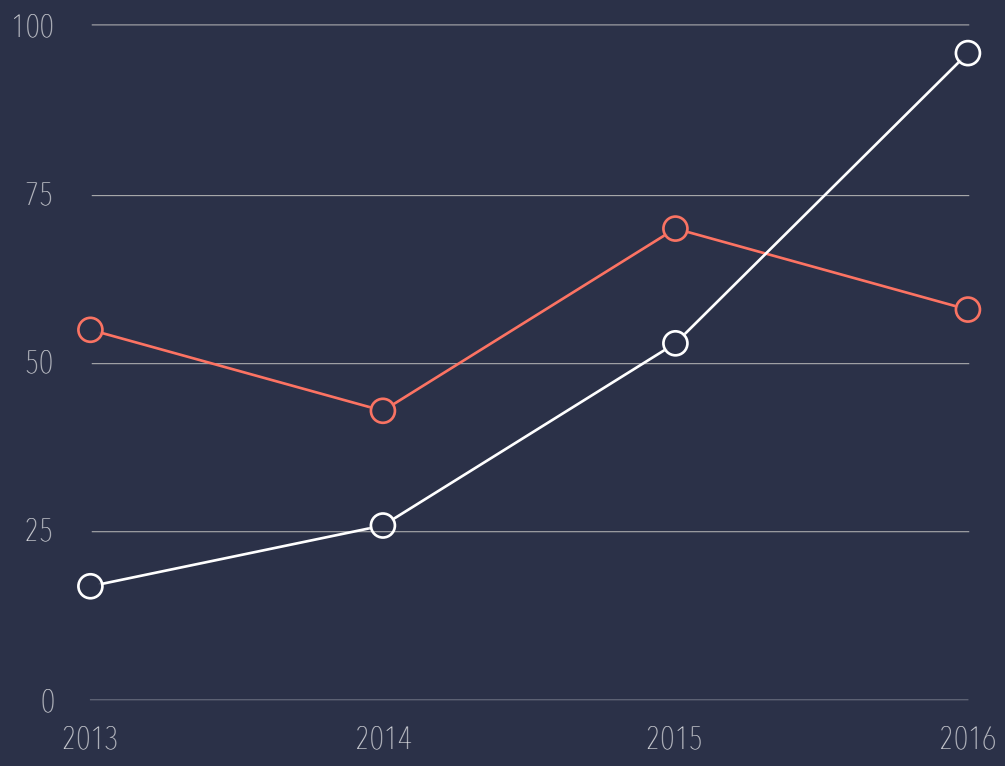
SOCIALR

Ontwikkelen van een statistieken webapplicatie



AFSTUDEER DOSSIER

DON KOOIJMAN

10053034
03-06-2016
Versie 1

Referaat

Dit verslag is geschreven in het kader van het afstudeerproject van Don Kooijman voor de opleiding Informatica aan de Haagse Hogeschool. De afstudeeropdracht is uitgevoerd van 1 februari 2016 tot 3 juni 2016.

In opdracht van het bedrijf SOCIALR is er een applicatie gemaakt die statistieken moet weergeven. De data in deze statistieken is afkomstig van verschillende social media platformen.

Descriptoren:
- Scrum
- Laravel
- MySQL
- JSON
- OAuth
- Javascript
- Dashboard

Voorwoord

De afgelopen maanden heb ik een afstudeerstage gelopen bij SOCIALR. Om de stage te beschrijven heb ik dit verslag geschreven. Ik vond het erg moeilijk en heb er hard aan moeten werken, maar ik denk dat het resultaat goed beschrijft wat ik de afgelopen maanden gedaan heb.

SOCIALR heeft mij deze plek aangeboden en daar wil ik ze voor bedanken. De onwijs fijne sfeer en de leuke opdracht hebben ervoor gezorgd dat ik geen probleem had met de uitvoering. Het geduld en begrip dat jullie hebben getoond wanneer iets niet helemaal lekker liep, was ontzettend fijn.

Ik wil ook graag Rianne Bechet en Okan Zor bedanken. Zij zullen dit verslag beoordelen en hebben gedurende de stage feedback gegeven op het verslag. Hierdoor heb ik het kunnen maken tot wat het nu is.

Don Kooijman

3 juni 2016, Den Haag

Inhoudsopgave

1. Inleiding

In februari 2016 ben ik begonnen met mijn afstudeerproject bij SOCIALR in Amsterdam. Ik heb mij bezig gehouden met het bouwen van een op maat gemaakte web applicatie.

Dit verslag dient inzicht te geven hoe ik de weken tijdens het afstuderen heb ingevuld, welke keuzes ik gemaakt heb en hoe ik mij aan een software ontwikkel methode heb gehouden.

Het verslag is geschreven zodat het voor iedereen leesbaar is. Ondanks dat blijft het de beschrijving van een proces waarbij er software ontwikkeld is. Daardoor kan het lastig te lezen zijn door personen die geen kennis hebben van het vak software engineering.

Hoofdstuk 2 t/m 4 gaan over het bedrijf, opdracht en de aanpak. Hoofdstuk 5 t/m 10 gaan over de uitvoering. In hoofdstuk 11 evalueer ik verschillende onderdelen.

Op basis van dit verslag moet aan te tonen zijn of ik het vak: ‘software engineering’ goed genoeg beheers om een diploma te ontvangen.

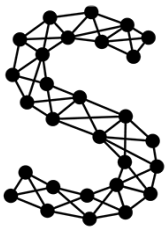
2. Organisatie

SOCIALR is een jong groeiend bedrijf gevestigd in Amsterdam. Het is ontstaan in april 2015 uit een fusie van twee bedrijven. Beide bedrijven hadden dezelfde visie en hebben besloten samen verder te gaan als SOCIALR.

SOCIALR heeft twee takken: services en software. De servicetak verleent de volgende diensten: online advertising, online strategieën en adviezen, en website & app development. Enkele klanten van SOCIALR zijn: Kentucky Fried Chicken, Monsterboard, RTL Nederland en Hema.

De software-tak gaat om in eigen huis ontwikkelde software, deze software plaatst automatisch advertenties op online media zoals Facebook, Twitter, Google en LinkedIn. De software is in staat om 'slimme' analyses te doen over de geplaatste advertenties, om zo in de toekomst voor de juiste strategie te kiezen. De software analyseert bijvoorbeeld dat een advertentie met een blauwe achtergrond het beter doet dan die met een rode achtergrond, in het vervolg zal de software dan voor de advertentie met blauwe achtergrond kiezen.

Het team van SOCIALR bestaat uit 12 personen, waarvan 8 in de servicetak werkzaam zijn en de overige 4 in de software-tak.



SOCIALR

Figuur 2.1 - Logo SOCIALR.

3. Opdracht

De service-tak van SOCIALR verleent o.a. online advertising voor haar klanten, hierbij geeft SOCIALR haar klanten graag inzicht in de statistieken over de online advertising campagnes. Deze statistieken dienen online en real-time te zijn. Tot nu toe heeft SOCIALR dit gedaan met behulp van een externe partij: cyfe.com.

Cyfe is een online platform waarin je allerlei statistieken kunt bekijken van online tools die bedrijven gebruiken. Cyfe combineert de statistieken van talloze verschillende aanbieders op 1 plek met behulp van verschillende widgets. Voorbeelden van die aanbieders zijn Google Analytics, Facebook Ads en Youtube statistieken.

3.1 Probleemstelling

SOCIALR is niet helemaal tevreden met het gebruik van Cyfe, doordat het bepaalde nadelen met zich meebrengt en omdat het functionaliteiten en opties mist:

1. Kosten; Het gebruik van Cyfe kost €50.- per maand, dit betaalt SOCIALR op dit moment voor de klant aan Cyfe. Dat komt doordat ze het aanbieden als een service van hun zelf.
2. Functionaliteiten
   A. In Cyfe kun je kiezen voor een widget die één enkele waarde laat zien of een door hun samengestelde widget. Vaak laten deze samengestelde widgets kosten zien, dat wil SOCIALR niet zichtbaar maken aan haar klanten.
   B. Het is niet makkelijk om statistieken van een periode met dezelfde statistieken van een andere periode te vergelijken. Dit doet SOCIALR nu door screenshots te nemen en die naast elkaar neer te leggen, niet ideaal.
   C. SOCIALR moet nu zelf rapportages samenstellen met data die ze uit de tool halen, vervolgens mailen ze die. Een simpele mail statistieken/rapportage knop ontbreekt.
   D. SOCIALR moet nu voor iedere nieuwe klant een dashboard helemaal opnieuw instellen, dit kost erg veel tijd terwijl de dashboards vaak op het zelfde neerkomen. Ze hebben niet de mogelijkheid om een dashboard her te gebruiken.
3. Layout; De layout van cyfe.com is verouderd.
4. Styling; Er is geen mogelijkheid om de styling van een dashboard te wijzigen naar de styling van de klant of naar die van SOCIALR. Bijvoorbeeld het wijzigen van een logo.

3.2 Doelstelling

Het doel van de afstudeeropdracht is om ervoor te zorgen dat SOCIALR in staat blijft statistieken weer te geven aan hun klanten, maar dat de verschillende problemen die zij op dit moment met Cyfe ondervinden opgelost zullen worden.

3.3 Resultaat

Het resultaat van de opdracht zal een op maat gemaakte webapplicatie voor SOCIALR zijn, gebaseerd op de door hun geformuleerde requirements. Hierin zullen de ervaren problemen met Cyfe worden opgelost en zal het voor SOCIALR niet meer nodig zijn om cyfe.com te gebruiken.

4. Aanpak

Voordat ik met u het project in duik wil ik mijn aanpak voor het project bespreken. Hierin komen onder andere aan de orde de ontwikkelmethodiek, de tools die ik gebruik en het plan van aanpak.

4.1 Ontwikkelmethodiek

Om te zorgen dat het project zo goed mogelijk verloopt wordt er gebruik gemaakt van een software ontwikkelmethodiek. SOCIALR heeft al een fijne ervaring met scrum. Dit is de reden dat zij mij hebben gevraagd scrum te gebruiken.

Het gebruik van scrum sluit goed aan bij dit project. Scrum is namelijk een agile methodiek, wat betekent dat er in kleine iteraties gewerkt wordt. Requirements ophalen, ontwerpen, bouwen en testen komen in iedere iteratie (sprint) terug. Dit is handig wanneer voorafgaand het project er niet precies duidelijk is wat de wensen en eisen zijn. Wanneer er iets veranderd of iets niet naar wens uitgevoerd is, is dat direct zichtbaar.

Zoals ik al zei komen de verschillende activiteiten steeds weer terug, dit gebeurt tijdens de sprints. Deze sprints duren meestal 2,3 of 4 weken, in mijn geval 2 weken. De bedoeling is dat er aan het eind van iedere sprint een “shippable product” wordt opgeleverd. Dit betekent dat het product zowel gebouwd als getest is. Er zit niet meer en niet minder in, dan wat er gepland is voor de desbetreffende sprint.

Figuur 4.1 - Verloop scrum project.

Zoals u kunt zien in het bovenstaand figuur, wordt iedere sprint ingepland (sprint backlog) met verschillende backlog items. Deze backlog items zijn afkomstig uit de product backlog, die wordt opgesteld in sprint 0. Sprint 0 is de allereerste sprint. In deze sprint wordt de product backlog opgesteld en worden er andere activiteiten uitgevoerd, bijvoorbeeld het opstellen van het plan van aanpak. Deze activiteiten maken het verloop van de toekomstige sprints makkelijker.

Aan het eind van iedere sprint is er een sprint meeting. Tijdens deze sprint meeting presenteer ik het product, evolueer ik samen met de opdrachtgever de sprint en de product backlog en tot slot wordt de volgende sprint ingepland. Door deze meetings wordt de opdrachtgever betrokken bij het project en kan hij vlug ingrijpen als iets niet naar wens gaat.

Daarnaast wordt er iedere dag een daily scrum meeting gehouden, waarin kort met het team gedeeld wordt wat er die dag gedaan wordt, wat er mogelijk lastig kan zijn en of iemand mij daarbij kan helpen. Deze dagelijkse mini-meeting houd ik met mijn technische begeleider.

4.2 Tools

Naast de ontwikkelmethodiek wil ik ook het framework en de database techniek bespreken die ik in dit project ga gebruiken.

4.2.1 Framework

Bij SOCIALR worden alle backend projecten gedaan met Laravel. Laravel is een gratis open-source PHP framework ontwikkelt door Taylor Otwell.
Het is bedoelt voor web applicaties die gebruik maken van het model-view-controller (MVC) pattern.

Figuur 4.2 - Logo Laravel

Tijdens de sollicitatie was het al duidelijk dat SOCIALR wilde dat ik het project in Laravel zou doen. Dit zorgt ervoor dat zij nog verder aan het project kunnen werken wanneer ik weg ben. Mijn technisch begeleider is zeer ervaren met het framework waardoor ik goede begeleiding kan krijgen.

4.2.2 Database & ORM

In dit project zal ik gebruik maken van MySQL voor de database. MySQL wordt per default ondersteund door Laravel. De ORM mapper die in Laravel gebouwd zit, maakt ook gebruik van MySQL.

Tijdens het maken van het afstudeerplan werd duidelijk dat er met grote hoeveelheden data gewerkt zou worden, dit maakte het mogelijk interessant om naar andere database technieken te kijken, bijvoorbeeld nosql. Het bedrijf wees mij er op dat ze liever hebben dat ik de opdracht met MySQL doe. De reden hiervoor is dat zij met andere technieken geen ervaring hebben en ze geen problemen verwachten met het gebruik van MySQL.

Zoals ik al zei heeft Laravel een eigen ORM mapper ingebouwd: EloquentORM. Dit is ook de ORM mapper die ik zal gebruiken tijdens dit project.

4.4 Plan van aanpak

In de eerste weken van de afstudeerstage heb ik een plan van aanpak opgebouwd. Hierin kwamen o.a. de planning en risico's aanbod.

4.4.1 Globale planning

In totaal ben ik 18 weken bezig, waarvan ik 1 week vrij neem. Deze vrije week valt in de 5e sprint. Tijdens de eerste sprint (Sprint 0) vinden er meerdere requirement meetings plaats. In de vervolg sprints worden de requirements besproken tijdens de sprint meetings, aan het eind van iedere sprint.

Week	Datums	Sprints	Onderdelen
Week 1 **Week 2**	1 Feb 12 Feb	Sprint 0	• Plan van aanpak maken • Requirement meetings • Product backlog vullen • Kennismaking met framework
Week 3 **Week 4**	15 Feb 26 Feb	Sprint 1	• Bouwen van de basis applicatie (inloggen, dashboards bekijken etc.) • Sprint meeting 1
Week 5 **Week 6**	29 Feb 11 Maart	Sprint 2	• Koppelen Facebook Ads API • Sprint meeting 2
Week 7 **Week 8**	14 Maart 25 Maart	Sprint 3	• Voortgangsverslag inleveren • Bezoek begeleidend docent (Okan Zor) • Koppelen Google Analytics API • Sprint meeting 3
Week 9 **Week 10**	28 Maart 8 April	Sprint 4	• Bespreking concept afstudeerdossier • Koppelen Instagram Ads • Sprint meeting 4
Week 11 **Week 12** **Week 13**	11 April 29 April	Sprint 5	• Koppelen Instagram pages en Facebook Pages • Deze sprint neem ik een week vrij • Sprint meeting 5
Week 14 **Week 15**	2 Mei 13 Mei	Sprint 6	• Koppelen Twitter API • Sprint meeting 6
Week 16 **Week 17** **Week 18**	16 Mei 3 Juni	-	• Tussentijds assessment afstudeerdossier • Afronden project, scriptie maken. • 3 juni definitief verslag inleveren.

4.4.2 Detail planning

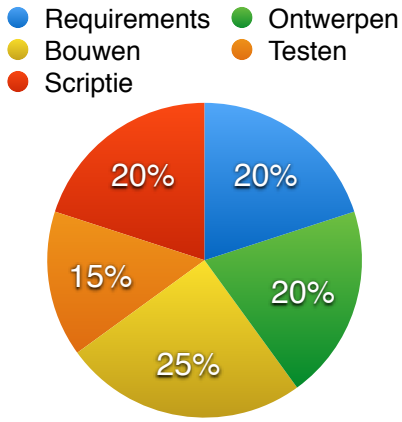
De detailplanningen worden in samenwerking met de opdrachtgever gemaakt. Dit gebeurt tijdens de sprint meetings van de voorgaande sprints of aan het begin van de desbetreffende sprint. In het verslag noteer ik ze altijd aan het begin van iedere sprint, zodat het verslag een duidelijke structuur behoudt.

4.4.3 Fasering

Kijkend naar de planning van het project kun je de activiteiten van het project opdelen in verschillende categorieën:

- Requirements ophalen & formuleren
- Ontwerpen database & applicatie
- Bouwen applicatie
- Testen applicatie
- Documenteren & scriptie

In figuur 4.3 is de fasering voor het project te zien.



Figuur 4.3 - Fasering

4.4.4 Risico's

In ieder project zijn er bepaalde risico's. Dit project kent er ook een aantal. In de tabel hieronder zijn de risico's te zien, gevolgd door de gevolgen die ze hebben en de maatregel die ik er tegen neem.

Risico	Gevolg	Maatregel
Verliezen van laptop	Kwijtraken van documenten zoals scriptie en/of gebouwde applicatie.	Project wordt in git bijgehouden en documenten in de cloud.
Langdurig ziek	Achterlopen op planning & niet behalen van het gewenste resultaat	In het geval van langdurige ziekte zal ik contact moeten nemen met begeleidend docent voor een passende oplossing.
Uitbreiding API past niet bij het tot nu toe ontworpen en gebouwde applicatie. (Toegevoegd eind sprint 1)	De applicatie zal opnieuw ontworpen en gebouwd moeten worden zodat nieuwe API wel aan het systeem gekoppeld kan worden.	Voordat ik het systeem ontwerp bekijk ik eerst de twee meest complexe API's. Vervolgens ontwerp ik een applicatie die met die twee API's overweg kan. Simpelere API's moeten dan ook geen probleem zijn.

5. Sprint 0 | Oriëntatie

Nu de aanpak bekend, is neem ik u mee het project in. In de eerste twee weken heb ik mij gericht op het voorbereiden van het project. Hierbij heb ik onder andere requirements opgesteld, een plan van aanpak geschreven en heb ik e-learning courses gevolgd.

5.1 Laracasts

Op de eerste werkdag kreeg ik een lijst met e-learning filmpjes van mijn technisch begeleider. De filmpjes waren afkomstig van laracasts.com. Dit is een betaalde website waarbij verschillende onderdelen omtrent software ontwikkeling worden uitgelegd. De nadruk ligt op het Laravel framework.

Figuur 5.1 - Logo Laracasts.

Laracasts is opgericht door Jeffrey Way, hij heeft al jaren ervaring met het maken van tutorials. Jeffrey is niet betrokken met Laravel zelf, maar heeft wel een goede band met Taylor Ottwel (oprichter Laravel). Daarnaast is Laracasts ook op de homepage van Laravel zichtbaar, als go-to website om te leren.

De filmpjes op Laracasts zijn van verschillende niveaus. SOCIALR heeft ze samengesteld omdat ze het moeilijk vonden om mijn niveau in te schatten. Ik heb ongeveer de helft van de filmpjes bekeken, omdat de rest te basis was. Hieraan was ik ongeveer twee dagen kwijt, maar het heeft mij wel geholpen een beter beeld te krijgen over de tools waarmee ik aan de slag zou gaan.

5.2 Requirements

5.2.1 Aanpak

De requirements worden genoteerd volgens de user story notatie. De notatie die ik aanhoud is als volgt: “Als *type gebruiker* wil ik kunnen *doen”.* Een simpel voorbeeld hiervan is: “Als admin wil ik kunnen inloggen”. Deze notatie is niet erg technisch en zorgt ervoor dat voor beide partijen duidelijk is wat er onder een user story verstaan wordt.

Indien de user story niet duidelijk genoeg is, maak ik een use case beschrijving van de user story. Deze beschrijft de stappen binnen de user story in detail. Bij het eerder gegeven voorbeeld zou je in de use case beschrijving bijvoorbeeld kunnen beschrijven of de admin met een gebruikersnaam of email inlogt.

Ik zal iedere user story een tijd inschatting geven in uren. Daarnaast geeft de opdrachtgever iedere user story een prioriteit. Dit gebeurd volgens de MoSCoW notatie:

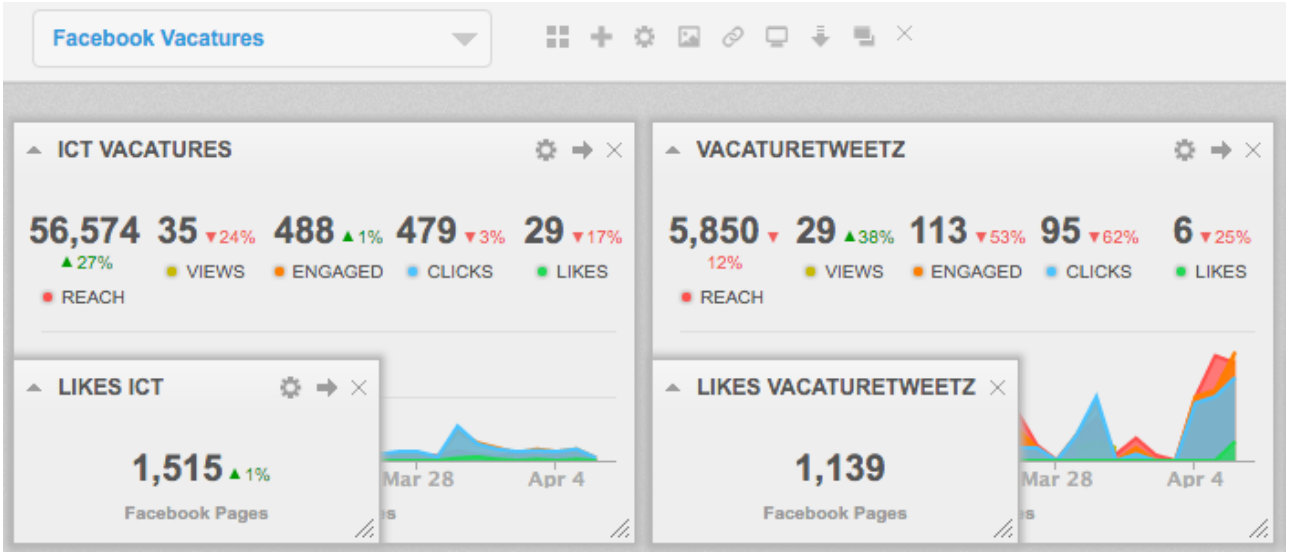
(M) Must	Moeten terugkomen in applicatie. Anders is product niet bruikbaar.
(S) Should	Zeer gewenst, maar zonder is het product wel bruikbaar.
(C) Could	Alleen wanneer er tijd genoeg is.
(W) Won’t	Komen niet in het project aanbod.

Er zijn ook onderdelen die ik niet als user story kan omschrijven, bijvoorbeeld: Een API doornemen of Laravel installeren. Deze onderdelen zijn voor mij persoonlijk, niet voor de eindgebruiker van de applicatie. Doordat ik deze onderdelen wel graag apart wil inplannen heb ik hiervoor een aparte categorie: ‘Items’. De items en user stories vormen samen de product backlog.

Er kunnen ook niet functionele requirements zijn, bijvoorbeeld: ‘Het systeem moet snel zijn’. Deze niet functionele requirements zal ik ook apart categoriseren onder de niet functionele requirements categorie.

5.2.2 Cyfe

Bij aanvang van het project heb ik toegang verzocht tot het Cyfe account van SOCIALR. Door de huidige situatie te bekijken was ik instaat om een beter beeld te krijgen van wat de opdrachtgever voor ogen heeft.



Figuur 5.2 - Screenshot Cyfe Dashboard. Twee widgets zichtbaar.

In figuur 5.2 zijn twee widgets te zien. Beide widgets laten ‘reach, views, engaged, clicks en likes’ zien. Deze data is afkomstig uit de Facebook Pages API. Één van de onderdelen die SOCIALR hierin mist is dat ze deze attributen niet zelf kunnen samenstellen. Cyfe heeft dit samengesteld en je hebt als gebruiker keuze uit een aantal samenstellingen. SOCIALR zou bijvoorbeeld alleen ‘reach en views’ willen laten zien, maar dat is niet mogelijk met Cyfe.

De 5 attributen die in figuur 5.2 in beide widgets te zien zijn, zijn allen afkomstig van de Facebook Pages API. SOCIALR zou graag de mogelijkheid willen hebben om attributen van verschillende API’s te combineren. Dit zou betekenen dat je bijvoorbeeld het reach attribuut van Facebook Pages ophaalt, maar de views van Google Analytics. (Als ze die aanbieden)

Bij Cyfe zit je als gebruiker altijd in een team, aan een team kunnen meerdere dashboards gekoppeld worden. Ieder dashboard bestaat uit widgets. Deze functionaliteit zou SOCIALR graag terug zien.

In Cyfe is het als gebruiker ook mogelijk om zelf dashboards en widgets toe te voegen. SOCIALR wil dit doen voor haar klant en wil niet dat de gebruiker dat zelf kan.

5.2.3 Requirements bij aanvang

Voorafgaand het project ben ik al een aantal keer langs SOCIALR geweest om de opdracht te bespreken. Hierin kwamen verschillende requirements naar voren die ik in het afstudeerplan verwerkt heb.

Tijdens de meetings kwam ik er al snel achter dat de requirements die in het afstudeerplan staan niet allemaal een hoge prioriteit zouden krijgen. Bepaalde requirements zoals het delen van statistieken op Facebook zou de opdrachtgever zelfs al een W (Won't have) willen geven.

Het werd mij steeds duidelijker dat het de opdrachtgever vooral gaat om de vrijheid om widgets helemaal zelf samen te stellen.

5.2.4 Product backlog

De verschillende meetings hebben uiteindelijk gezorgd voor een lijst met user stories en items. Samen vormen deze de product backlog. De user stories heb ik opgedeeld per type gebruiker om duidelijk te maken wat ieder type gebruiker met het systeem kan. Dit wilde ik in eerste instantie weergeven in een use case diagram maar door het opdelen per type gebruiker werd dat overbodig.

Waar nodig heb ik use case beschrijvingen toegevoegd om te verduidelijken wat een user story inhoudt, deze beschrijvingen zijn terug te vinden in de bijlage. Evenals de uren schatting die ik aan iedere user story en item gegeven heb. In het begin heb ik van alle user stories beschrijvingen gemaakt, hier wist ik nog niet of ik de opdrachtgever altijd goed begreep. Naar mate het project vorderde merkte ik dat er weinig onduidelijkheden waren tussen mij en de opdrachtgever. Hierdoor was de behoefte voor duidelijkere beschrijven niet meer nodig. In het geval dat dit wijzigde

Gebruiker (U)
Een gebruiker is een klant van SOCIALR die is ingelogd.

UR	Beschrijving	Prioriteit
(U) UR 1.0	Als gebruiker wil ik kunnen inloggen in het systeem.	M
(U) UR 2.0	Als gebruiker wil ik een password aanmaken voor mijn account.	M
(U) UR 3.0	Als gebruiker wil ik mijn dashboards kunnen bekijken.	M
(U) UR 4.0	Als gebruiker wil ik subgebruikers uitnodigingen voor in mijn team.	C
(U) UR 5.0	Als gebruiker wil ik uitnodigingen kunnen cancelen.	C
(U) UR 6.0	Als gebruiker wil ik subgebruikers uit mijn team kunnen verwijderen.	C
(U) UR 7.0	Als gebruiker wil ik statistieken van een widget kunnen delen op Facebook.	W
(U) UR 8.0	Als gebruiker wil ik de indeling van mijn dashboard kunnen wijzigen.	C
(U) UR 9.0	Als gebruiker wil ik kunnen chatten met admins.	W

Guest (G)
Een guest is iemand die niet is ingelogd.

UR	Beschrijving	Prioriteit
(G) UR 1.0	Als guest wil ik een cijfer voor mijn website krijgen bij het invoeren van mijn domeinnaam.	W
(G) UR 2.0	Als guest wil ik snel contact kunnen opnemen met SOCIALR na het bekijken van mijn website cijfer.	W

Admin (A)
Admin is een medewerker van SOCIALR die is ingelogd.

UR	Beschrijving	Prioriteit
(A) UR 1.0	Als admin wil ik kunnen inloggen in het systeem.	M
(A) UR 2.0	Als admin wil ik gebruikers kunnen uitnodigen voor het systeem.	M
(A) UR 3.0	Als admin wil ik meerdere dashboards kunnen aanmaken in het account van een gebruiker.	M
(A) UR 4.0	Als admin wil ik een dashboard kunnen samenstellen (widgets toevoegen).	M
(A) UR 4.1	Als admin wil ik een grafiek groep selecteren bij het maken van een nieuwe widget template.grafiek / diagram selecteren bij het maken van een nieuwe widget.	M
(A) UR 5.0	Als admin wil ik alle dashboards kunnen bekijken.	M
(A) UR 6.0	Als admin wil ik automatisch versturen van rapportages kunnen instellen.	C
(A) UR 7.0	Als admin wil ik een dashboard samenstellen en als template opslaan.	S
(A) UR 8.0	Als admin wil ik templates kunnen wijzigen.	S
(A) UR 9.0	Als admin wil ik templates kunnen verwijderen.	S
(A) UR 10.0	Als admin wil ik gebruikers rechten kunnen geven zodat ze de indeling van hun dashboard kunnen wijzigen.	S
(A) UR 11.0	Als admin wil ik het logo voor ieder account kunnen aanpassen.	C
(A) UR 12.0	Als admin wil ik de styling van de website kunnen wijzigen.	C
(A) UR 15.0	Als admin wil kunnen reageren op chatberichten van gebruikers.	W
(A) UR 16.0	Als admin wil ik queries kunnen schrijven om data op te halen.	W
(A) UR 17.1	ls admin wil ik facbook-pagina’s kunnen selecten als bron voor data	M
(A) UR 17.2	Als admin wil ik facebook advertenties kunnen selecten als bron voor data	M
(A) UR 17.3	Als admin wil ik google analytics kunnen selecten als bron voor data	M
(A) UR 17.4	Als admin wil ik adwords kunnen selecten als bron voor data	M
(A) UR 17.5	Als admin wil ik linkedin kunnen selecten als bron voor data	C
(A) UR 17.6	Als admin wil ik twitter search kunnen selecten als bron voor data	C
(A) UR 17.7	Als admin wil ik twitter kunnen selecten als bron voor data	M
(A) UR 17.8	Als admin wil ik youtube kunnen selecten als bron voor data	C
(A) UR 17.9	Als admin wil ik instagram kunnen selecten als bron voor data	M
(A) UR 17.10	Als admin wil ik google web masters / search console kunnen selecten als bron voor data	C
(A) UR 17.11	Als admin wil ik google analytics real-time kunnen selecten als bron voor data	C
(A) UR 18.0	Als admin wil ik een account + team kunnen aanmaken voor een gebruiker	M
(A) UR 19.0	Als admin wil ik een Widget Template met bijbehorende bronnen kunnen toevoegen	M

Gebruiker & Admin (UA)

UR	Beschrijving	Prioriteit
(UA) UR 2.0	Als user wil ik een PDF kunnen genereren.	C
(UA) UR 4.0	Als user wil ik de indeling van dashboards kunnen wijzigen.	S
(UA) UR 5.0	Als user wil ik een nieuw wachtwoord kunnen aanvragen.	M
(UA) UR 7.0	Als user wil ik mijn naam kunnen wijzigen.	M
(UA) UR 12.0	Als user wil ik kunnen instellen over welke periode ik statistieken wil inzien.	M

Items (U)

Item	Beschrijving	Prioriteit
1	Opzetten Laravel base application	M
2	Opzetten database application	M
3	Automatisch ophalen data (cron jobs)	M
4	Facebook pages data ophalen en opslaan	M
5	Facebook ads data ophalen en opslaan	M
6	Google Analytics data ophalen en opslaan	M
7	Google adwords data ophalen en opslaan	C
8	Google analytics real-time data ophalen en opslaan	C
9	Google web masters / search console data ophalen en opslaan	C
10	LinkedIn data ophalen en opslaan	C
11	Twitter data ophalen en opslaan	M
12	Twitter search data ophalen en opslaan	C
13	Youtube data ophalen en opslaan	C
14	Instagram data ophalen en opslaan	M
15	Git configureren voor het project	M

5. Sprint 1 | Basis applicatie

De applicatie die ik ga bouwen gaat voornamelijk om het ophalen en weergeven van data. De data moet natuurlijk wel te bekijken zijn binnen een applicatie. Deze applicatie dient over bepaalde basisfuncties te beschikken. Functies zoals: inloggen, gebruikers uitnodigen of dashboards samenstellen. Tijdens de eerste sprint zal ik mij voornamelijk richten op die basisfuncties.

5.1 Planning

De data die de applicatie uiteindelijk moet gaan weergeven is afkomstig van verschillende sociale platformen. Je kunt met zo een platform communiceren door de verschillende API's die zij aanbieden. Doordat ik over deze API's nog geen duidelijk beeld heb, wil ik alvast eens rond kijken. Mogelijk moet ik met zaken rekening houden tijdens het ontwerpen. Daarnaast zal het mij vast en zeker helpen bij het plannen van de toekomstige sprints.

5.1.1 Detailplanning

In overleg met de opdrachtgever en mijn technisch begeleider zijn de volgende user stories en taken ingedeeld voor sprint 1. Het '$' symbool in de rechter kolom staat voor het aantal uren dat ik voor het onderdeel heb ingeschat.

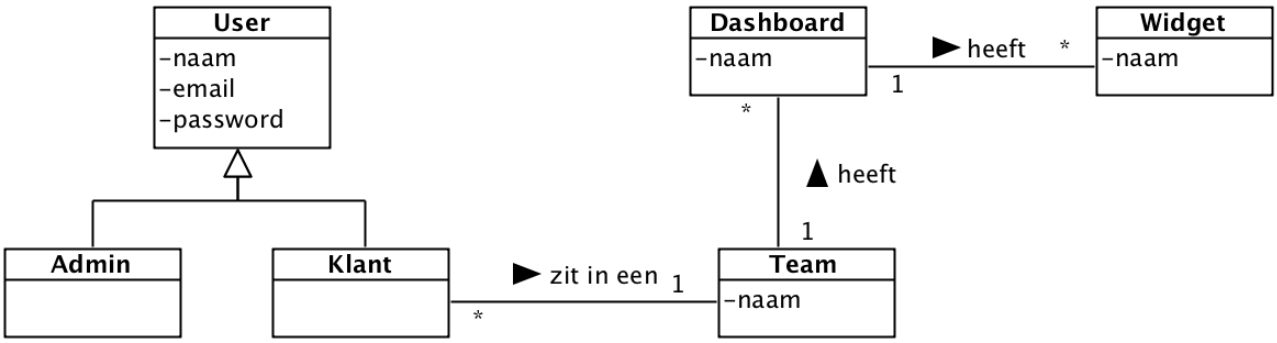
Taak	**$**
(U) UR 1.0 Als gebruiker wil ik kunnen inloggen in het systeem	3
(U) UR 2.0 Als gebruiker wil ik kunnen registreren voor het systeem, zodat ik in een team geplaatst wordt of er een team voor mij wordt aangemaakt.	4
Item 4 Facebook ads data ophalen en opslaan	20
(A) UR 3.0 Als admin wil ik meerdere dashboards kunnen aanmaken in het account van een gebruiker.	2
(A) UR 2.0 Als admin wil ik gebruikers kunnen uitnodigen voor het systeem.	4
(A) UR 5.0 Als admin wil ik alle dashboards kunnen bekijken.	4
(U) UR 3.0 Als gebruiker wil ik mijn dashboards kunnen bekijken.	2
(A) UR 1.0 Als admin wil ik kunnen inloggen in het systeem.	5
(UA) UR 5.0 Als user wil ik een nieuw wachtwoord kunnen aanvragen.	2
(UA) UR 12.0 Als user wil ik kunnen instellen over welke periode ik statistieken wil inzien.	3
Item 2 Opzetten database	1
(UA) UR 7.0 Als user wil ik mijn naam kunnen wijzigen.	3
Item 1 Opzetten Laravel application	1
(A) UR 4.0 Als admin wil ik een dashboard kunnen samenstellen (widgets toevoegen).	6
(A) UR 18.0 Als admin wil ik een account + team kunnen aanmaken voor een gebruiker	4
Sprint 1 review, requirements eind sprint 1, planning sprint 2	8
Testen	8
Totaal	80

5.2 Ontwerp

5.2.1 Concept

Kijkend naar de ingeplande onderdelen is het de bedoeling dat er tijdens deze sprint een applicatie gemaakt wordt waarin kan worden ingelogd met twee type gebruikers, admins en gebruikers.

Gebruikers kunnen zich registreren nadat zij een uitnodiging hebben gekregen van de admin. Vervolgens kunnen zij de dashboards bekijken die de admin eventueel voor hun team gemaakt heeft. Dashboards hebben widgets. Op basis van deze eisen heb ik een concept diagram gemaakt.



Figuur 5.1 - Concept diagram 1

Nadat de technisch begeleider dit concept diagram had goedgekeurd kon ik verder om het om te zetten naar een database.

5.2.2 Database

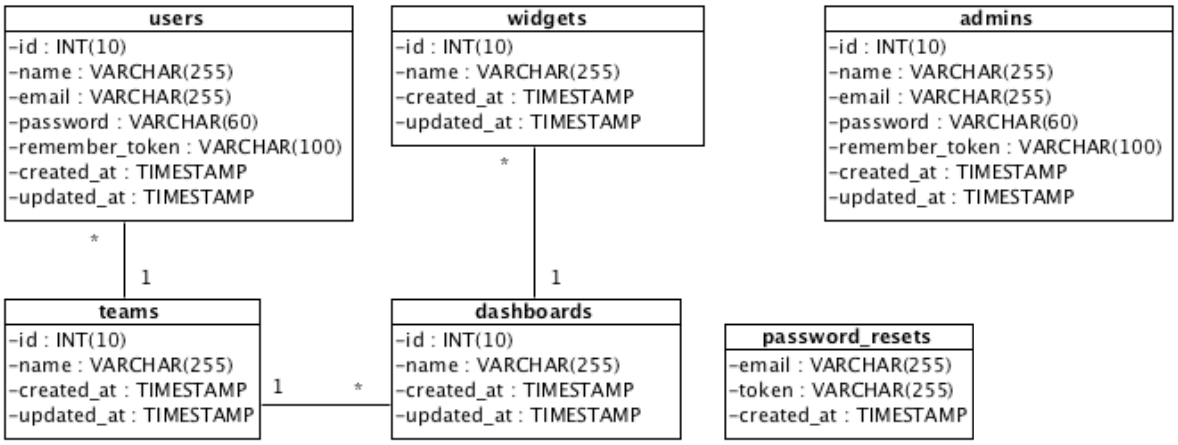
Om het concept diagram om te zetten naar de database heb ik het diagram allereerst vertaald naar een RRM, daarna naar een RIM. Deze heb ik beide in de bijlage toegevoegd. De RIM was niet nodig doordat Laravel gebruik maakt van migrations. Ondanks dat heb ik het RIM toegevoegd voor de lezer van dit verslag.

Laravel Migrations
Het komt weleens voor dat je als developer met een database verbind via een bepaalde database tool. Vervolgens maak of wijzig je via deze tool je database. Wanneer je versie beheer gebruikt staat daar niet de configuratie van je database in. Dit vormt een probleem wanneer je de applicatie uitwisselt met andere developers of wanneer je terug gaat naar een andere versie.

Laravel introduceert hiervoor migrations. Migrations zijn files binnen je project waarin je volgens een Laravel notatie de database formuleert. Vervolgens kun je met een command in de terminal de database bouwen, verwijderen of wijzigen. Hierdoor zit de juiste versie van de database altijd in je versie beheer en is het makkelijk om je project met andere uit te wisselen of terug te gaan naar andere versies.

Wanneer je de database bouwt, voegt Laravel automatisch een aantal attributen toe. Er wordt aan ieder model een created_at en updated_at attribuut toegevoegd. Daarnaast voegt Laravel een remember_me attribuut toe aan modellen waarmee je wil inloggen.

Er wordt ook een password_resets tabel toegevoegd voor de wachtwoord vergeten functionaliteit. In figuur 5.2 is de uiteindelijk database te zien voor sprint 1.



Figuur 5.2 - Database diagram sprint 1

5.2.3 Authenticatie modellen

In het database diagram is te zien dat de admin en users tabellen los van elkaar staan. Ik had er ook voor kunnen kiezen om dezelfde tabel te gebruiken. Ik had in dat geval een attribuut toe moeten voegen die aangeeft of iemand een admin is of niet. (figuur 5.3)

Ik heb dit overwogen maar doordat gebruikers in teams zitten en admins niet, zou dat betekenen dat er gebruikers zijn zonder relatie. Daarnaast zou ik in de applicatie altijd moeten controleren of de gebruiker een admin is of niet.

users	
id	INT(10)
name	VARCHAR(255)
email	VARCHAR(255)
password	VARCHAR(60)
remember_token	VARCHAR(100)
created_at	TIMESTAMP
updated_at	TIMESTAMP
isAdmin	BOOL

Figuur 5.3 - Voorbeeld users model bij het gebruik van 1 model.

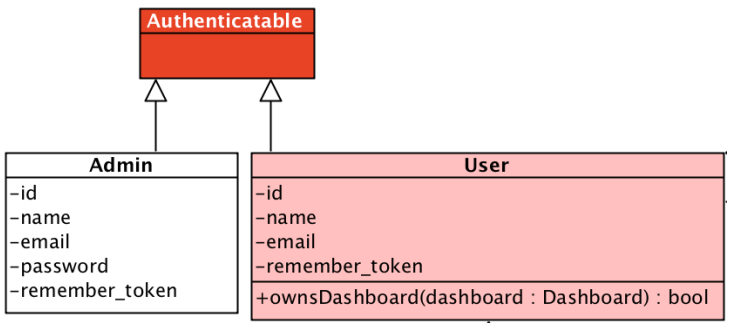
5.2.4 Applicatie

Voordat ik het ontwerp van de applicatie kan toelichten, wil ik eerst Traits behandelen.

Laravel Traits
Het komt wel eens voor dat je bepaalde functies wil delen tussen verschillende klassen. Dit kan gedaan worden middels overerving, maar dit is niet altijd geschikt. Soms hebben klassen namelijk niet de zelfde superklasse. Een oplossing voor deze situatie zijn traits.

Traits zijn klassen die je in iedere andere klassen kunt importeren om toegang te krijgen tot de functies in de trait. Je kunt de functies vervolgens ook overschrijven. Het zijn een soort helper klassen.

Authenticatie modellen
De opdrachtgever wil de mogelijkheid om met twee verschillende type gebruikers in te loggen. Laravel bevat zelf al een User model waarmee je kunt inloggen maar om inloggen mogelijk te maken met een ander model moet ik dezelfde super klasse overerven als user. In figuur 5.3 is te zien hoe ik dat wil doen.

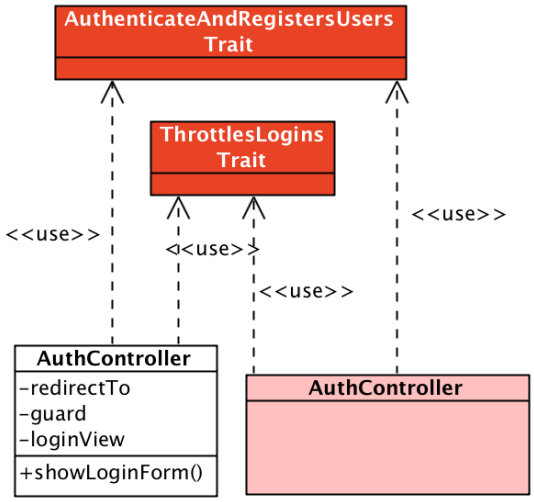


Figuur 5.3 - Stukje uit klassendiagram sprint 1, auth modellen.

Met alleen een nieuw model red ik het niet. Om het inloggen mogelijk te maken moet ik ook een extra controller maken. Ik kan hierin dezelfde traits importeren als de AuthController van user. Vervolgens kan ik overschrijven wat voor een admin verschilt. In figuur 5.4 is te zien hoe ik dit wil doen.

In figuur 5.3 en 5.4 is te zien dat ik klassen verschillende kleuren geef. De donker roze klassen zijn klassen die vanuit Laravel zelf komen, waarin het niet de bedoeling is dat code wordt aangepast of toegevoegd. De licht roze klassen komen ook vanuit Laravel zelf, maar hierin kan wel code worden toegepast en/of gewijzigd.

Ik geef niet alle klassen van het framework weer in het klassendiagram. Slechts de klassen die ik heb hergebruikt om mijn eigen functionaliteit te realiseren.



Figuur 5.4 - Stukje uit klassendiagram, AuthControllers.

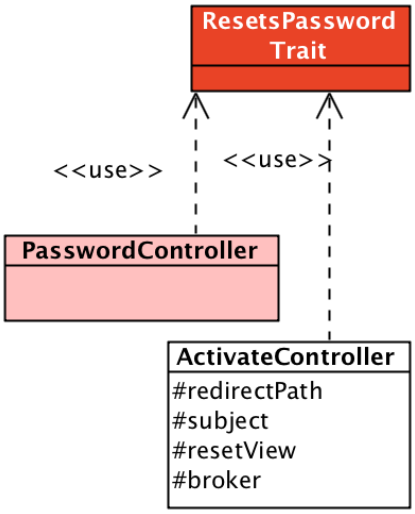
Gebruiker uitnodigen

Voor (U) UR 2.0 & (A) UR 18.0 is het de bedoeling dat de admin een gebruiker en team aanmaakt. Daarna moet er een mail verstuurd worden zodat de gebruiker een wachtwoord kan instellen voor het aangemaakte account.

De mail versturen zodat de gebruiker een wachtwoord kan instellen lijkt veel op de functionaliteit wanneer je jouw wachtwoord vergeten bent. Die zit al standaard in Laravel.

Mijn plan is om deze functionaliteit te hergebruiken. Ik ga een controller toevoegen, waarin ik dezelfde traits importeer als de PasswordController van Laravel. Hierin overschrijf ik vervolgens een aantal attributen zodat de mail weergeeft dat het om een registratie gaat, niet om een wachtwoord vergeten mail.

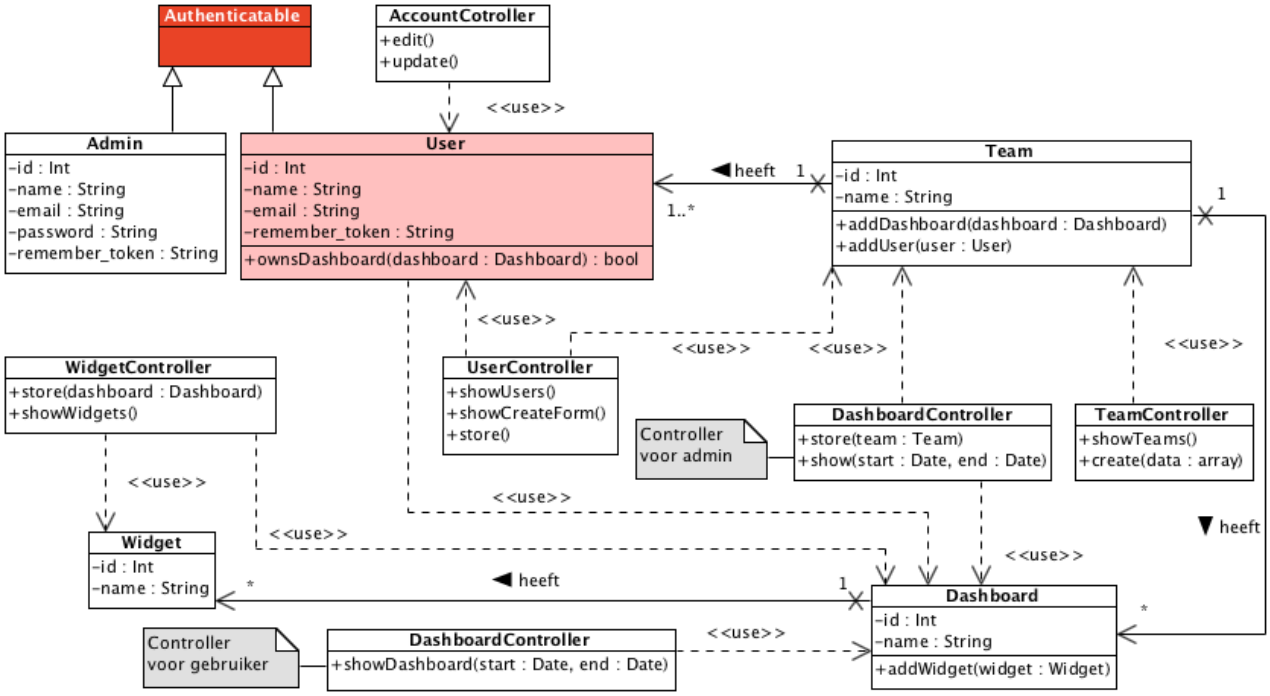
Hierdoor hergebruik ik op een slimme manier het framework voor mijn eigen functionaliteiten.



Figuur 5.5 - Stukje uit klassendiagram, Wachtwoord vergeten & ActivatieController.

Design diagram
Na de database heb ik de applicatie verder ontworpen. Het gehele design diagram voor deze sprint is te zien in figuur 5.6. De figuren 5.4 en 5.5 zijn hier buiten gelaten doordat ze losstaan.

5.3 Facebook Ads API



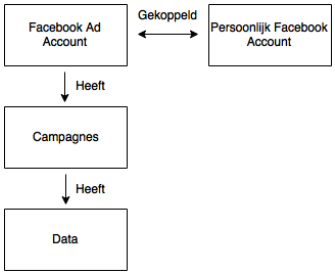
Figuur 5.6 - Design diagram sprint 1.

Tijdens deze sprint heb ik ook de Facebook Ads API bekeken. Ik heb besloten te beschrijven wat ik erover gevonden heb. Dit helpt mij overzicht te houden over de verschillende API's. Daarnaast geeft het de lezer van dit verslag een beter beeld over waarmee ik te maken heb gehad.

5.3.1 Accounts

In figuur 5.7 hier rechts is te zien dat data gekoppeld is aan een campagne. Deze campagne is gekoppeld aan een Facebook Ad account, bijvoorbeeld Ad Account van KFC.

Om data op te kunnen halen of om handelingen te verrichten moet je persoonlijke Facebook account gekoppeld worden aan een Facebook ad account. Dit kan gedaan worden door een admin.

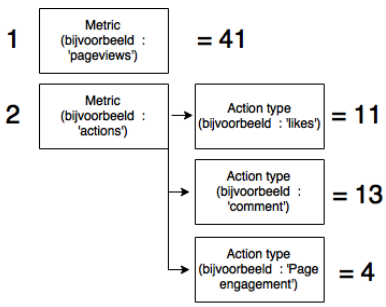


Figuur 5.7 - Facebook account structuur.

5.3.2 Data

De gegevens die je kunt ophalen vanuit de API noemen ze metrics. Wanneer je een bepaalde metric opvraagt krijg je een waarde terug. In figuur 5.8 punt 1 is dit te zien.

Er zijn ook metrics die geen waarde teruggeven, deze geven een collectie terug. Dit is te zien in figuur 5.8 punt 2. Om van deze metric een waarde terug te krijgen zul je naast de metric ook een action type moeten meegeven.



Figuur 5.8 - Facebook data structuur.

5.3.3 Ophalen data

Wanneer je vanuit een applicatie data wil ophalen uit de Facebook Ads API moet je verschillende gegevens meegeven:

- Ad Account ID (figuur 5.6)
- Campagne ID (figuur 5.6)
- Metric naam (figuur 5.6)
- **optioneel:** Action type naam (figuur 5.6)
- accesstoken

Het laatste gegeven: accesstoken is een token die je kunt krijgen als je middels OAuth inlogt bij Facebook. Dit zorgt ervoor dat alleen mensen met toegang, data kunnen ophalen.

Om data op te vragen stuur je de gegevens normaal gesproken mee aan een bepaalde URL, je krijgt dan de resultaten terug. Facebook bied ook een officiële PHP Facebook Ads SDK, deze kun je installeren in je applicatie. Dit zorgt ervoor dat je je verzoeken op PHP Objecten doet, niet via een URL. Ik wil deze gaan gebruiken, want het maakt het iets makkelijker.

5.3.4 Limieten

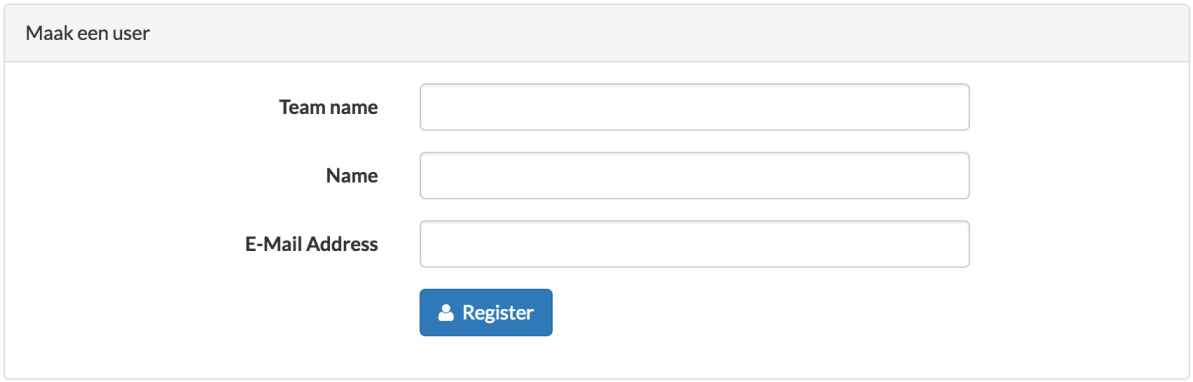
De Facebook Ads API accepteert een maximaal aantal verzoeken. Als deze bereikt worden, kun je voor een bepaalde periode geen verzoeken meer doen. Dit is niet wat ik wil, ik zal dus rekening met de limieten moeten houden bij het ontwerpen van de applicatie.

Het maximaal aantal verzoeken is 600, per 600 seconden, per IP, per token. IP is hier het IP-adres van de computer waar de verzoeken vandaan komen. Token de Facebook accesstoken. Gedurende het project heeft Facebook dit veranderd en geven ze geen concrete cijfers meer. Zij raden aan om een ander account te gebruiken wanneer de API aangeeft dat het limiet bereikt is.
RF01

5.4 Bouwen

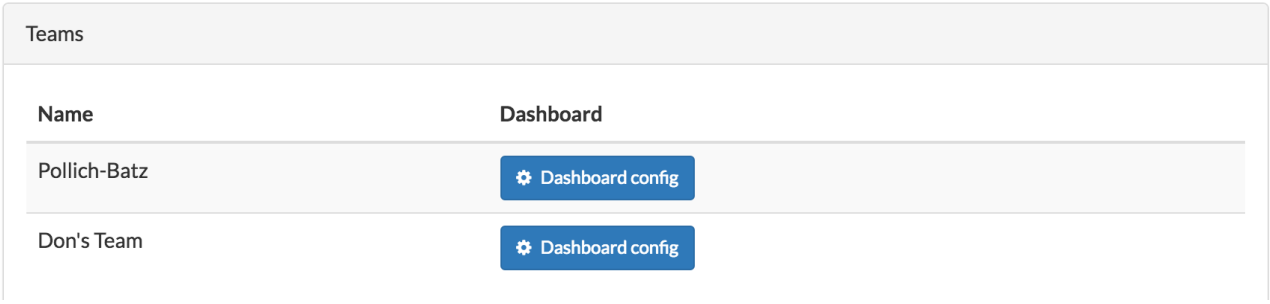
Na het systeem ontworpen te hebben ben ik gaan bouwen. Ik laat een aantal van de gebouwde onderdelen zien.

Gebruiker en team aanmaken
Een admin krijgt de mogelijkheid om nieuwe gebruikers en teams aan te maken. Ik heb een gebruikers overzicht toegevoegd met een knop: ‘Nieuwe gebruiker aanmaken’. Wanneer je op de knop klikt kom je op het scherm om een nieuwe gebruiker en team aan te maken (figuur 5.9)



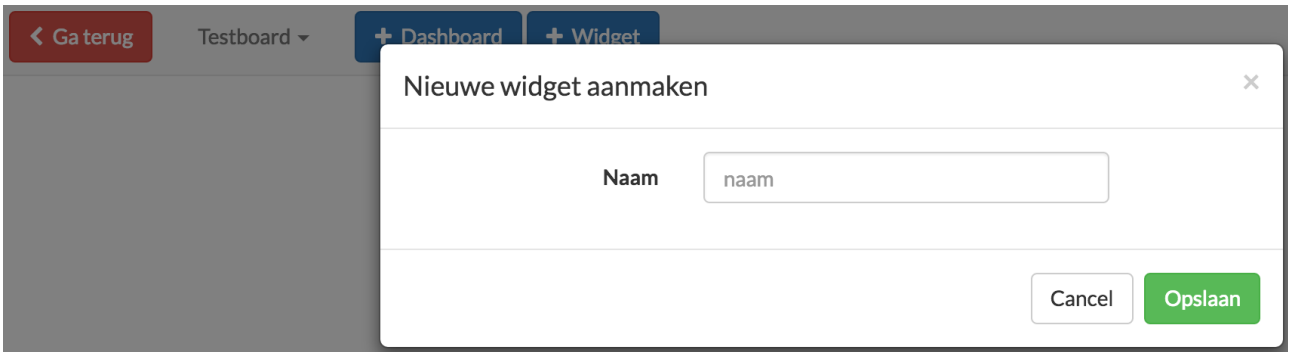
Figuur 5.9 - Screenshot applicatie, team en gebruiker aanmaken door admin.

Wanneer je een gebruiker en team hebt aangemaakt, heb je de mogelijkheid om de dashboards in te stellen. Hiervoor heb ik een teams overzicht toegevoegd. Hierop kun je navigeren naar een dashboard, door op ‘dashboard config’ te klikken. Dit is te zien in figuur 5.10.



Figuur 5.10 - Screenshot applicatie, overzicht teams.

Widgets toevoegen
Wanneer de admin op dashboard config heeft geklikt, kom je op de dashboard pagina. Hier krijgt de admin de mogelijkheid om widgets toe te voegen (figuur 5.11) of om nieuwe dashboards toe te voegen.



Figuur 5.11 - Screenshot applicatie, Nieuwe widget toevoegen.

Gebruiker uitnodigen
Wanneer de admin denkt dat het dashboard klaar is, kan hij de gebruiker een uitnodiging verzenden. Dit doe je door op het gebruikers overzicht op ‘stuur activatie mail’ te klikken. Hierdoor gaat mijn aangepaste wachtwoord vergeten functionaliteit in werking. (figuur 5.5)

5.5 Testen

Doordat ik testen niet als beroepstaak heb, heb ik besloten alleen te unit testen. Dit bespaard mij tijd, maar zorg ik dat de geschreven code toch getest is. Dit geldt voor iedere sprint. Alleen in deze eerste sprint test ik ook nog of emails correct verzonden worden, dit behandel ik in 5.5.3.

5.5.1 Uitgevoerde tests

Naam test	Beschrijving
user_can_view_his_dashboard	Er wordt getest of een user een dashboard kan bekijken die aan hem gelinkt is.
user_cannot_view_other_teams_dashboards	Er wordt getest of een ingelogde user niet het dashboard kan bekijken die niet aan hem gelinkt is, maar aan iemand anders.
change_user_name	Er wordt getest of een ingelogde user zijn user name kan wijzigen.
user_can_change_the_date_on_the_dashboard	Er wordt getest of een ingelogde user de weergave datum op een dashboard kan wijzigen.
user_cannot_visit_home_if_not_logged_in	Er wordt getest of de gebruiker naar de inlog pagina wordt doorverwezen als die home bezoekt en niet is ingelogd.
admin_can_add_a_new_user_that_is_attached_to_a_team	Er wordt getest of een admin naar de user overzicht pagina kan gaan, of hij daar op nieuwe user kan klikken en dan het nieuwe user formulier ziet. Er wordt getest of de user is aangemaakt, of er een team is aangemaakt en of de user is gekoppeld aan dat team, nadat er op opslaan is geklikt na het invullen van het formulier.
admin_can_create_dashboard_for_a_team	Er wordt getest of een admin een dashboard kan maken voor een team.
admin_can_view_dashboards	Er wordt getest of een dashboard een willekeurig dashboard kan bekijken.
user_can_register_with_a_valid_invitecode	Er wordt getest of iemand kan registreren als hij een geldige invitecode heeft
user_cannot_register_without_an_invitecode	Er wordt getest of een iemand niet kan registreren wanneer er geen invitecode is.
user_cannot_register_with_an_invalid_invitecode	Er wordt getest of een gebruiker niet kan inloggen met een ongeldige invitecode
as_a_user_i_want_to_register_and_be_placed_in_a_team	Er wordt getest of een persoon in een team geplaatst is wanneer die heeft geregistreerd.
as_a_user_i_want_to_be_able_to_save_a_new_password_after_a_activate_email	Er wordt getest of de gebruiker een password kan instellen voor zijn aangemaakt account. En of het correcte password wordt opgeslagen.

5.5.2 Factories

Laravel Factories
Factories zijn een manier om modellen aan te maken met random data. Deze random data is afkomstig uit de PHP Faker library. Door bijvoorbeeld: Factory('User')->create() wordt er een random user aangemaakt. In de Factory klasse formuleer je wat voor random data de verschillende attributen van het model moeten hebben, bijvoorbeeld naam, email of een alinea.

Factories zijn uitermate geschikt voor het testen. Het is namelijk zo dat je in veel gevallen functies wil testen waarbij een bestaand model nodig is. Factories maken het aanmaken van modellen zeer gemakkelijk, waardoor je je in je tests volledig kunt richten op testen. Ik heb factories gebruikt om de tests uit te voeren.

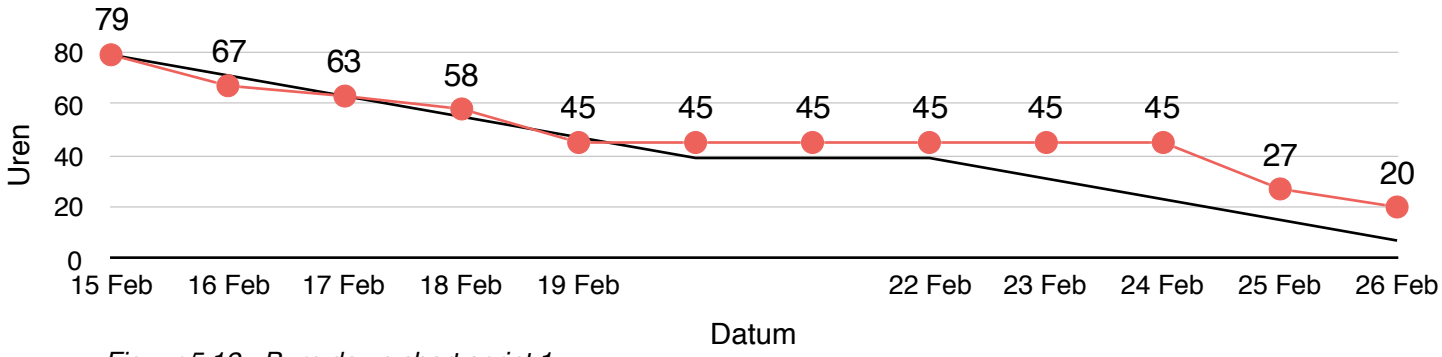
5.5.3 mailtrap.io

Het versturen van emails komt terug in verschillende user stories tijdens deze sprint. Je kunt natuurlijk controleren of de sendMail() functie wordt aangeroepen, maar hiermee test je niet of je mail aankomt en of de links in je mail functioneren.

Om dit toch te kunnen testen heb ik mailtrap.io gebruikt. Mailtrap.io is een gratis dienst waarbij je een account aanmaakt en dan toegang tot een inbox krijgt. De credentials die je van mailtrap.io ontvangt kun je in je Laravel applicatie instellen. Vervolgens worden alle mails vanuit je applicatie naar je mailtrap.io inbox verzonden. Een prima manier om te testen of mail aankomt en hoe de mail eruit ziet bij de gebruiker.

5.6 Reflectie

De eerste sprint is goed en fijn verlopen. De keuze om eerst de 'basis applicatie' te maken was erg verstandig. Hierdoor heb ik namelijk goed het framework leren kennen. De tutorials op laracasts.com hebben mij hierin uitstekend ondersteund. Echter is in de burn down chart hieronder te zien dat er ook zaken niet goed zijn gegaan.



Figuur 5.12 - Burn down chart sprint 1

In de burn down chart is te zien dat de eerste week volgens planning verliep. Deze week had ik mij gericht op Laravel specifieke delen zoals inloggen en dashboards bekijken. De problemen begonnen in de tweede week toen ik met de Facebook Ads API aan de haal ging. Het item waaraan ik ging werken was de volgende:

> *'Facebook ads data ophalen en opslaan`.'*

Het is niet zo zeer dat er iets niet ging goed maar het item is niet goed beschreven. Het had opgedeeld moeten worden in meerdere verschillende items, maar door mijn gebrek aan kennis over de API had ik dit nog niet eerder kunnen opdelen. Uiteindelijk heb ik zoveel mogelijk over de API beschreven zodat ik het in de tweede sprint kan gebruiken om de connectie te maken.

Het is jammer dat de Facebook Ads API deze afwijking in de burn down chart heeft veroorzaakt, maar achteraf is dit wel de reden dat ik dit item had ingepland. Hierdoor weet ik dat ik in toekomstige sprints het item moet opdelen en vooral ook in wat voor items ik het moet opdelen.

De opdrachtgever, technisch begeleider en ik hebben besloten dat we in de eerste sprint eerst de Google Analytics API zullen bekijken. Dit doen we om te voorkomen dat ik eerst iets ontwerp dat voor Facebook Ads werkt maar dan alles moet omgooien wanneer ik een andere API wil koppelen. De Analytics API is de juiste API om te bekijken, doordat de Analytics API en Facebook Ads API samen de twee meest complexe API's zijn. De gedachte is dat wanneer ik iets ontwerp dat voor deze twee API's werkt, de andere API's ook geen probleem zullen zijn. Ik heb als risico toegevoegd dat er de kans is dat ik de applicatie moet omgooien wanneer ik niet eerst goed een andere API bekijk.

6. Sprint 2 | Facebook Ads

6.1 Requirements

Facebook Ads
Ik begin de sprint met het verwerken van nieuwe requirements. Deze zijn bekend geworden tijdens de afgelopen sprint meeting. Het eerste onderdeel is het opsplitsen van item 5: ‘Facebook ads ophalen en opslaan’. De volgende items beschrijven beter wat ieder item inhoud en daardoor zijn ze beter te plannen.

Item	Beschrijving	Prioriteit
5.1	**Facebook ads** Uitzoeken hoe de API in elkaar zit	M
5.2	**Facebook ads** Verbinding maken met de API	M
5.3	**Facebook ads** Uitdenken beste manier data opslaan / cache	M
5.4.1	**Facebook ads** Data ophalen van API	M
5.4.2	**Facebook ads** Layer die controlleert of data beschikbaar is in eigen DB of vanuit API moet.	M
5.4.3	**Facebook ads** Nieuwe data opslaan in eigen DB	M
5.5	**Facebook ads** Bouwen frontend	M
5.6	**Facebook ads** Testen	M

Nieuwe requirements

UR	Beschrijving	Prioriteit
(A) UR 20.0	Als admin wil ik bij het toevoegen van facebook ads als data zien of de campagne nog actief is.	C
(A) UR 21.0	Als admin wil ik een widget template kunnen toevoegen waarin ik de widget data samenstel.	M
(A) UR 23.0	Als admin wil ik bij het maken van een nieuwe widget template kiezen of ik een legenda wil laten zien.	C
(A) UR 24.0	Als admin wil ik op een databron een filter toevoegen waarbij ik kan kiezen dat de value van een bepaalde data eindigt, begint, bevat een bepaalde value.	S
(A) UR 25.0	Als admin wil ik een facebook account aan mijn account kunnen koppelen	M
(A) UR 26.0	Als admin wil ik een facebook account kunnen configureren in een widget	M
(A) UR 27.0	Als admin wil ik een ad account kunnen kiezen bij het configureren van een widget	M
(A) UR 28.0	Als admin wil ik een campagne kunnen selecteren bij het configureren van een nieuwe widget.	M
(UA) UR 14.0	Als user wil ik datum kunnen aanpassen van iedere individuele widget	C

(A) UR 20, 25, 26, 27 & 28 zijn user stories die ook bij de opdeling van item 5 horen.

Het zal vaak voorkomen dat SOCIALR dezelfde widgets wil gebruiken voor verschillende klanten. Zij willen niet iedere keer opnieuw widgets moeten samenstellen en instellen. Daarom zijn er requirements voor widget templates bijgekomen, (A) UR 21 & 23.

Gewijzigde requirements

UR	Beschrijving	Prioriteit
(A) UR 10.0	Als admin wil ik gebruikers rechten kunnen geven zodat ze de indeling van hun dashboard kunnen wijzigen.	S —> W
(A) UR 12.0	Als admin wil ik de styling van de website kunnen wijzigen.	C —> W

De opdrachtgever heeft laten weten dat iedere gebruiker de indeling op een dashboard moet kunnen wijzigen. Er is dus geen bepaald recht meer voor nodig en daarom is (A) UR 10.0 een W geworden. Ik heb de beschrijving van de user story niet gewijzigd omdat het wijzigen van de indeling al vermeld staat in (UA) UR 4.0.

Ook (A) UR 12.0 is een W geworden omdat de opdrachtgever heeft laten weten dat het wijzigen van een logo genoeg is. Deze staat al in (A) UR 11.0.

Google Analytics

Na het bekijken van de Google Analytics API kon ik ook item 6 ‘Google analytics data ophalen en opslaan’ opdelen in verschillende items. Deze items komen overeen met de items die bij Facebook Ads zijn gekomen.

Item	Beschrijving	Prioriteit
6.1	**Google analytics** Uitzoeken hoe de API in elkaar zit	M
6.2	**Google analytics** Verbinding maken met de API	M
6.3	**Google analytics** Uitdenken beste manier data opslaan / cache	M
6.4.1	**Google Analytics** Data ophalen vanuit API	M
6.4.2	**Google Analytics** toevoegen aan layer data ophalen API / DB	M
6.4.3	**Google Analytics** opslaan in eigen DB na ophalen.	M
6.5	**Google analytics** Bouwen frontend	M
6.6	**Google analytics** Testen	M

6.2 Planning

Na het bekijken van de Google Analytics API heb ik de volgende planning gemaakt. Vervolgens heb ik hem besproken met de opdrachtgever en die is akkoord gegaan.

6.2.1 Detailplanning

Taak	Schatting
(A) UR 19.0 Als admin wil ik een Widget Template met bijbehorende bronnen kunnen toevoegen	8
5.3 Facebook ads Uitdenken beste manier data opslaan / cache	4
(A) UR 26.0 Als admin wil ik een facebook account kunnen configureren in een widget	1
(A) UR 25.0 Als admin wil ik een facebook account aan mijn account kunnen koppelen	5
5.6 Facebook ads Testen	16
(A) UR 28.0 Als admin wil ik een campagne kunnen selecteren bij het configureren van een nieuwe widget.	4
(A) UR 27.0 Als admin wil ik een ad account kunnen kiezen bij het configureren van een widget	3
(A) UR 21.0 Als admin wil ik een widget template kunnen toevoegen waarin ik de widget data samenstel.	5
6.1 Google analytics Uitzoeken hoe de API in elkaar zit	8
6.2 Google analytics Verbinding maken met de API	8
6.3 Google analytics Uitdenken beste manier data opslaan / cache	4
16 API's naast elkaar leggen en juiste datamodel uitdenken (FacebookAds & Google Analytics)	8
Sprint meeting	8
Totaal	82

In deze planning ontbreken nog een aantal onderdelen voor het koppelen van de Facebook Ads API. Dit betekent dat wanneer alles volgens planning verloopt, ik tijdens deze sprint nog niet de gehele Facebook Ads connectie af zal hebben. Dit gaat in principe niet mee met de scrum ideologie om iedere sprint een shippable product af te hebben.

De reden is dat ik al een aantal onderdelen van de Google Analytics API heb ingepland. Het is jammer van de scrum ideologie maar het is nu verstandiger eerst een beter beeld te hebben van de verschillende API's. Dit zorgt ervoor dat ik iets kan ontwerpen dat voor meerdere API's werkt en niet alleen voor de Facebook Ads API.

Dit zijn de eerste weken van het project. Hierdoor is het nog geen probleem om een beetje af te wijken van de scrum ideologie. Het is wel mijn streven om in toekomstige sprints steeds een 'shippable' product af te hebben.

6.3 Google Analytics API

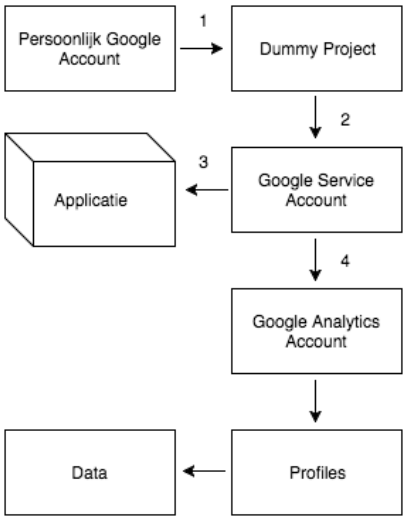
Net als de Facebook Ads API wil ik de Analytics API beschrijven. Het helpt mij overzicht te houden over de verschillende API's. Daarnaast geeft het de lezer van dit verslag een beter beeld over waarmee ik gewerkt heb.

6.3.1 Accounts

In figuur 6.1 geef ik weer hoe de Analytics API eruit ziet. Voordat je iets kunt doen met de API zul je met je persoonlijke Google account een dummy project aan moeten maken (1). Vervolgens kun je via dat dummy project een Google service account aanmaken (2). Hierbij krijg je gegevens die je in je applicatie moet instellen (3). Er is dus geen accesstoken nodig zoals bij Facebook Ads.

Wanneer je een service account hebt aangemaakt kan dat account gekoppeld worden aan een Google Analytics account (4). Een voorbeeld van een Analytics account is het account van Hema. Door gekoppeld te zijn krijg je toegang tot de gegevens van het Analytics account.

Ieder analytics account kan profiles hebben, waaraan data gekoppeld is.



Figuur 6.1 - Google Analytics structuur accounts.

6.3.2 Data

De structuur van data in Google Analytics komt overeen met die van Facebook Ads (figuur 5.8). Alleen de benaming is anders. Inplaats van action types heb je in analytics dimensions.

6.3.3 Ophalen Data

Om data op te halen van de Analytics API moet je een aantal gegevens meesturen.

- Analytics account ID
- Profile ID
- Metric
- **optioneel:** Dimension

Net als bij Facebook Ads kunnen de gegevens opgevraagd worden door een URL aan te roepen, echter biedt Analytics ook een officiële PHP SDK. Ik ben van plan deze te gebruiken.

6.3.4 Limieten

Net als de Facebook Ads API hanteert analytics ook limieten die je naar hun API kunt doen. Er mogen 10.000 verzoeken per dag, per profiel gedaan worden. Er kunnen tegelijk maximaal 10 verzoeken per profiel gedaan worden. [RF02]

6.4 Ontwerp

6.4.1 Bronnen

Het is de bedoeling dat iedere widget uit verschillende gegevens bestaat. Deze gegevens kunnen afkomstig zijn van verschillende API's. Een widget zal dus bestaan uit bronnen. In figuur 6.2 laat ik dit zien.

Figuur 6.2 - Dashboard, widgets en bronnen.

Door widget een relatie te geven met een bron (source) kunnen er meerdere toegevoegd worden per widget. Daarnaast maakt het niet uit van welke API die bron is, waardoor ik API's kan combineren.

6.4.2 Bronnen opslaan

Bij het bekijken van de Facebook Ads en Google Analytics API's ben ik erachter gekomen dat de structuur van de data overeenkomt, maar dat er wel andere namen aangehouden worden.

Facebook Ads Bron
-adAccountID -campagneID -metric -actiontype

Google Analytics Bron
-accountID -profileID -metric -dimension

Figuur 6.3 - Voorbeeld bron tabel voor iedere API.

Ik wil niet elke keer wanneer ik een dashboard open, opnieuw widgets instellen. Daarom wil ik opslaan welke data een source moet ophalen. Ook wil ik opslaan waar die data opgehaald kan worden. (Facebook of Google account)

In figuur 6.3 heb ik weergegeven hoe ik dat kan doen. Doordat er verschillende namen worden gebruikt, kan ik voor iedere API een aparte tabel maken. Dit is niet ideaal want het zou betekenen dat ik iedere keer de database moet aanpassen wanneer ik de applicatie uitbreid met een nieuwe API.

Bron
-account -profile -metric -metric_specific

Figuur 6.4 - Universele bron tabel.

Doordat beide API's vier gegevens nodig hebben, twee voor de data (metric, action type / dimension) en twee voor waar de data te vinden is (account, campagne / profile), kan ik een universele tabel maken. Een voorbeeld hiervan is te zien in figuur 6.4.

Het grootste probleem met deze universele tabel is dat wanneer ik de applicatie uitbreid met een API die meer dan vier gegevens nodig heeft. Dan moet ik alsnog de database aanpassen en dat is niet wat ik wil.

De oplossing die ik heb bedacht is gegevens op te slaan als JSON. Dit geeft mij de mogelijkheid om verschillende attributen op te slaan in één veld. Je kunt bijvoorbeeld in een text field opslaan: {'metric' : 'uitgave'}. Ik kan echter ook twee gegevens opslaan: {'metric' : '123', 'action_type' : 'likes'}.

Dit wil ik toepassen in twee velden, één voor de data (metric, action type / dimension) en één voor waar de data te vinden is (account, campagne /profile). Ik wil de velden gescheiden houden doordat ze ergens anders voor dienen.

6.4.3 Widget templates

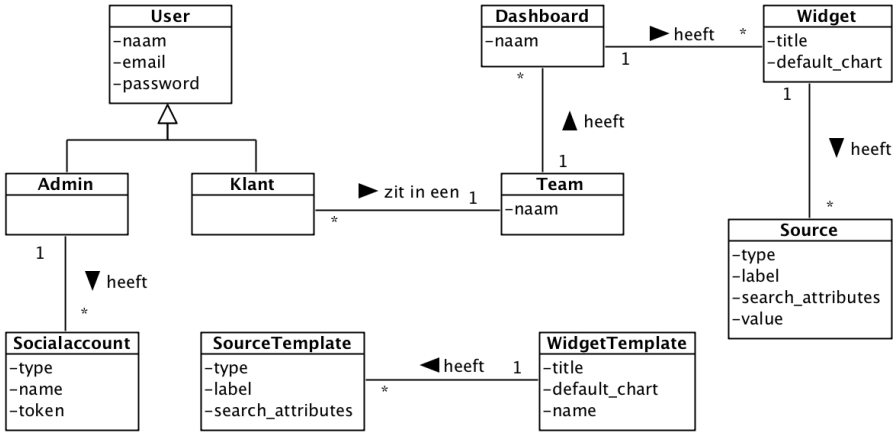
De opdrachtgever heeft aangegeven dat hij voor verschillende klanten vaak dezelfde widgets zal gebruiken. Nu betekent dat nog dat je iedere keer dezelfde widget moet samenstellen. Om dit te voorkomen wil hij graag templates kunnen samenstellen. Bij het toevoegen van een nieuwe widget moet er uit een template gekozen worden, waardoor de widget automatisch de instellingen van de template overneemt.

Om dit mogelijk te maken ga ik een nieuwe klasse introduceren, widget template. Deze template moet alleen bijhouden welke data er opgehaald moet worden. De gegevens van welk account die data opgehaald moet worden zijn per klant namelijk verschillend. Die stel je dus alleen in op een widget, niet op een widget template.

6.4.3 Concept diagram

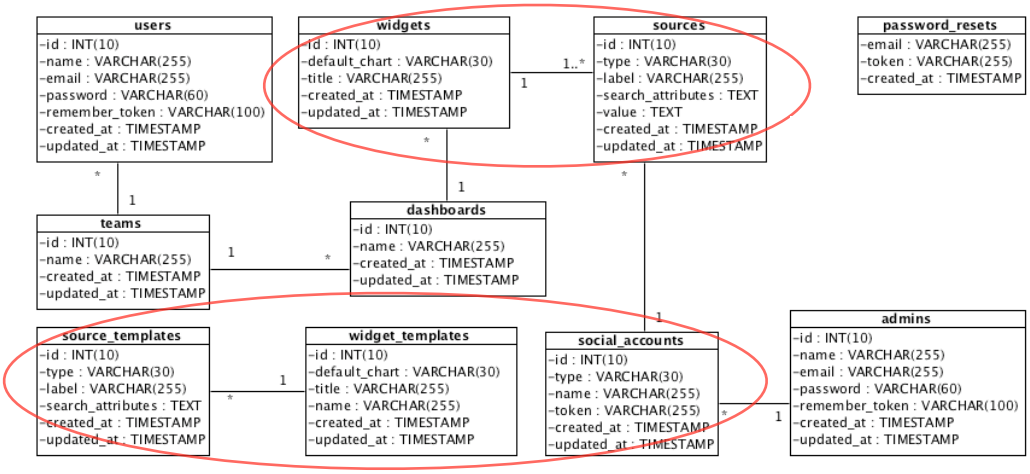
Naast de eerder besproken onderdelen komt er in deze sprint ook een nieuwe klasse bij om de accesstoken van Facebook te kunnen opslaan. In deze klasse kan ik de accesstoken van Facebook opslaan. Hierdoor moet ik niet iedere keer opnieuw inloggen.

De admin stelt de widgets in met bijvoorbeeld een social account. Het is dus de admin die zijn social accounts moet kunnen toevoegen. Hierdoor krijgt admin de relatie met social account. Deze nieuwe onderdelen hebben geleid tot een aangepast concept diagram. Die is hieronder in figuur 6.5 te zien.



Figuur 6.5 - Concept diagram sprint 2

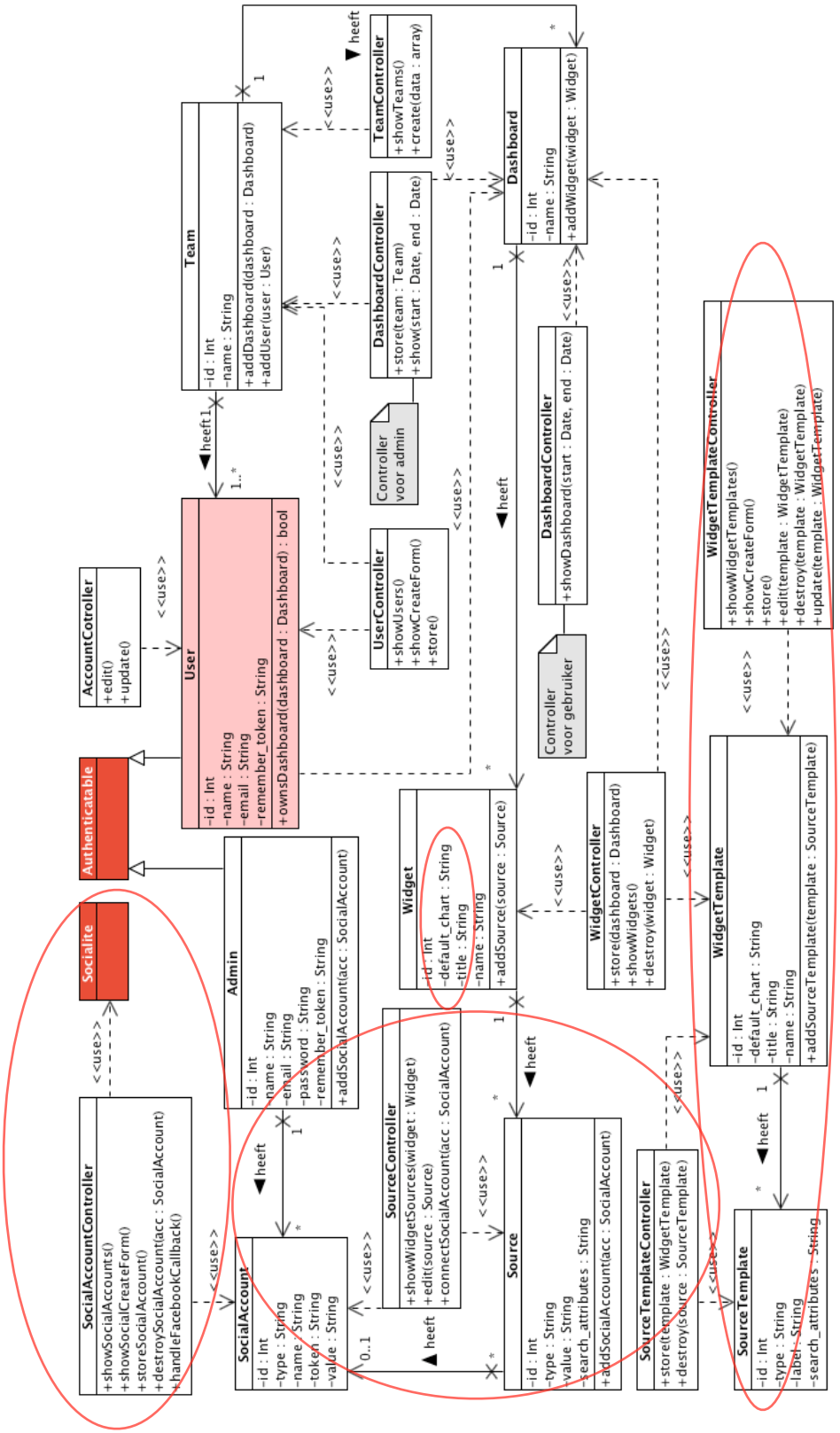
Om dit te vertalen naar een database heb ik RRM en vervolgens RIM uit de vorige sprint aangepast. Deze zijn terug te vinden in de bijlage. De uiteindelijke database voor sprint twee is te zien in figuur 6.6.



Figuur 6.6 - Database sprint 2.

6.5.2 Applicatie

In het design diagram (figuur 6.7) zijn een aantal nieuwe controllers en modellen te zien. Deze hebben te maken met de nieuwe tabellen die ik heb toegevoegd. Er zijn ook kleine wijzingen gemaakt. Ik heb omcirkelt wat vernieuwd is.



Figuur 6.7 - Design diagram sprint 2.

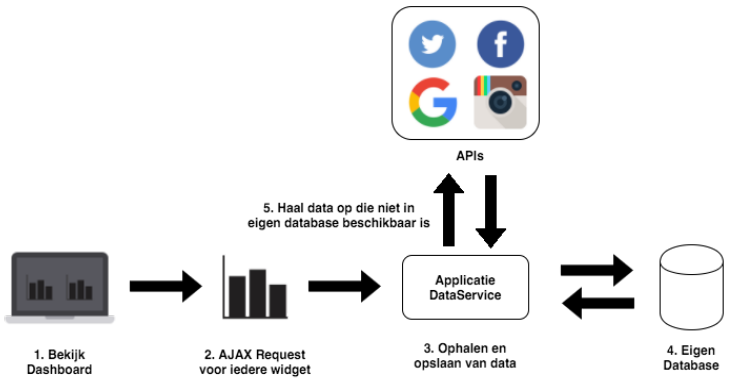
6.5.2 Uitdenken ophalen data

In de eerste sprint kwam ik erachter dat de Facebook Ads API erg uitgebreid is. Nu ik ook de Analytics API bekeken heb, weet ik dat het daar hetzelfde geval is.

Bij het maken van het afstudeerplan heeft het bedrijf al vermeld dat data opgeslagen zou moeten worden in verband met de limieten die de API's hanteren. Wanneer je namelijk verzoeken zou doe, iedere keer wanneer je het dashboard opent of herlaad, dan zou je misschien het limiet bereiken. Daarnaast is het ook niet handig om iedere keer wanneer je een dashboard laad, meerdere verzoeken naar verschillende API's te doen. Dit zou de applicatie natuurlijk behoorlijk langzaam kunnen maken.

De oplossing die het bedrijf voor ogen had, is om dagelijks de applicatie automatisch data te laten ophalen. Nu ik de API's bekeken heb, denk ik niet dat dit de slimste wijze is. Het is onnodig om iedere dag alle data, van iedere API, voor elke klant op te halen. Dit zou enorm veel tijd kosten en zou veel ruimte in beslag nemen. Dit is zonde als de klant maar een aantal gegevens zichtbaar heeft op het dashboard.

De oplossing die ik heb verzonnen heb is om data van de API op te halen wanneer het dashboard opgevraagd wordt. Zodra de data is opgehaald wordt het ook direct opgeslagen in de database. Als daarna dezelfde data wordt opgevraagd, dan kijkt de applicatie eerst in eigen database. Pas wanneer er data ontbreekt wordt het bij de API opgevraagd. In figuur 6.8 probeer ik dit te weergeven.



Figuur 6.8 Opgehaalde data opslaan voor hergebruik.

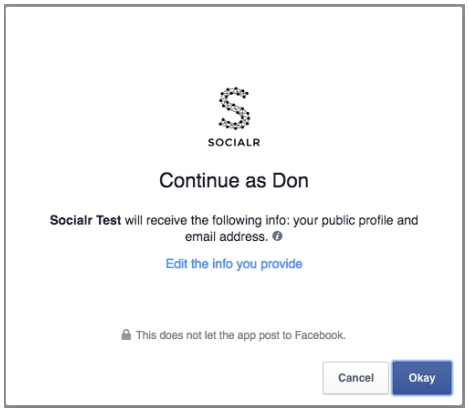
De technisch begeleider en opdrachtgever zijn akkoord gegaan met deze methode. Ik ga het deze sprint echter nog niet implementeren, doordat ik daar nog geen tijd voor heb. Dit komt in de volgende sprint.

6.6 Bouwen

6.6.1 Socialite

Om in te kunnen loggen bij Facebook heb ik gebruik gemaakt van Laravel Socialite. Socialite is een first party package die OAuth inlog mogelijk maakt voor een aantal verschillende sociale platformen.

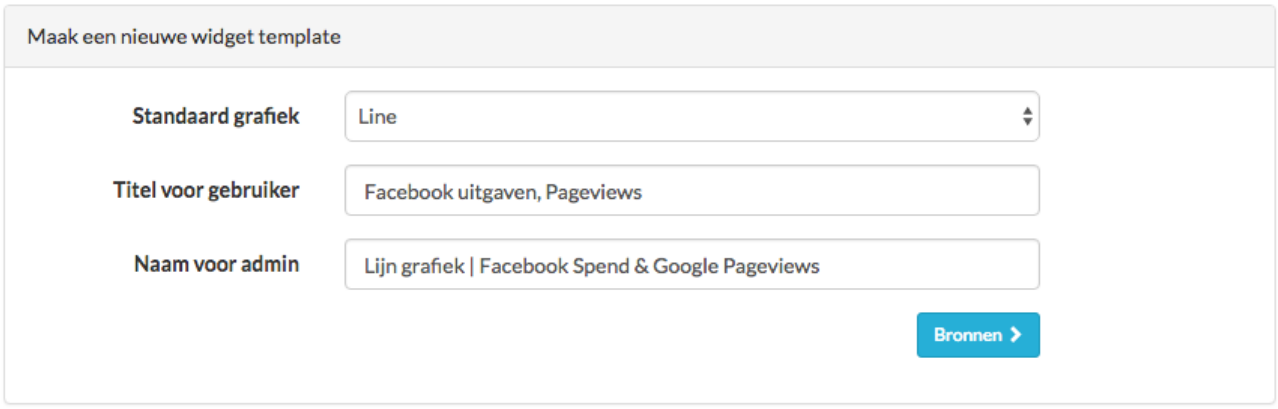
Ik heb voor de admin een scherm toegevoegd waarin hij zijn gekoppelde social accounts ziet. Nadat de admin op 'koppel nieuw account' klikt, kan hij Facebook selecteren.
Als je op volgende klikt verschijnt het scherm uit figuur 6.9.
Wanneer je de melding van Facebook accepteert, is je Facebook account gekoppeld en je token opgeslagen in de database.



Figuur 6.9 Facebook vraagt toestemming OAuth

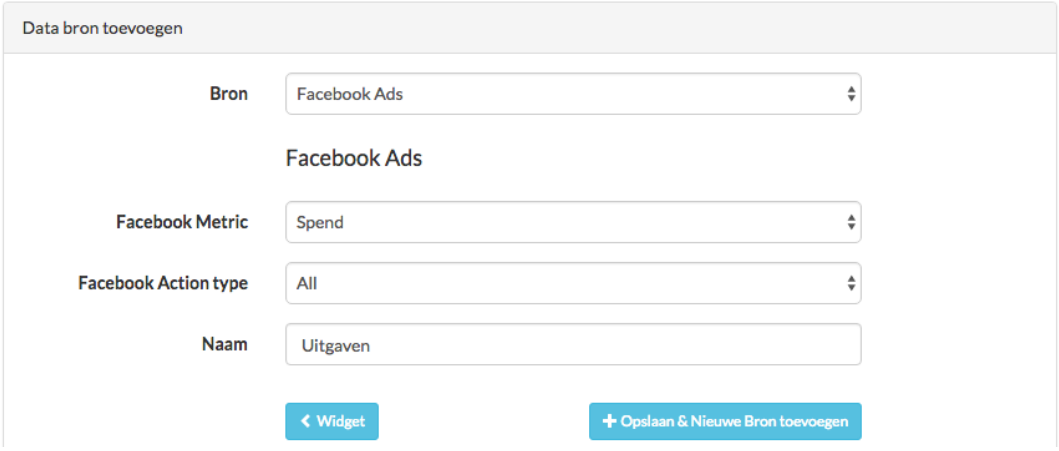
6.6.3 Widget templates

Ik heb widget templates geintroduceerd zodat de admin de instellingen van templates kan hergebruiken voor verschillende klanten. Bij het toevoegen van widgets kies je een template. Deze templates hebben een titel voor gebruiker en naam voor admin. Dit geeft de admin de mogelijkheid om uitgebreid te beschrijven wat de template inhoudt. (figuur 6.10)



Figuur 6.10 Screenshot applicatie, toevoegen van een nieuwe widget template.

Vervolgens klikt de admin op bronnen om sources toe te voegen. Hij kan er meerdere toevoegen van verschillende API's. Hierdoor kan een admin de widget (template) helemaal samenstellen hoe die zelf wil, iets dat niet mogelijk is in Cyfe. Dit is te zien in figuur 6.11.



Figuur 6.11 Screenshot applicatie, toevoegen van source template aan widget template

6.7 Testen

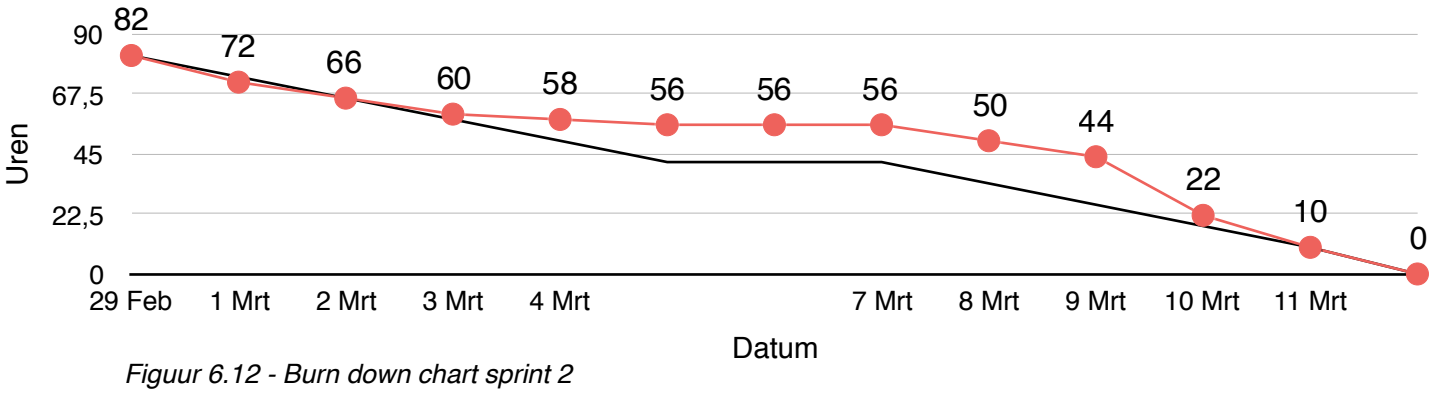
6.7.1 Uitgevoerde tests

#	Naam test	Beschrijving
1	admin_can_add_a_widget_template_and_add_sources_to_it	Er wordt getest of een admin een widget template kan aanmaken met bijbehorende source templates.
2	admin_can_remove_source_from_widget_template	Er wordt getest of een admin sources van een widget template kan verwijderen.
3	admin_can_edit_a_widget_template	Er wordt getest of een admin een widget template kan wijzigen.
4	admin_can_remove_a_widget_template	Er wordt getest of een admin een widget template kan verwijderen.
5	admin_can_remove_his_connected_facebook_account	Er wordt getest of een admin een geconnecte facebook account kan verwijderen uit zijn account.
6	there_will_be_a_widget_and_sources_made_for_widget_template_and_source_template	Er wordt getest of er bij aanmaken van een widget vanaf een widget template daadwerkelijk een widget gemaakt wordt met de zelfde instelling als de widget template die uitgekozen werd. Daarnaast wordt dit ook voor iedere van die widget template z'n source templates gecheckt of er een source is aangemaakt.

6.8 Reflectie

Deze sprint heb ik de eerste stappen gezet voor de koppeling van de Facebook Ads API. Daarnaast heb ik de Analytics API bekeken. Dit heeft me de mogelijkheid gegeven een manier te bedenken waardoor ik beter data kan ophalen en opslaan. (paragraaf 6.5.2)

Het is jammer dat ik nog steeds geen data kan weergeven aan het eind van de tweede sprint. Ik ben wel een stuk opgeschoten en ik heb de analytics API goed kunnen bekijken. De opdrachtgever vond het niet erg. We hadden immers samen besloten dat ik beter eerst de analytics API moest bekijken. Dit had wel als gevolg dat ik de koppeling met Facebook Ads niet heb kunnen afmaken.



Figuur 6.12 - Burn down chart sprint 2

In de burn down chart is te zien dat ik aan het eind van de eerste week een vertraging opliep. Het bouwen van widget templates duurde langer dan ik dacht. Ik moest met javascript ophalen welke velden er moesten worden weergeven en dat ging moeilijker dan ik dacht.

Uiteindelijk heb ik alle onderdelen van de sprint nog kunnen maken doordat het testen beter ging dan verwacht. Ik kon veel afkijken van vorige sprint en daarom ging het sneller dan gepland. Ik neem dit mee naar de volgende sprint, zodat ik daar eventueel minder tijd reserveer voor het testen.

7. Sprint 3 | Google Analytics

7.1 Requirements

Tijdens de afgelopen sprint meeting zijn er requirements bijgekomen. Allereerst zijn er twee requirements bijgekomen die voor zowel de gebruiker als admin gelden. (UA) UR 16.0 & 17.0.

UR	Beschrijving	Prioriteit
(UA) UR 16.0	Als user wil ik widgets kunnen resizen	C
(UA) UR 17.0	Als user wil ik de default chart van een widget kunnen wijzigen	S

De opdrachtgever heeft laten weten dat het makkelijk zou zijn als je de hoogte en breedte van een widget kunnen veranderen. Daarnaast zou het fijn zijn als je het type grafiek dat je hebt ingesteld kunt wijzigen op het dashboard. Beide zijn echter niet essentieel voor de werking van de applicatie.

Er zijn ook een aantal requirements voor de admin bijgekomen. (A) 37.0 t/m 49.0.

UR	Beschrijving	Prioriteit
(A) UR 37.0	Als admin wil ik bij het maken van een widget template iedere source een titel kunnen geven	S
(A) UR 38.0	Als admin wil ik de titel van een source op een widget kunnen wijzigen	S
(A) UR 39.0	Als admin wil ik een dashboard kunnen verwijderen	S
(A) UR 40.0	Als admin wil ik dashboard templates kunnen verwijderen	S
(A) UR 41.0	Als admin wil ik widget kunnen verwijderen van een dashboard	M
(A) UR 42.0	Als admin wil ik widget templates kunnen verwijderen	S
(A) UR 43.0	Als admin wil ik bij het toevoegen van een source op een widget template kunnen kiezen voor data filter.	S
(A) UR 46.0	Als admin kan ik de titel van een widget wijzigen	S
(A) UR 47.0	Als admin wil ik social accounts aan teams kunnen toevoegen zodat die per default gebruikt worden	S
(A) UR 48.0	Als kan ik een source uit een widget verwijderen	C
(A) UR 49.0	Als admin wil ik een google analytics account aan mijn account kunnen koppelen	M

Vorige sprint kwam er al het verzoek voor widget templates. Deze zorgen ervoor dat je een widgets kunt samenstellen en deze vervolgens voor verschillende klanten kunt hergebruiken. De opdrachtgever zou het fijn vinden als er ook templates komen voor gehele dashboards.

Wanneer de opdrachtgever een standaard dashboard heeft samengesteld met bijvoorbeeld 20 verschillende widgets, kan dit in 1x toegevoegd worden voor een nieuwe klant. Deze dashboard templates heeft de opdrachtgever overigens met een S geprioriteerd.

De items (A) 17.1, 17.2 en 17.4 zijn gewijzigd van drie naar acht uur. Deze onderdelen beschrijven het koppelen van accounts aan bronnen. Bij aanvang was de verwachting dat dit een simpel formulier zou zijn maar de accounts moeten ook nog opgehaald worden.

7.2 Planning

Vorige sprints heb ik het voorwerk gedaan voor de Facebook Ads en Google Analytics API. In deze sprint wil beide gaan afronden. Het is de bedoeling dat er aan het eind van deze sprint data te bekijken is van beide API's.

7.2.1 Detailplanning

Taak	Schatting
Item 6.4.1 Google Analytics Data ophalen vanuit API	6
Item 6.5 Google Analytics bouwen frontend	2
Item 6.6 Google Analytics Testen	8
Item 6.4.2 Google Analytics toevoegen aan layer data ophalen API / DB	8
Item 6.4.3 Google Analytics opslaan in eigen DB na ophalen.	8
(A) UR 17.3 Als admin wil ik google analytics kunnen selecteren als bron voor data	8
Item 5.4.2 Facebook Ads layer die controleert of data beschikbaar is in DB of moet ophalen uit API	12
Item 5.4.3 Facebook Ads opgehaalde data opslaan in DB	8
Sprint meeting, requirements en voorbereiden volgende sprint	12
(A) UR 49.0 Als admin wil ik Google Analytics account aan mijn account kunnen koppelen.	8
Totaal	72

Indien alles volgens planning verloopt, laat de applicatie aan het eind van deze sprint statistieken van zowel Facebook Ads als Google Analytics zien.

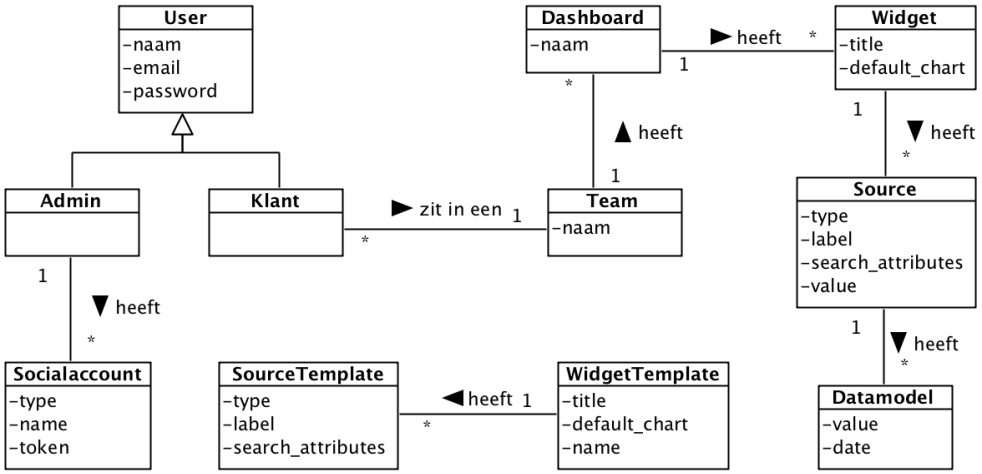
7.4 Ontwerpen

7.4.1 Datamodel

De vorige sprint heb ik uitgedacht dat het het beste is om opgehaalde data op te slaan. Dit zorgt ervoor dat wanneer ik de data weer nodig heb, ik niet opnieuw een verzoek moet sturen naar de API's. Alleen wanneer er data wordt opgevraagd die ik nog niet heb opgeslagen, vraag ik het op. (paragraaf 6.5.2)

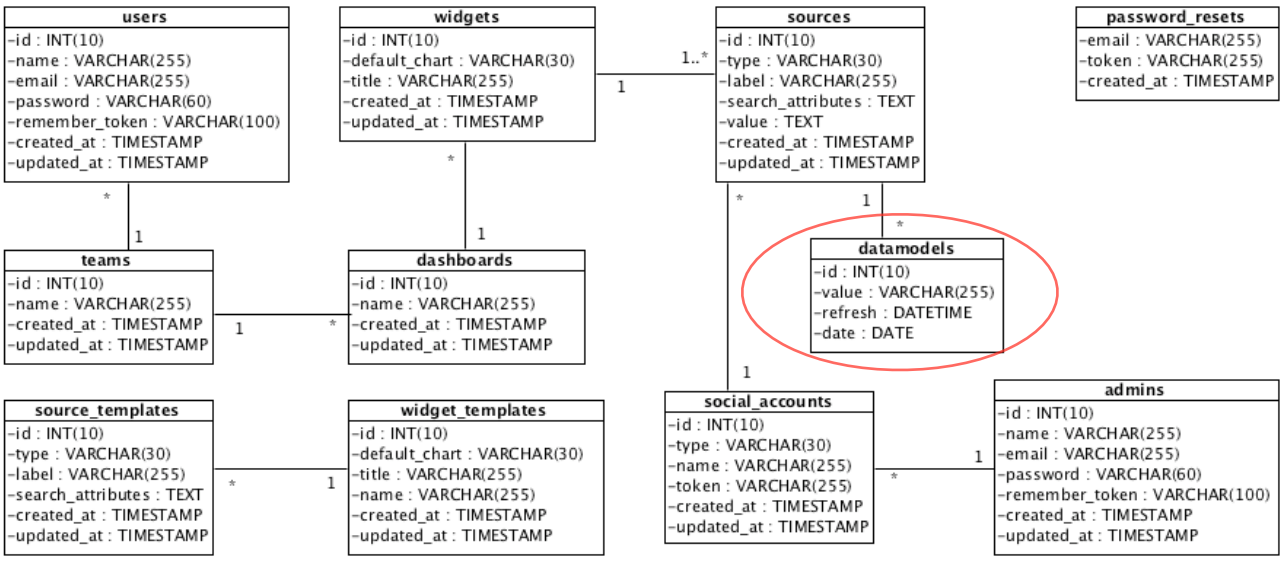
Om de gegevens te kunnen opslaan introduceer ik een nieuwe klasse: 'datamodel'. Widgets bestaan nu uit sources, hierin houd ik bij welke data er opgehaald moet worden (bijvoorbeeld: metric) en waar die data vandaan gehaald moet worden (bijvoorbeeld: Google account).

In het datamodel ga ik opslaan wat voor iedere dag de waarde van de source is. Wanneer je dus gegevens opvraagt voor de datums 1 t/m 10 februari. Dan worden er bij de desbetreffende source tien datamodels aangemaakt. Voor iedere dag één. In het concept diagram in figuur 7.1 is te zien hoe dit eruit ziet.



Figuur 7.1 - Concept diagram sprint 3.

Na deze wijziging toegepast te hebben in het RRM en RIM is er de volgende database aan overgehouden. Hierin is ook een attribuut 'refresh' te zien, deze bespreek ik in 7.5.1.



Figuur 7.2 - Datamodel sprint 3.

7.4.1 Applicatie

Dataservice

De logica om te bepalen of data al opgehaald is of niet, en het eventueel ophalen en opvragen van data wil ik een aparte klasse zetten. Deze klasse fungeert als een soort laag tussen de source en de SDK's die bij de API horen. Dit is de implementatie van de methode die ik in 6.5.2 heb uitgedacht.

Providers en Fetchers

Voor iedere API zijn er verschillende gegevens nodig. Wanneer er bijvoorbeeld een nieuwe widget wordt aangemaakt die Facebook Ads data gaat ophalen. Dan wil ik een formulier tonen waarin je een Facebook Metric kunt selecteren. Als ik analytics selecteer, dan moet er een andere pagina weergeven worden.

Deze verschillen per API wil ik opslaan in een aparte klasse, die noem ik een provider. In de providers staat dus welke pagina er weergeven moet worden, maar bijvoorbeeld ook de attributen die uit een formulier gehaald moeten worden. Er moet dus voor iedere API een provider komen.

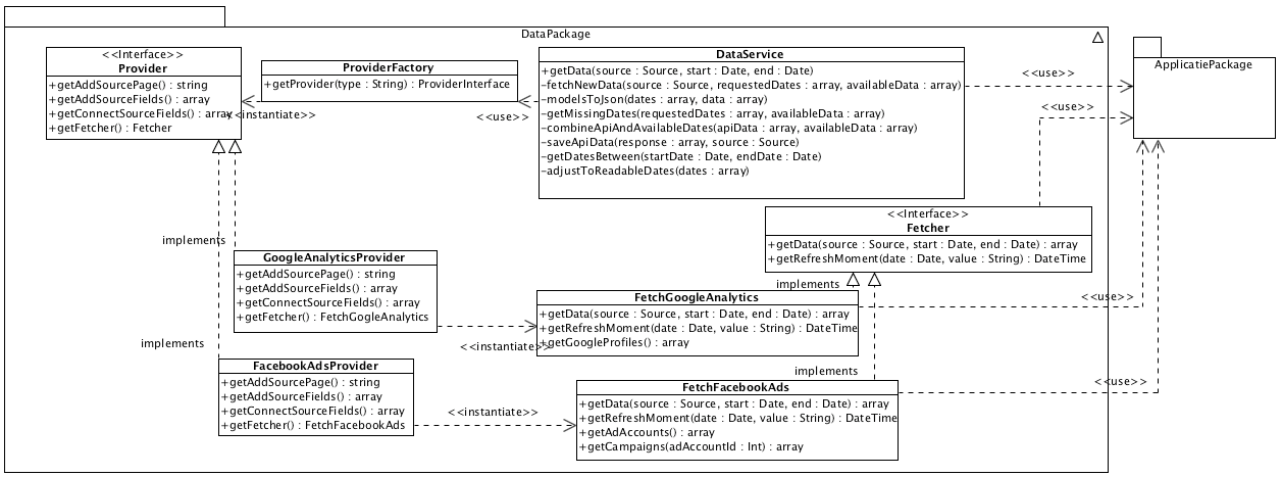
Het communiceren met SDK's en verwerken van data is ook per API verschillend. Ik wil dit echter niet in de providers opslaan, want ze doen iets heel anders. De klasse waarin ik dat wil opslaan noem ik fetchers. Voor iedere API komt er dus ook een fetcher klasse.

Factories

Door de verschillende API's kan de situatie ontstaan dat ik in code moet controleren om welk type source het gaat. Aan de hand daarvan kan ik bijvoorbeeld de juiste provider of fetcher selecteren. Dit heeft als gevolg dat ik bij uitbreiding op alle plekken waar ik het type controleer, code moet wijzigen.

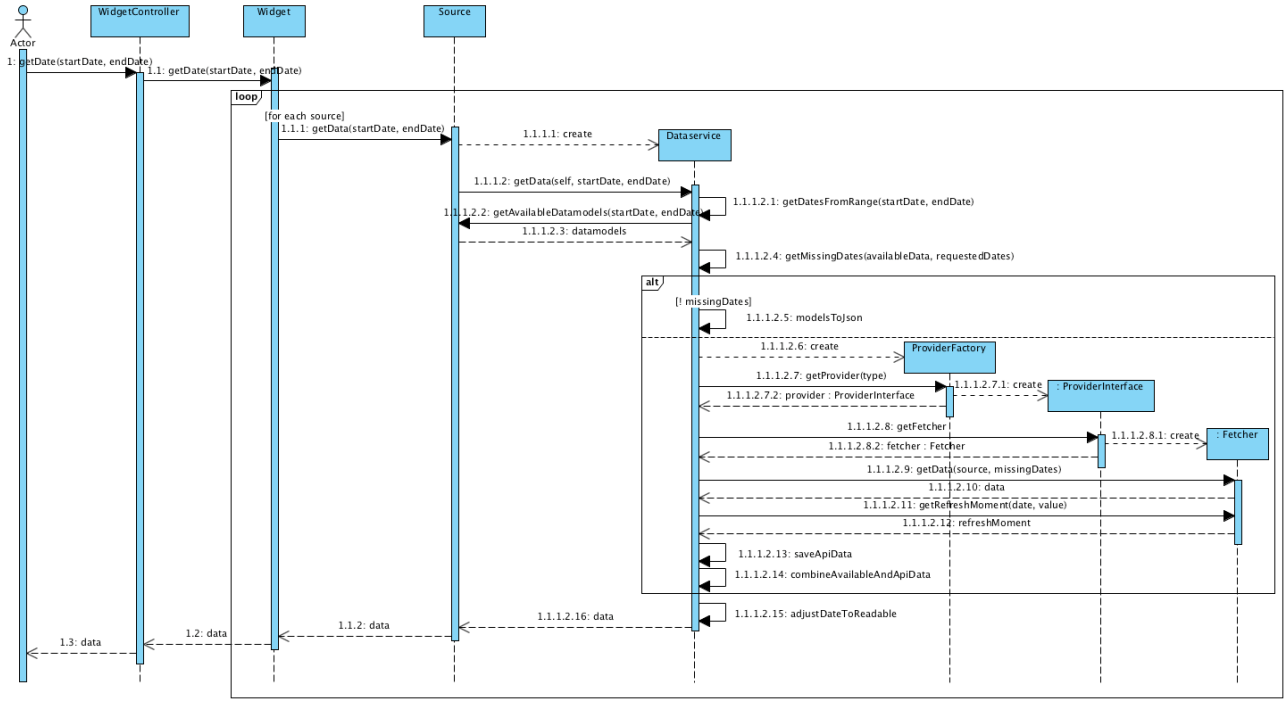
Om dit te voorkomen ga ik gebruik maken van een Factory. De bedoeling van de factory is om de enige plek te zijn waar op een type gecontroleerd wordt. Op plekken waar je dat anders zou doen, verwijs je naar de factory. Ze noemen dit ook wel het simple factory design pattern. Ik wil niet dat ik voor zowel de providers als fetchers een factory nodig heb, dan zou ik alsnog op meerdere plekken wijzigingen moeten doorvoeren. De oplossing die ik hiervoor heb verzonnen is om de provider een functie te geven die de juiste fetcher teruggeeft.

In figuur 7.3 zijn deze nieuwe klassen te zien. Dit is een design diagram in een package. Deze communiceert met de ApplicatiePackage, dat is wat ik in sprint 1 en 2 al heb ontworpen.



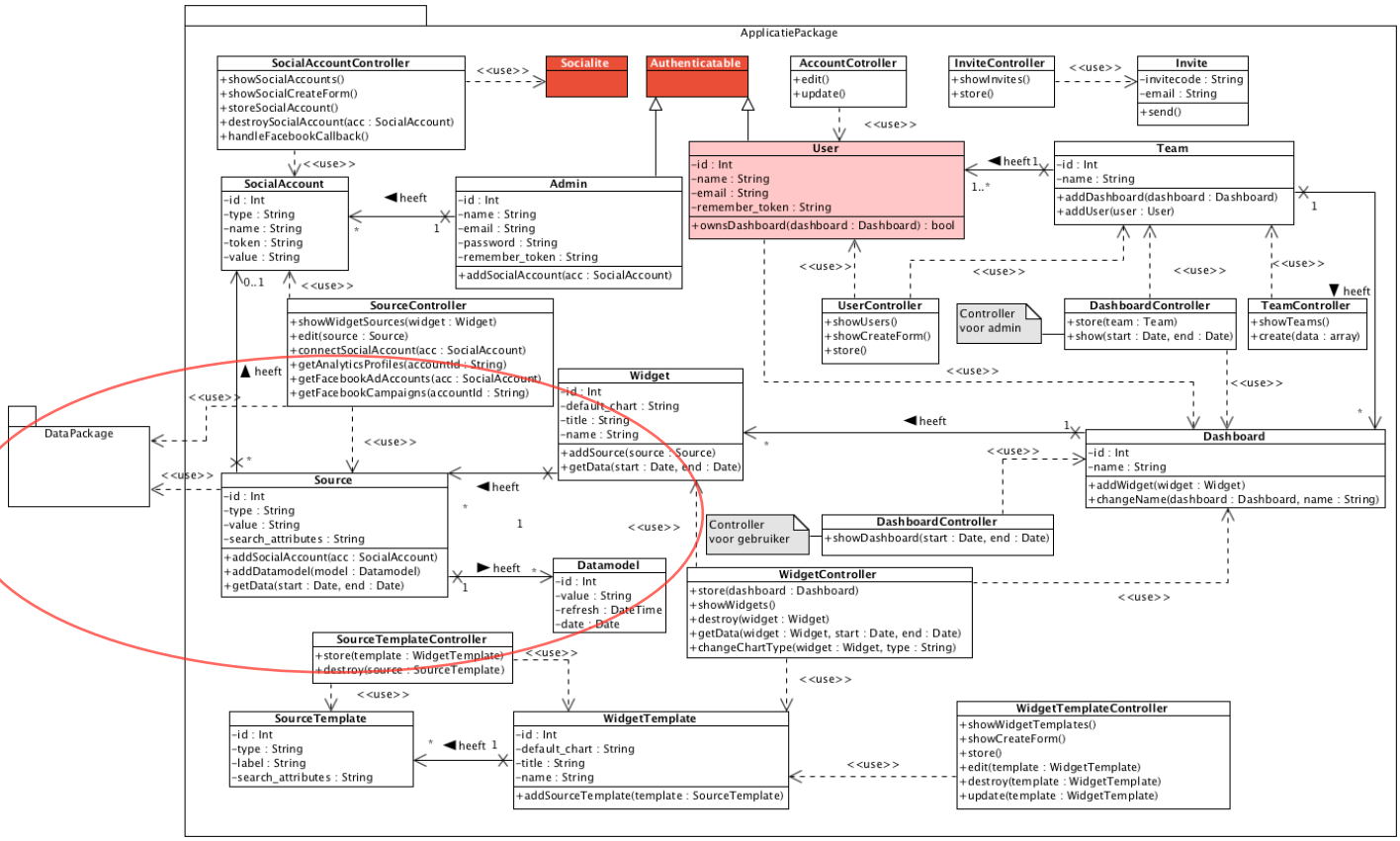
Figuur 7.3 - Design diagram, DataPackage sprint 3.

Zoals ik al zei is de dataservice de klasse die de bedachte methode uit 6.5.2 concreet maakt. Om de gehele flow te laten zien van wat er gebeurt wanneer er data opgevraagd wordt, heb ik het sequence diagram in figuur 7.4 gemaakt. Data wordt alleen opgehaald als het nog niet beschikbaar is in eigen database. Nadat het opgehaald is, wordt het opgeslagen voor hergebruik.



Figuur 7.4 - Sequencie diagram, ophalen data.

In figuur 7.5 is het aangepaste design diagram uit sprint 2 te zien. Deze heb ik nu de ApplicatiePackage genoemd. Om de onderdelen van deze sprint te maken heb ik nieuwe functies toegevoegd. Ik omcirkel wat nieuw is.



Figuur 7.5 - Design diagram, ApplicatiePackage sprint 3.

7.5 Bouwen

7.5.1 Refresh

Tijdens het bouwen van de connecties kwam ik erachter dat Facebook 0 terug geeft als ik data van dezelfde dag ophaal. De reden hiervoor is dat Facebook de volgende dag pas data beschikbaar stelt over de dag ervoor. Hierdoor ben ik ook gaan kijken hoe dat zit bij Google Analytics. Bij Google Analytics kun je wel data over dezelfde dag ophalen, maar die data kan in het verloop van de dag wijzigen.

Dit betekent dat wanneer ik data opvraag, maar ik die data daarvoor al eens had opgeslagen, de kans bestaat dat ik verouderde data uit mijn database ophaal. Hiervoor moest ik een oplossing bedenken.

De oplossing die ik heb bedacht is om een refresh attribuut toe te voegen. In het geval van Facebook betekent dit dat wanneer ik data van vandaag ophaal, ik een refresh moment meegeef. Dat refresh moment is bij Facebook de dag erna omdat dan pas de data beschikbaar is. Bij Analytics is het refresh moment een paar minuten later omdat de data constant bijgewerkt wordt.

Toen ik hier achter kwam heb ik wijzingen moeten doorvoeren in de ontwerpen. Ik heb een refresh attribuut toegevoegd aan de datamodel tabel in de database. Daarnaast heb ik de Fetcher Interface in de applicatie een methode ‘getRefreshMoment’ gegeven. Doordat de refresh momenten voor iedere API verschillen houd ik het bij per Fetcher klasse.

7.5.2 ChartJS

Om de grafieken frontend te bouwen heb ik gebruik gemaakt van ChartJS. Deze Javascript library heeft de technisch begeleider mij aangeraden te gebruiken. De verschillende grafieken die ik nodig heb, worden door ChartJS ondersteund. Daarnaast is het een gebruiksvriendelijke library.

Wanneer het dashboard geladen wordt, laat ik iedere widget een AJAX request doen om de data van de widget op te halen. AJAX requests zijn asynchroon. Dit zorgt ervoor dat mijn dashboard altijd snel geladen is. Als één van de widgets bijvoorbeeld nog data moet ophalen van een API, dan zullen de overige widgets direct hun data laten zien. Ze zullen niet moeten wachten tot die ene widget de data heeft opgehaald.

7.5.3 Beschikbare data query

De dataservice gaat aan de hand van de beschikbare data bepalen of er nieuwe data opgehaald moet worden vanuit de API. Hierbij moet ik rekening houden met het refresh attribuut. Als het refresh attribuut namelijk vroeger is dan nu, dan moeten de gegevens van die dag opnieuw opgehaald worden.

In figuur 7.6 is de query te zien die ik heb geschreven om dit mogelijk te maken.

```
public function getAvailableDatamodels($startDate, $endDate)
{
    $now = Carbon::now()->toDateTimeString();

    return Datamodel::where('date','>=', $startDate )
        ->where('date','<=', $endDate)
        ->where('source_id', $this->id)
        ->orWhere(function($query) use ($now) {
            $query->whereNotNull('refresh')
                ->where('refresh','<', $now)
                ->where('source_id', $this->id);
        })
        ->get();
}
```

Figuur 7.6 - Query ophalen beschikbare data in DB.

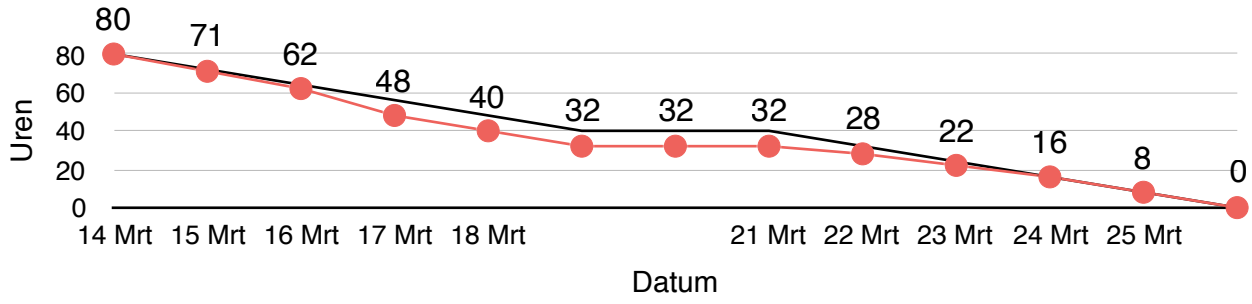
7.6 Testen

7.6.1 Uitgevoerde tests

#	Naam test	Beschrijving
1	a_admin_can_attach_googleanalytics_account	Er wordt getest of een admin een widget kan instellen met google analytics gegevens.
2	a_user_cannot_attach_googleanalytics_account	Er wordt getest of de user geen widgets kan configureren met google analytics gegevens.
3	test_dates_between_dates	Er wordt getest of de functie om datums tussen een begin en einddatum, de juiste datums teruggeeft.
4	test_readable_dates	Er wordt getest of de functie om datums naar mens vriendelijke tekst te vertalen, de juiste vertalingen maakt.
5	test_can_handle_facebook_data	Er wordt aan de hand van dummy data getest of de applicatie de facebook ads data op de juiste manier verwerkt.
6	will_not_retrieve_available_data	Er wordt getest of de applicatie geen verbinding maakt met de API's als alle opgevraagde data al beschikbaar is.
7	will_save_data	Er wordt getest of opgehaalde data juist wordt opgeslagen. Daarbij wordt getest of er geen dubbele dagen worden opgeslagen en dat het refresh attribuut op null gezet wordt als die niet meer nodig is.
8	test_can_handle_analytics_data	Er wordt aan de hand van dummy data getest of de applicatie de analytics data op juiste manier verwerkt.

7.7 Reflectie

Tijdens de afgelopen sprint zijn alle ingeplande onderdelen gelukt. Dit betekent dat ik een applicatie heb kunnen demonstreren die werkt met twee API's. De reactie hierop was erg positief, het is natuurlijk leuk om een werkend product te zien.



Figuur 7.7 - Burn down chart sprint 3

In de burn down chart is te zien dat de eerste week sneller liep dan gepland. Dit had ermee te maken dat het koppelen van Facebook accounts sneller ging dan gepland. Uiteindelijk heb ik niet meer in kunnen plannen, doordat in de tweede week het bouwen van de dataservice langer duurde dan verwacht. Er moest veel code geschreven worden om te zorgen dat alle data juist verwerkt wordt.

Doordat ik de connectie met beide API's heb kunnen afmaken deze sprint, kan ik vanaf nu proberen om mij iedere sprint met één API bezig te houden. Hierdoor zouden de onderdelen niet meer door elkaar moeten lopen.

8. Sprint 4 | Grafieken en Instagram Ads

8.1 Requirements

Tijdens de derde scrum meeting was er voor het eerst een applicatie die ook daadwerkelijk data in grafieken laat zien. Dit betekent dat het voor de opdrachtgever ook een stuk makkelijker was om te kunnen zien wat er nog aan functionaliteiten ontbreekt.

Grafieken
In figuur 8.1 is te zien hoe de grafieken er op dit moment uitzien. De opdrachtgever heeft laten weten dat hij de grafieken niet duidelijk vindt aangeven wat iedere lijn (source) betekent. Hij wil dat iedere lijn een willekeurige kleur krijgt. Daarnaast moet er een legenda boven de grafiek zichtbaar zijn. Hierin moet de kleur en titel van de lijn te zien zijn.

Figuur 8.1 Voorbeeld widget begin sprint 4.

Dit heeft geleid tot de volgende nieuwe items:

Item	Beschrijving	Prioriteit
22	Bouwen van legenda	M
23	Bouwen kleuren charts	M

Filters
Facebook Ads en Google Analytics bieden in hun eigen web omgevingen de mogelijkheid om filters toe te passen op de data. Je zou bijvoorbeeld pageviews kunnen ophalen, maar doormiddel een filter te gebruiken kun je het aantal pageviews vinden van mensen tussen de 18 en 25. Deze optie wil de opdrachtgever ook graag in de applicatie hebben.

Voor beide API's gelden er andere filters en ze worden anders toegepast. Ik zal dus voor beide apart moeten uitzoeken hoe ze werken om ze te kunnen implementeren. Ik heb de volgende items toegevoegd:

Item	Beschrijving	Prioriteit
18	Uitzoeken filters Analytics	M
19	Uitzoeken filters Facebook Ads	M
20	Ontwerpen filters	M
24	Bouwen filters	M

Facebook Campagnes
Bij de demonstratie van de applicatie in de laatste sprint meeting heeft de opdrachtgever laten weten dat hij graag de mogelijkheid heeft om campagnes op actieve widgets te wijzigen. Het komt namelijk wel eens voor dat een campagne eindigt. Het is dan niet meer interessant om gegevens van die campagne op te halen. Ik heb daarom de volgende user story toegevoegd:

UR	Beschrijving	Prioriteit
(A) UR 50.0	Als admin wil ik de campagne van een Facebook source wijzigen	S

Instagram
Voorheen werd er alleen over een Instagram API gesproken, maar de opdrachtgever heeft mij nu laten weten dat er onderscheid gemaakt wordt tussen twee API's. Instagram Ads API en Instagram Pages API. De Ads API is verantwoordelijk voor het ophalen van statistieken over advertenties in de Instagram app tijdlijn. De Instagram pages is daarentegen verantwoordelijk voor het ophalen van gegevens over Instagram accounts, zoals het aantal volgers.

Beide API's moeten in de applicatie komen, dus krijgen een Must have prioriteit. Item 14 die voorheen 'Instagram' beschreef heb ik veranderd naar Instagram Pages. Voor de Instagram Ads API heb ik een nieuw item toegevoegd:

Item	Beschrijving	Prioriteit
21	Instagram Ads connectie bouwen	M

Uit een kort vooronderzoek is gebleken dat de Instagram Ads API precies dezelfde is als de Facebook Ads API. Ik zal alleen een filter moeten toevoegen. De opdrachtgever wil wel dat bij het aanmaken van een nieuwe widget template, de Instagram Ads expliciet aangeklikt kan worden. Dit betekent dat er wel een aparte provider en fetcher voor moeten komen.

8.1 Planning

De opdrachtgever wil dat deze sprint de Instagram Ads API gekoppeld wordt. Doordat het veel overeenkomsten heeft met de Facebook Ads API, is er meer tijd over voor de andere onderdelen. Daarnaast wil de opdrachtgever dat ik de grafieken duidelijker maak.

De volgende items en user stories zijn ingepland. Er is een dag minder ingepland in verband met tweede Paasdag.

Taak	**Schatting**
(A) UR 7.0 Als admin wil ik een dashboard samenstellen en als template opslaan.	6
(A) UR 9.0 Als admin wil ik dashboard templates kunnen verwijderen.	2
Item 18 Uitzoeken filters Facebook Ads	3
Item 19 Uitzoeken filters Google Analytics	3
Item 20 Ontwerpen filters	6
Item 21 Instagram connectie bouwen	8
(A) UR 38.0 Als admin wil ik de titel van een source op een widget kunnen wijzigen	2
(A) UR 46.0 Als admin kan ik de titel van een widget wijzigen	3
Item 22 Bouwen legenda	6
Item 23 Bouwen kleuren charts	3
(A) UR 50.0 Als admin wil ik de campagne van een Facebook source wijzigen	8
Item 24 Bouwen filters	2
Testen	8
Voorbereiding sprint & sprint meeting	12
Totaal	72

8.2 Instagram API

Tijdens het uitzoeken van de Instagram API kwam ik erachter dat deze inderdaad overeenkomt met de Facebook Ads API. Ze zijn zelfs hetzelfde. Sterker nog, data die ik tot nu toe heb opgehaald van Facebook Ads was mogelijk ook van Instagram.

Instagram is een aantal jaar geleden overgenomen door Facebook. Wanneer je een advertentie plaatst wordt die advertentie automatisch op Facebook of op Instagram geplaatst, afhankelijk van waar volgens Facebook de beste plek is.

Bij het ophalen van de data is er een filter die je kunt meegeven om alleen data van Facebook of van Instagram op te halen. Dat komt mij goed uit want deze sprint ga ik ook de filters bouwen.

De opdrachtgever heeft wel aangegeven dat Instagram Ads als aparte API geselecteerd moet kunnen worden bij het aanmaken van een nieuwe widget template. Dit betekent dat ik zoals gewoonlijk een API toevoeg (provider en fetcher) maar dan een hard coded filter toepas.

8.3 Filters uitzoeken

Facebook & Instagram Ads
Facebook biedt de mogelijkheid om filters toe te voegen op een beperkt aantal velden. Per veld worden verschillende operators geaccepteerd. Ik zal er dus voor moeten zorgen dat de applicatie alleen maar valide filters accepteert die admin instelt.

Ik zal een filter mee moeten geven ‘placement = instagramstream’ om alleen gegevens van Instagram te ontvangen. Voor de rest is de Instagram Ads API hetzelfde als de Facebook Ads API. Ik kan zelfs dezelfde SDK gebruiken.

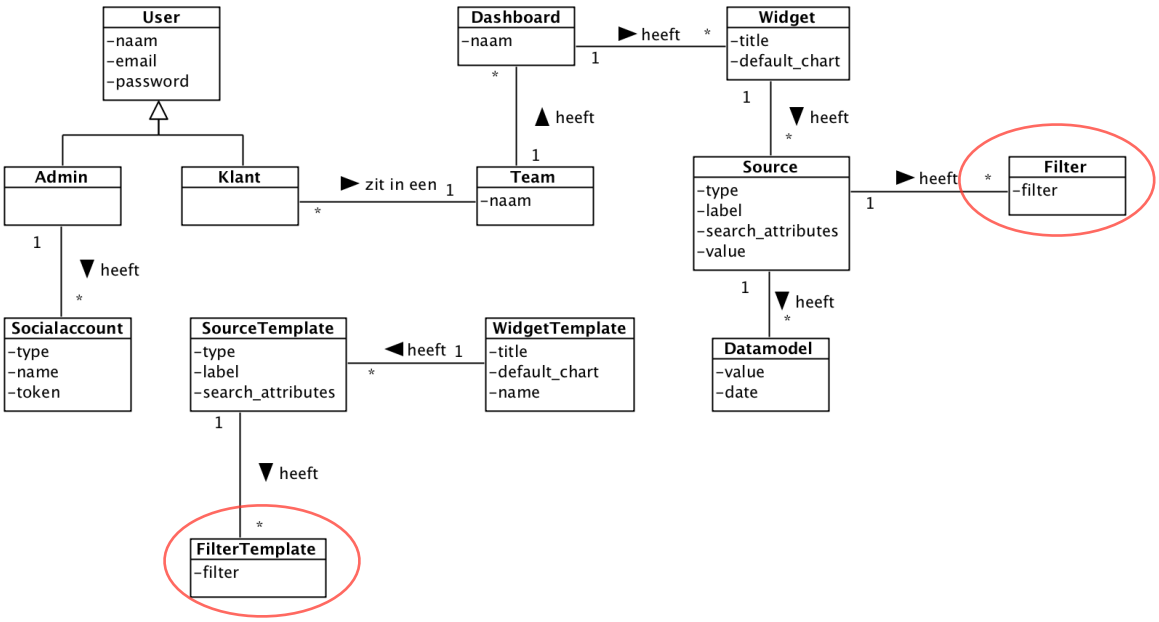
Google Analytics
Ook bij Google Analytics kunnen filters worden toegepast op verschillende metrics en dimensions. Hierbij gelden ook verschillende operators en die verschillen ook nog eens van die uit de Facebook Ads API.

Je kunt in de Analytics API filters combineren door de ‘and’ en ‘or’ operator.

8.4 Ontwerpen

8.4.1 Datamodel

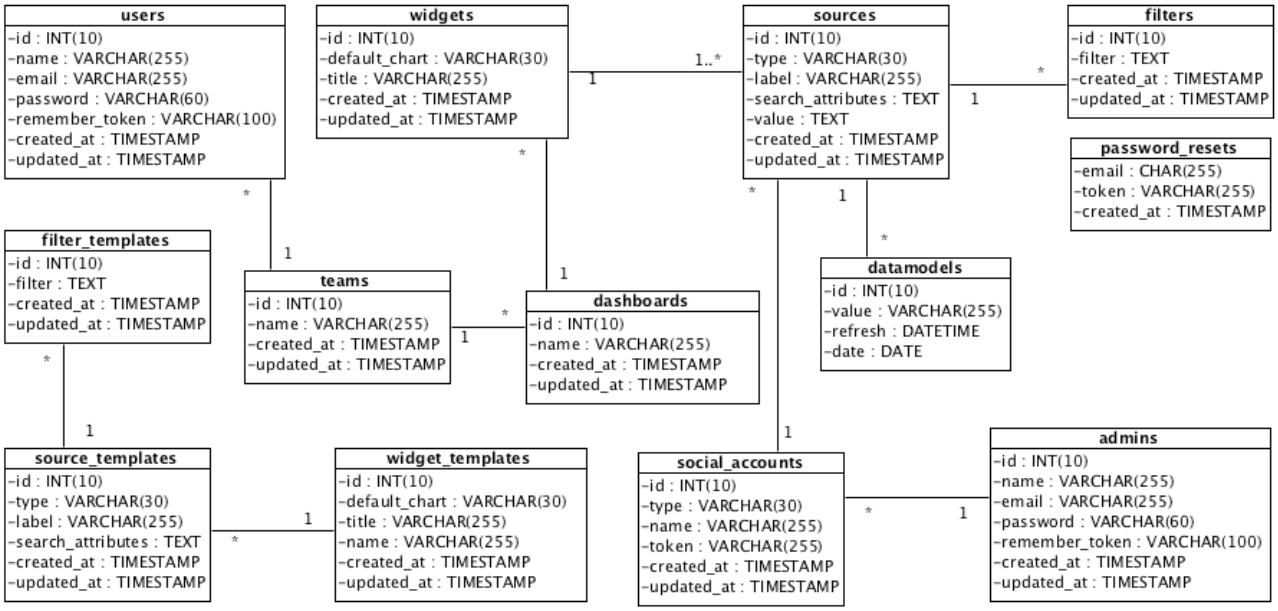
In sprint 4 worden de filters toegevoegd. Een filter wordt op een source toegepast en er kunnen er meerdere zijn. Er is dus een één op meer relatie vereist. Doordat je filters instelt tijdens het maken een template komt er ook een filter template. Dit heeft geleid tot een nieuw concept diagram. (figuur 8.2)



Figuur 8.2 - Concept diagram sprint 4.

Filter templates en filter krijgen een filter attribuut. In dit attribuut wordt aangegeven waarop gefilterd moet worden. Wanneer je bij Google Analytics bijvoorbeeld alleen data afkomstig uit het land Nederland wil hebben, wordt de waarde van de filter: **ga:country=@Netherlands.**

Na de wijzigingen toegepast te hebben in het RRM en RIM en de database gebouwd te hebben. Heb ik er de volgende database aan overgehouden.

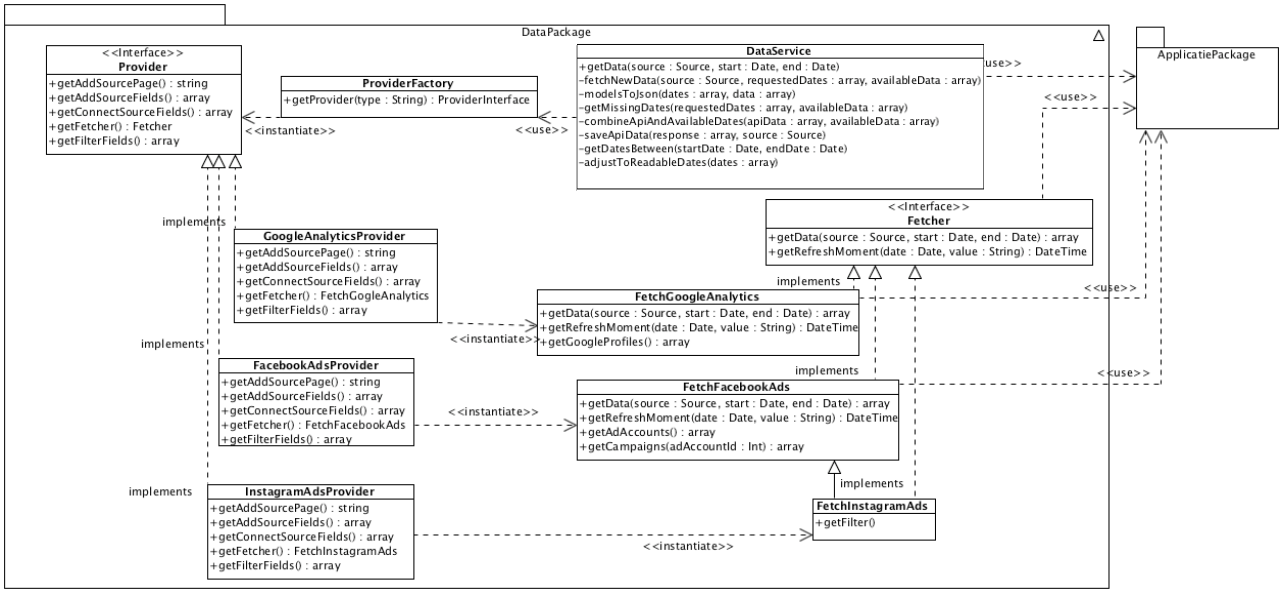


Figuur 8.3 - Database sprint 4.

8.4.2 Applicatie

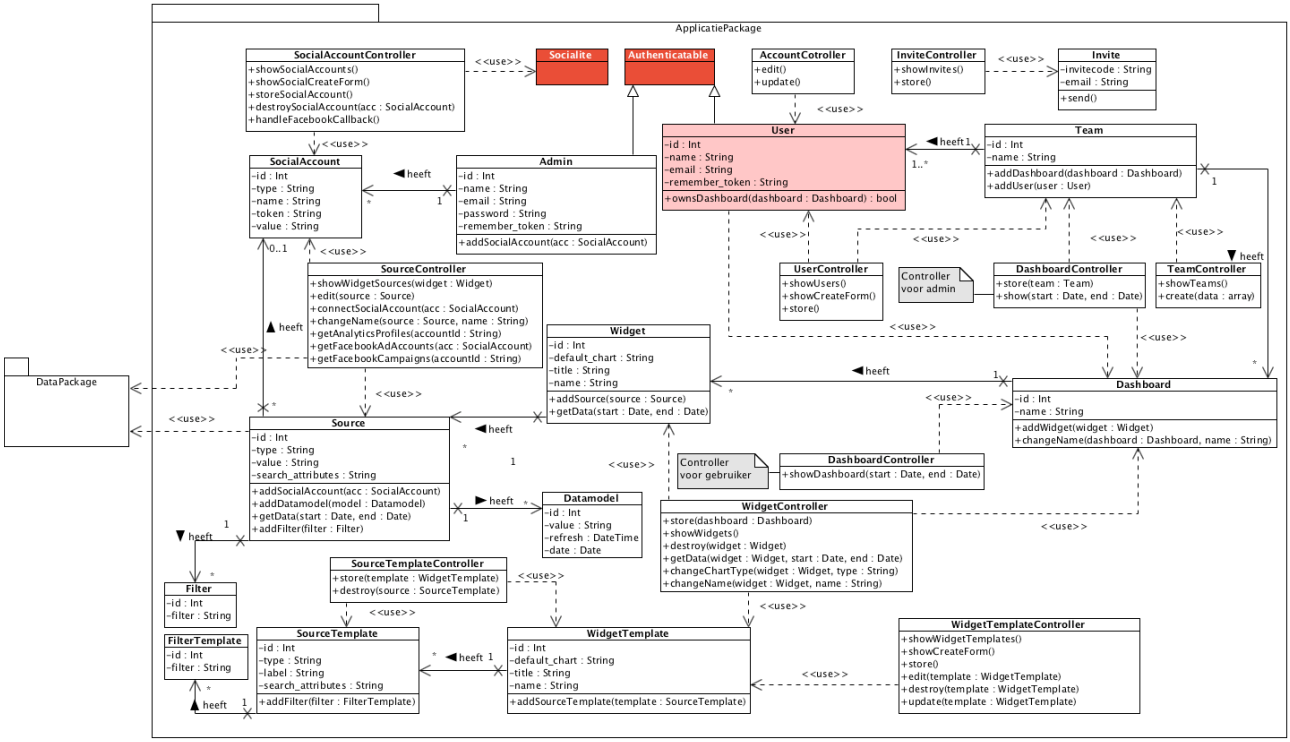
Bij het ontwerpen van de applicatie ga ik voor iedere provider een getFilterFields() toevoegen. Deze haalt de filter gegevens op en slaat de filters op. Daarnaast voeg ik een nieuwe provider toe voor de Instagram Pages API en een fetcher voor de Instagram Ads API. Doordat de Instagram Ads API in principe de Facebook Ads API is, alleen een hard coded filter nodig heeft, laat ik de Instagram Ads fetcher overerven van de Facebook Ads fetcher.

Hierdoor breid ik functionaliteit uit zonder bestaande code te wijzigen. Deze wijzigingen zijn te zien in figuur 8.4.



Figuur 8.4 - Design diagram, datapackage sprint 4.

In de ApplicatiePackage heb ik ook kleine wijzigingen doorgevoerd voor de filter templates. Dit is te zien in figuur 8.5.



Figuur 8.5 - Design diagram, applicatiepackage sprint 4.

8.5 Bouwen

8.5.1 Grafieken

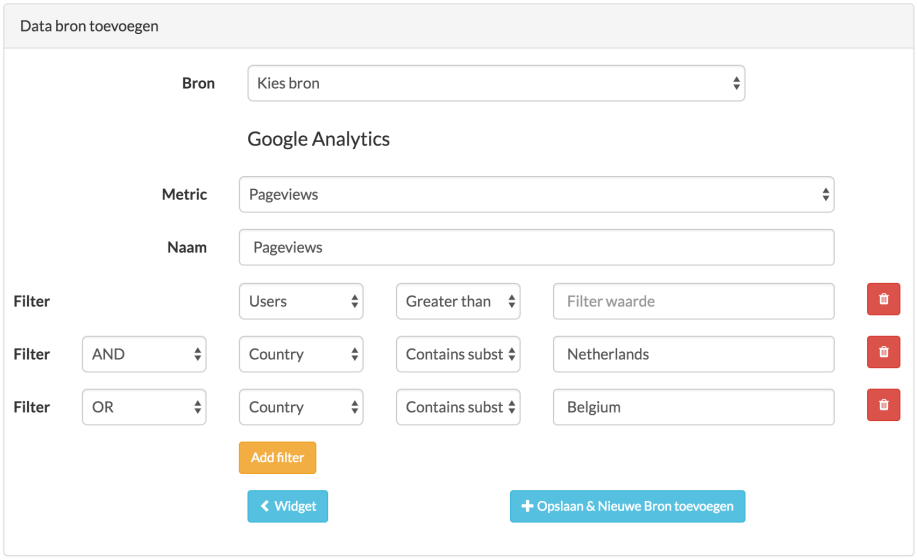
De grafieken moesten in deze sprint duidelijker weergeven welke data ze bevatte. We hebben besloten dit te doen door een legenda toe te voegen en kleuren toe te voegen. De kleuren worden random gegenereerd bij het laden van de pagina. De legenda wilde de opdrachtgever graag overnemen van Cyfe, daar staat de legenda horizontaal boven de grafiek zoals te zien in figuur 5.2. Het resultaat is te zien in het plaatje hier rechts. Deze kun je vergelijken met de grafiek uit figuur 8.1.

Figuur 8.6 Widget na toevoegen kleuren en legenda.

8.5.2 Filter

Wanneer de admin kiest voor een nieuwe widget template en daaraan een bepaalde bron toevoegt, moeten er nu ook filters toegepast kunnen worden. De filters waaruit de admin keuze heeft verschillen per API.

In het plaatje hieronder is te zien hoe ik het heb gebouwd voor Google Analytics. Je kunt dus zelf kiezen hoeveel en welke filters je toepast.



Figuur 8.7 Screenshot van applicatie, filters toevoegen.

De filter input velden kunnen per Provider verschillen, zo is er bij facebook geen verschil met AND en OR. Welke velden de backend moet ophalen, wordt in de functie getFilterFields in de desbetreffende provider neergezet.

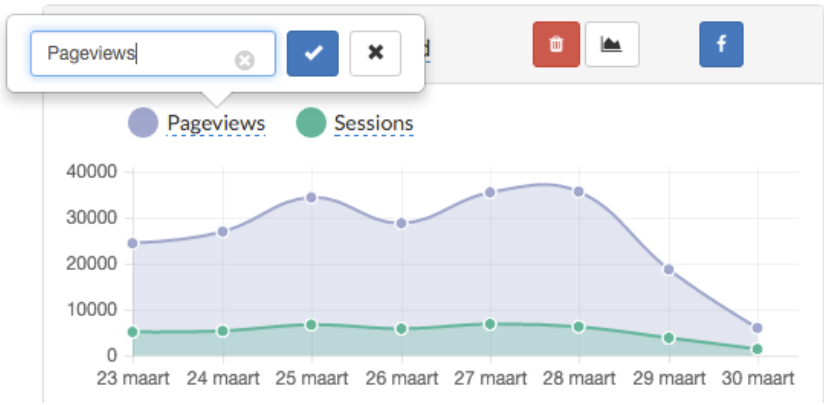
8.5.3 Inline edit

Een taak die deze sprint ingepland was, was inline edit van namen en titels op het dashboard. Dit betekent dat je niet naar een aparte pagina met formulier hoeft te gaan om iets aan te passen. Dit zorgt ervoor dat de dashboards makkelijker te configureren zijn door de SOCIALR admin.

Om dit te realiseren heb ik gebruik gemaakt van de javascript library X-editable. Het werkt door een AJAX request naar mijn backend te sturen, dit zorgt ervoor dat de pagina niet herladen hoeft te worden.

De inline edit is toegepast op de widget en source naam/titel. Hiervoor zijn er ook in de controllers van de twee modellen de functie changeName toegevoegd.

Het resultaat is te zien in figuur 8.8.



Figuur 8.8 Voorbeeld inline edit op source naam

8.6 Testen

8.6.1 Uitgevoerde tests

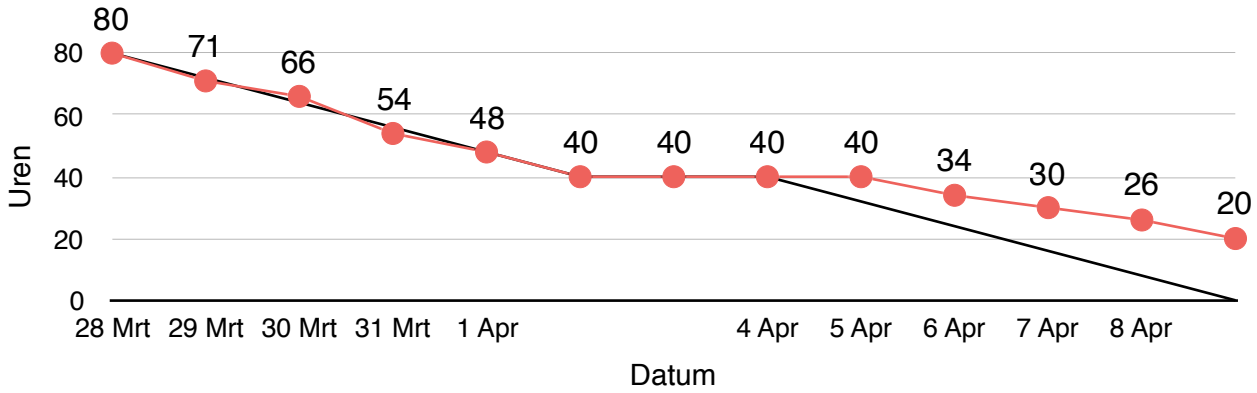
#	Naam test	Beschrijving
1	admin_can_create_filter_templates	Er wordt getest of een admin filter templates kan aanmaken.
2	fillter_attached_to_request	Er wordt getest of filters correct worden meegegeven aan de requests.
3	admin_can_change_source_title	Er wordt getest of de admin source titels kan wijzigen.
4	admin_can_change_widget_title	Er wordt getest of een admin een widget titel kan wijzigen.
5	user_cannot_change_source_title	Er wordt getest of een gebruiker een widget titel niet kan wijzigen.
6	user_cannot_change_widget_title	Er wordt getest of een gebruiker een widget titel niet kan wijzigen.
7	handle_instagram_data	Er wordt aan de hand van dummy data getest of de applicatie Instagram data op een correcte manier verwerkt.

8.7 Reflectie

8.7.1 Burn down

De eerste week van sprint 4 heb ik mij gefocust op het mooier en duidelijker maken van de grafieken. Daarnaast heb ik de Instagram Ads API gekoppeld. Deze taken gingen goed.

Om de Instagram Ads connectie te maken moest ik een filter hard coded toevoegen. Toen ik de filters ook mogelijk wilde maken voor Facebook Ads en Google Analytics kwamen er een aantal problemen. Het duurde langer dan verwacht, vooral doordat beide andere filters hanteren. Je kunt niet zomaar ieder filter op iedere metric gebruiken. Hiervoor moest ik veel code schrijven om te



Figuur 8.9 - Burn down chart sprint 4.

valideren dat de gebruiker toegestane filters toepast.

In figuur 8.9 is de burn down chart te zien. Doordat de filters langer duurde dan verwacht, heb ik de onderdelen voor dashboard templates en het wijzigen van Facebook campagnes niet kunnen doen.

Ondanks dat er onderdelen niet gelukt zijn, was de opdrachtgever erg positief tijdens de sprint meeting. Wat betreft de hoeveelheid gekoppelde API's komen we nog niet in de buurt van Cyfe maar wat betreft het samenstellen van widgets, overtreffen we Cyfe.

De opdrachtgever is ook positief over de manier waarom het systeem uit te breiden is en over hoe de grafieken er nu uit zien. Voor de volgende sprint heeft de opdrachtgever aangegeven volledig te willen focussen op uitbreiden met API's.

9. Sprint 5 | Instagram en Facebook Pages

9.1 Requirements

De vorige sprint zijn er twee onderdelen niet gelukt: het wijzigen van een Facebook campagne op een actieve widget en het samenstellen van dashboard templates.

Normaal zou ik deze onderdelen automatisch meenemen naar de eerst volgende sprint, echter heeft de opdrachtgever laten weten dat hij voor deze sprint liever de koppeling met meerdere API's ziet. Instagram Pages en Facebook Pages.

Over de Instagram Pages API heeft de opdrachtgever laten weten alleen geïnteresseerd te zijn in de volgende gegevens:

- Het aantal mensen dat een account volgt
- Het aantal mensen dat een account volgen
- Het aantal geplaatste media door een account

Hierbij heeft hij ook vermeld dat het niet mogelijk is om de data uit de geschiedenis op te vragen. Je kunt dus niet opvragen hoeveel volgers iemand drie weken geleden had. Omdat het verloop wel zichtbaar moet zijn, zal de data iedere dag automatisch moeten worden opgehaald. In tegenstelling tot de eerdere API's waarbij data werd opgevraagd als iemand het dashboard opent.

Dit heeft geresulteerd in een nieuw item:

Item	Beschrijving	Prioriteit	Uren
25	Automatisch Instagram data ophalen	M	8

9.2 Planning

Doordat ik inmiddels meer ervaring heb opgedaan met het koppelen van API's en een applicatie heb ontwikkeld waarin het toevoegen van nieuwe API's simpel moet gaan. Hebben we besloten dat we voor deze sprint twee API's kunnen inplannen, Facebook Pages API en Instagram Pages API. Er zullen geen andere onderdelen worden ingepland.

9.2.1 Detailplanning

In verband met Koningsdag (27 April) zijn er voor deze sprint 72 uren ingepland.

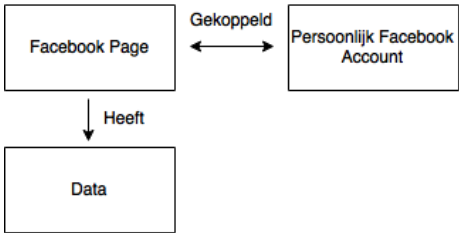
Taak	**Schatting**
Uitzoeken Instagram Pages API	8
Uitzoeken Facebook Pages API	8
Instagram Pages inbouwen in applicatie	10
Bouwen automatisch ophalen data Instagram	8
Bouwen Instagram Pages	10
Bouwen Facebook Pages	10
Testen	8
Scrum meeting & voorbereiden volgende sprint	10
Totaal	72

9.3 Uitzoeken APIs

9.3.1 Facebook Pages

Ik begin met het bekijken en beschrijven van een API. De eerste is de Facebook Pages API.

Om toegang te krijgen tot een Facebook Page, moet je toegevoegd worden als admin. De data is vervolgens gekoppeld aan een page. Dit lijkt een beetje op de Facebook Ads API, alleen zit er hier geen campagne tussen.



Figuur 9.1 - Facebook pages structuur.

Data
De data in de Facebook Pages API is hetzelfde opgebouwd als de data in de Facebook Ads API. Je hebt metrics die een waarde teruggeven en metrics die een collectie teruggeven (figuur 5.6). Om waardes uit de collectie te krijgen zul je een nog specifiekere gegeven moeten meegeven.

De specifiekere waarde noemen ze een action type, in figuur 5.6 heb ik weergeven hoe dit eruit ziet. Het verschil met de Facebook Pages API is dat de lijst met mogelijke action types hier groter is. Je kunt bij bepaalde metrics bijvoorbeeld een leeftijdsgroep meegeven, bij andere kun je een regio meegeven. Ik zal moeten afvangen dat de gebruiker niet een regio meegeeft, wanneer er bijvoorbeeld alleen leeftijdsgroepen geaccepteerd worden. Het is niet mogelijk om filters toe te passen.

Data ophalen
Om data op te halen moet ik de volgende gegevens meegeven:

- Page ID
- Metric
- optionele action type
- access token

De accesstoken is precies dezelfde als die voor de Facebook Ads API. De OAuth functionaliteit die ik voor die API gebouwd heb kan hier dus opnieuw gebruiken. De data zal ik ophalen via de officiële pages SDK.

Limieten
Het limiet is 200 calls gebruiker per uur[RF03]. Dit limiet staat los van de Facebook Ads API limieten.

9.3.2 Instagram Pages

Bij het uitzoeken van de Instagram Pages API liep ik tegen een probleem aan. Om met de API te kunnen communiceren moet ik een applicatie in hun developer omgeving aanmaken. Om deze applicatie te activeren moet ik een formulier invullen en een video maken van mijn applicatie. Op basis daarvan beslist Instagram of mijn applicatie wordt goedgekeurd. Dit kan een aantal weken duren.

Doordat dit veel tijd zou kunnen kosten besloot ik een korte meeting in te plannen met de opdrachtgever. Tijdens deze meeting hebben we besloten dat we de gegevens ook kunnen krijgen door HTML te scrapen. Wanneer je naar mijn Instagram pagina gaat (instagram.com/donkooijman) zie je daar mijn aantal volgers, aantal mensen die ik zelf volg en het aantal geplaatste media. Dit is de data die de opdrachtgever wil hebben. Wanneer je de gegevens gaat scrapen navigeer je via code naar de gewenste URL, download je de HTML en ‘scrape’ je de gewenste data uit de HTML.

Ik heb hierbij aangegeven dat wanneer Instagram de HTML op hun pagina wijzigt, er een kans is dat het scrapen niet meer werkt. Ondanks dit wilde de opdrachtgever toch deze methode gebruiken. Het heeft ook een bijkomend voordeel, er is geen limiet om een URL te bezoeken.

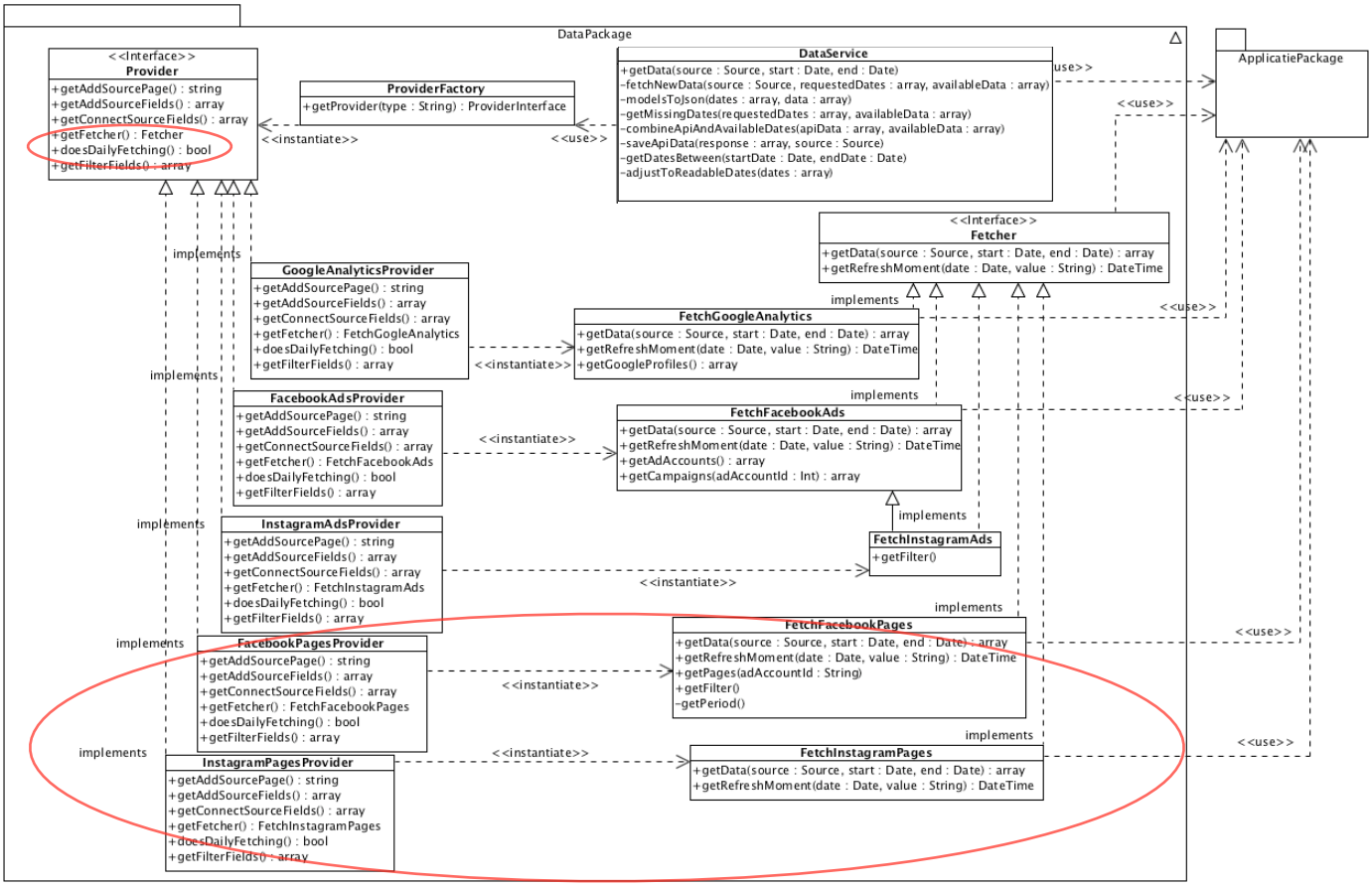
De eerder gekoppelde API's halen data op wanneer het dashboard geopend wordt. Als ik dezelfde methode hier gebruik en het dashboard een aantal dagen niet open, dan heb ik over die dagen geen data. Het is namelijk niet mogelijk om data te scrapen van een week geleden. Om dit op te lossen hebben we besloten dat de applicatie automatisch dagelijks data van Instagram Pages ophaalt en opslaat.

Om te kunnen scrapen heb ik een library op github gevonden die de technisch begeleider heeft goedgekeurd. Hierdoor zal ik niet zelf de scrape functionaliteit moeten schrijven.

9.4 Ontwerp

9.4.1 Data package

Door het ontwerp dat ik eerder gemaakt heb kan ik de applicatie gemakkelijk uitbreiden met nieuwe API's. Ik voeg voor beide API's een provider en fetcher toe, in figuur 9.2 is te zien hoe dat eruit gaat zien. De FetchFacebookPages klasse communiceert met de Facebook Pages SDK. De FetchInstagramPages klasse communiceert met de Instagram scrape library.

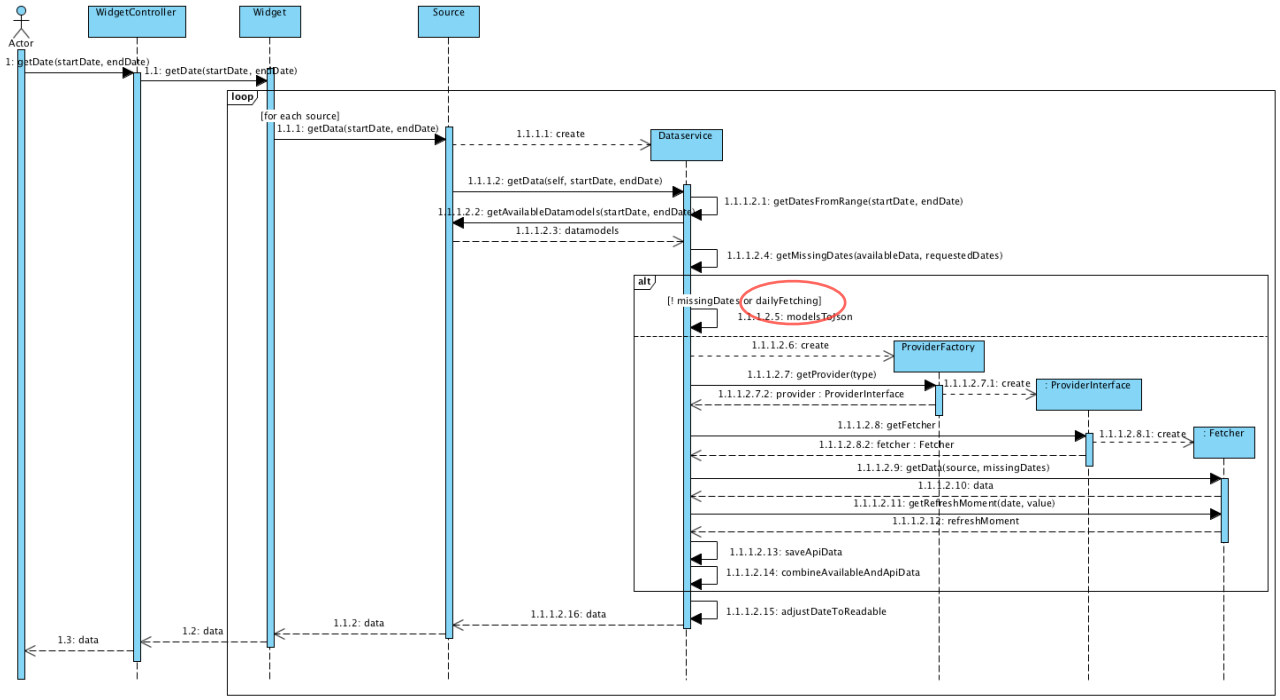


Figuur 9.2 - DataPackage sprint 5.

De enige wijziging in de applicatie package is een functie om pages op te halen. Deze wordt gebruikt wanneer je een widget wil instellen en een page moet uitkiezen. Doordat het maar één functie is geef ik niet de hele package weer in dit diagram, hij is wel te vinden in de bijlage.

In het diagram is ook te zien dat ik een functie aan de Provider interface heb toegevoegd. Deze geeft aan of om een provider gaat die dagelijks wordt opgehaald of niet.

Dit had ik nodig om te zorgen dat wanneer het om een provider gaat die dagelijks gegevens ophaalt, ik niet de gegevens ophaal wanneer ik het dashboard open. Ik heb dus een wijziging moeten maken aan de getData() functie in dataservice. In figuur 9.3 is deze wijziging te zien.



Figuur 9.3 - Aangepast sequencie diagram sprint 5.

9.5 Bouwen

9.5.1 Automatisch ophalen Instagram Pages

Voor de Instagram pages API moet ik automatisch data ophalen. Hierdoor heb ik over iedere dag data beschikbaar. Om dit te realiseren ga ik gebruik maken van Laravel Jobs. Deze jobs zijn klassen in je applicatie waarin je gewoon kunt coderen. Normaal gesproken gaat dit wat lastiger doordat je in de server zo genoemde cron jobs moet instellen. Dat is bij Laravel niet nodig.

De job die ik heb toegevoegd heet FetchInstagramPages. Doordat het een job is zal Laravel de handle() functie in deze klasse automatisch dagelijks uitvoeren.

In de handle() functie haal ik alle sources op die van het type instagrampages zijn. Vervolgens wordt voor ieder van die sources de data opgehaald en opgeslagen.

9.5.1 Caching instagram HTML

Bij het ophalen van de Instagram gegevens kwam ik erachter dat het laden van de HTML een aantal seconden duurt. Wanneer je van één pagina meerdere gegevens nodig hebt, moet de pagina meerdere keren opgevraagd worden. Dit komt door mijn widget bestaat uit sources constructie. Elk gegeven is namelijk in een source opgeslagen en iedere source haalt apart gegevens op.

Om dit te versnellen heb ik besloten de HTML pagina's altijd een minuut te cachen, dit betekent dat wanneer ik eerst de volgers opvraag ik de HTML cache. Wanneer ik daarna het aantal volgend en het aantal media nodig heb, dan is de pagina al opgehaald.

Om te kijken hoeveel verschil het cachen maakt heb ik een korte test uitgevoerd. In figuur 9.3 zijn de resultaten te zien.

```
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
2.2313098907471
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
3.001363992691
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
2.0724301338196
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
2.8669090270996
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
2.5483729839325
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
1.1155340671539
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
1.354306936264
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
1.4317109584808
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
1.5376880168915
vagrant@homestead:~/Homestead/klanten_dashboard$ php artisan FetchInstagramPages
1.2026860713959
```

Figuur 9.4 - Testen duur ophalen van Instagram gegevens.

In figuur 9.4 is te zien hoelang het duurt om van één Instagram pagina de volgers, volgend en aantal media op te halen. Het gaat hier dus om een widget die uit drie sources bestaat. Ik heb de test vijf keer uitgevoerd zonder cache (boven rode lijn). Daarna heb ik de test vijf keer uitgevoerd met cachen (onder rode lijn). Je kunt zien dat het vaak net iets meer dan een seconde scheelt.

Dit gaat om één Instagram account. Wanneer de applicatie voor bijvoorbeeld honderd accounts data zou moeten ophalen scheelt dit dus erg veel.

9.5 Testen

Ik heb de volgende tests gemaakt:

#	Naam test	Beschrijving
1	test_can_handle_facebookpages_data	Ik test aan de hand van dummy data of de facebook pages data goed wordt verwerkt.
2	test_can_handle_instagram_data	Ik test aan de hand van dummy data of de instagram pages data goed wordt verwerkt.
3	cronjob_retrieves_instagramdata	Ik test of de job die automatisch dagelijks instagram data moet ophalen naar behoren functioneert.
4	admin_can_save_instagramname	Er wordt getest of de admin de instagram pages account naam kan instellen op een widget / source.
5	admin_can_save_facebookpage	Er wordt getest of de admin pages kan instellen op een widget / source.
6	test_instagrampages_caching_pages	Er wordt getest of het cachen van Instagram Pages naar behoren functioneert.

9.6 Reflectie

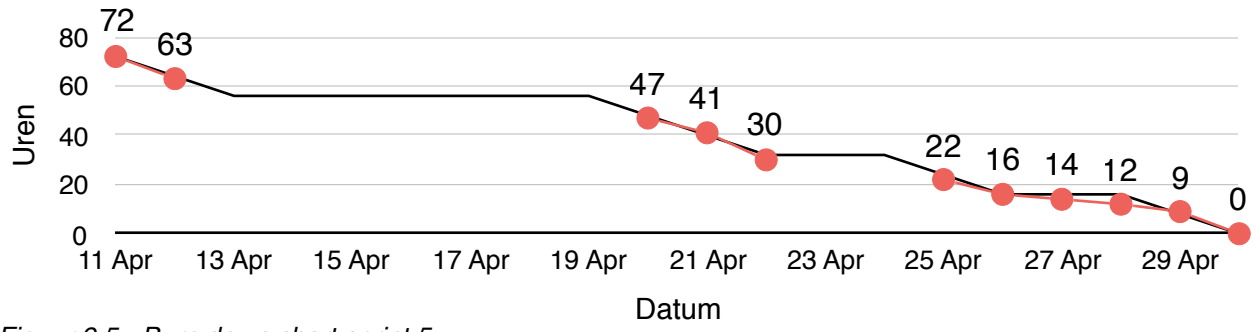
De vijfde sprint stond geheel in het kader van API's toevoegen. Dit is gelukt. Het ontwerp dat ik heb gemaakt (providers en fetchers) maakt het makkelijker en gestructureerd om API's toe te voegen. Dat is niet alleen fijn voor mezelf, maar ook voor de uitbreidingen in de toekomst (na mijn stage).

Het toevoegen van de Facebook Pages API ging zoals gepland. De API sluit aan bij de eerdere gekoppelde API's, hierdoor was de planning die ik ervoor heb gemaakt correct. De koppeling met de Instagram Pages API ging iets anders. Hier moest ik HTML scrapen en data automatisch dagelijks ophalen.

Het HTML scrapen was niet moeilijk te implementeren, doordat ik functionaliteit gescheiden heb gehouden in de dataservice, providers en fetchers, heb ik niet op veel code moet aanpassen. Alleen in de fetcher, deze spreekt nu met de scrape library inplaats van een SDK.

Voor de Instagram Pages API koppeling moest ik ook wat bestaande code aanpassen in de dataservice en moest ik een functie toevoegen aan de providers. Dit is in principe niet zoals ik het had ontworpen, want ik wil die code niet aanpassen. De aanpassingen waren overigens minimaal en daarmee maak ik het mogelijk om meerdere API's toe te voegen die dagelijks data moeten ophalen.

De opdrachtgever was erg positief. Er is nu een systeem dat veel vrijheid biedt bij het samenstellen van widgets. Daarnaast ondersteund het systeem nu bijna alle API's die als een Must geprioriteerd waren.



Figuur 9.5 - Burn down chart sprint 5.

10. Sprint 6 | Twitter en afronden applicatie

10.1 Requirements

Voor de zesde (laatste) sprint heb ik de opdrachtgever gevraagd om goed na te denken over wat er nog in de applicatie ontbreekt. Op de koppeling met de Twitter API na, zijn namelijk alle onderdelen gedaan die met een M geprioriteerd zijn.

De opdrachtgever heeft aangegeven dat ik voor de Twitter API, dezelfde scrape methode kan gebruiken als die voor de Instagram Pages API. Van de Twitter API is namelijk ook alleen publieke data nodig (volgers, volgend en aantal tweets). Hiermee bespaar ik tijd die normaal gesproken zou besteden aan het uitzoeken van de API. Het is ook niet mogelijk om oude data op te halen in de Twitter API, ik zal dus net als bij de Instagram Pages API automatisch dagelijks data moeten ophalen. Dit zorgt ervoor dat ik over iedere dag data beschikbaar heb.

Naast de Twitter API ziet de opdrachtgever ook graag de mogelijkheid voor gebruikers om widgets van positie te laten wijzigen. Dit zorgt ervoor dat de klanten de dashboards meer naar hun eigen wens kunnen inrichten.

Als laatste wil de opdrachtgever dat de applicatie een beter design krijgt. Om klanten met het systeem te laten werken moet het SOCIALR uitstralen. De designer van SOCIALR heeft een design gemaakt die ik deze sprint op de applicatie moet toepassen.

10.2 Planning

10.2.1 Detailplanning

In verband met Hemelvaartsdag (5 Mei) zijn er deze sprint 72 uren ingepland.

Taak	Schatting
Twitter scrape library zoeken	2
Bouwen Twitter	8
Design theme toepassen op applicatie	40
(UA) UR 4.0 Als user wil ik de indeling van dashboards kunnen wijzigen.	6
(A) UR 17.7 Als admin wil ik twitter kunnen selecten als bron voor data	
Laatste sprint meeting, voorbereiden presentatie & uiteindelijke presentatie applicatie	8
Testen	8
Totaal	72

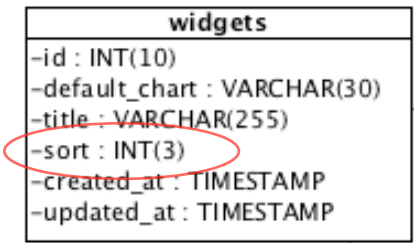
10.3 Ontwerp

10.3.1 Verplaatsen widgets

Het eerste onderdeel waarmee ik aan de slag ging was het verplaatsen van widgets.

Figuur 10.1 Voorbeeld verplaatsen widgets

In figuur 10.1 is aan de linkerzijde een dashboard te zien met 4 widgets. Het eerste getal op iedere widget is het id van de widget. Het nummer tussen haakjes is hun volgorde nummer. Op de rechterzijde is te zien dat de widget die eerst op plek 4 stond, verplaatst is naar plek 1. Dit betekent dat het volgorde nummer van alle 4 de widgets moet wijzigen.

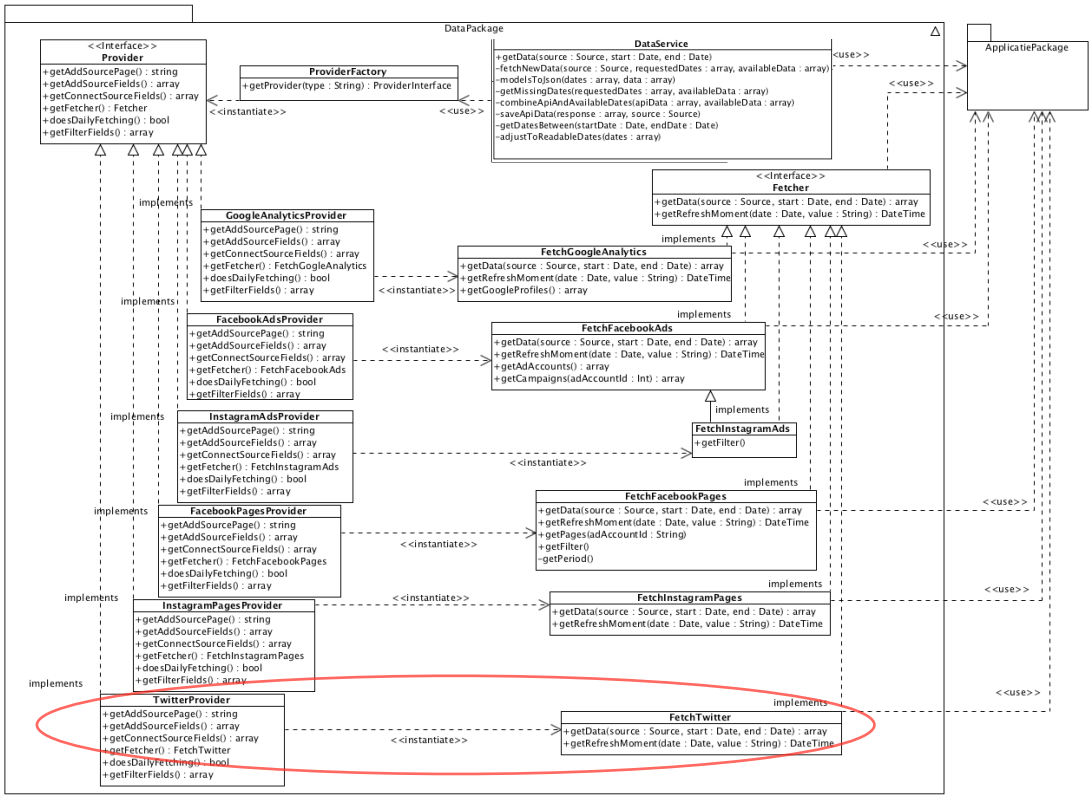


Figuur 10.2 - Sort toegevoegd aan database

Om dit op te kunnen slaan moest ik een volgorde nummer toevoegen aan de widgets in de database. In figuur 10.2 is dit te zien. In de bijlage zijn een aangepast RRM, RIM en Datamodel te vinden.

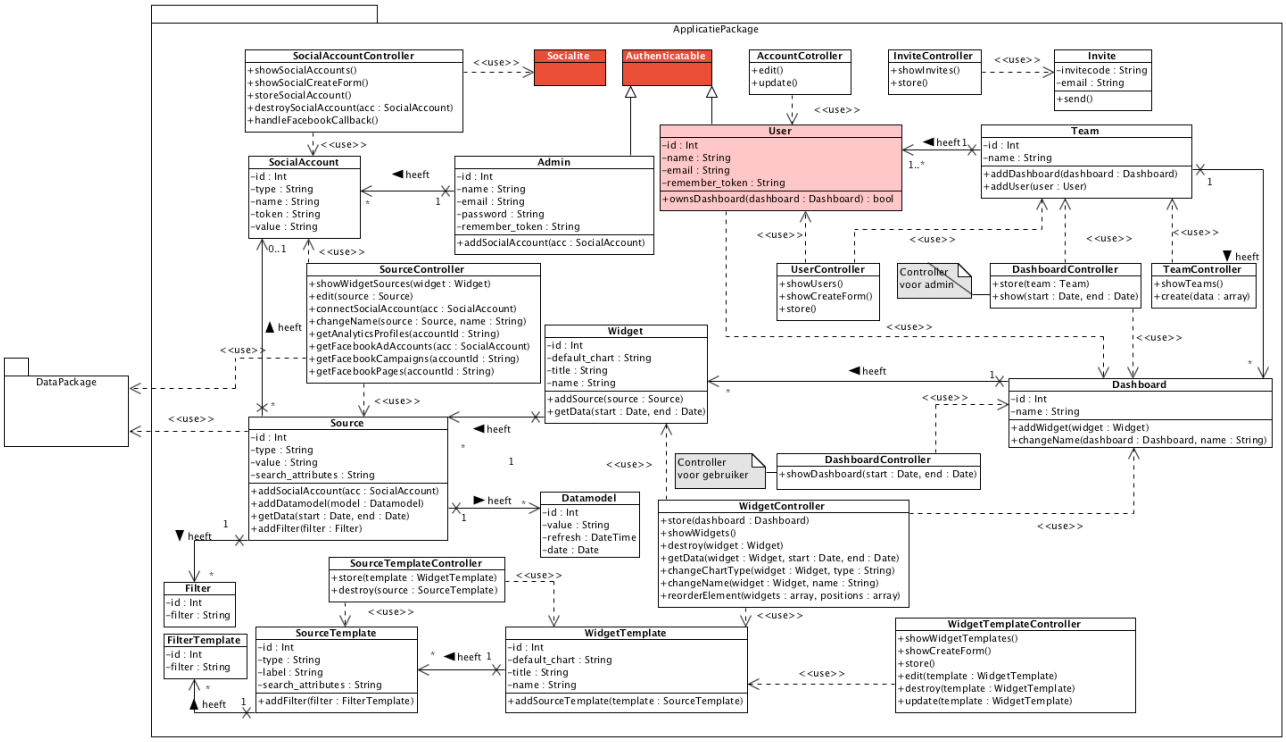
10.3.2 Twitter API

Bij de Instagram API heb ik er een library gevonden die dat scrapen voor mij afhandelt. Na een vlugge zoektocht op github.com vond ik ook een library voor de Twitter API. Ik heb de technisch begeleider om goedkeuring gevraagd om deze library te gebruiken en die heeft hij gegeven. Ik heb een provider en fetcher toegevoegd voor Twitter. De fetcher klasse communiceert met de scrape library. In figuur 10.3 is de nieuwe provider en fetcher te zien.



Figuur 10.3 - Design diagram, DataPackage, Nieuwe Twitter provider en fetcher klasse.

In de ApplicatiePackage zijn alleen het widget model aangepast voor het nieuwe sort attribuut. Daarnaast is er een functie toegevoegd op de WidgetController om nieuwe volgorde op te slaan. Dit is te zien in figuur 10.4.

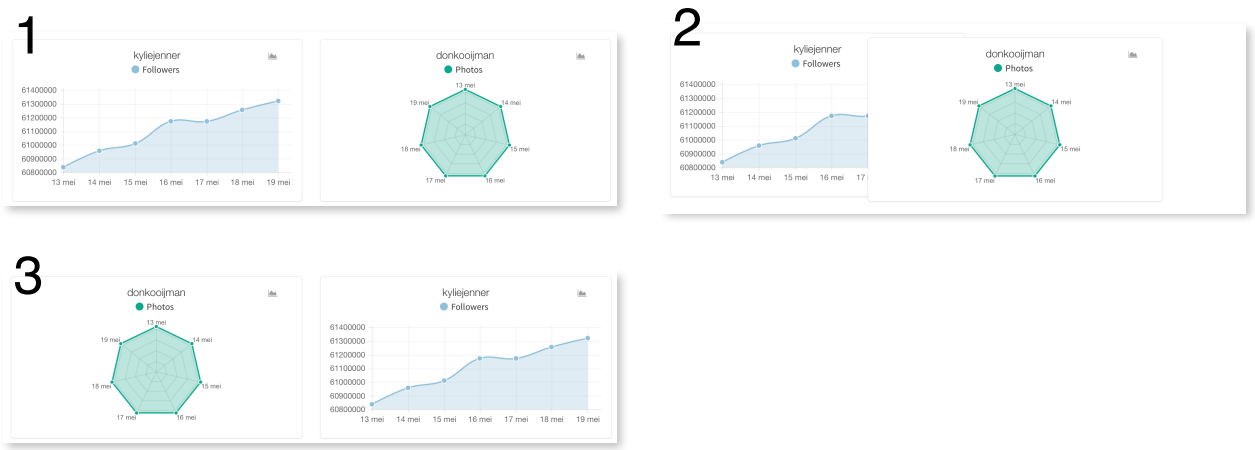


Figuur 10.4 - Design diagram, applicatiepackage sprint 6.

10.4 Bouwen

10.4.1 Verplaatsen widgets

Voor het verplaatsen van de widgets heb ik gebruik gemaakt van Jquery UI. Dit is een javascript library die het makkelijk maakt om gedeeltes van een website te verslepen. Daarnaast bevat het ook een functie om van iedere widget een nieuw sorteer nummer te krijgen. Die kan ik vervolgens meegeven aan de AJAX request naar de backend, zodat de nieuwe posities daadwerkelijk worden opgeslagen.



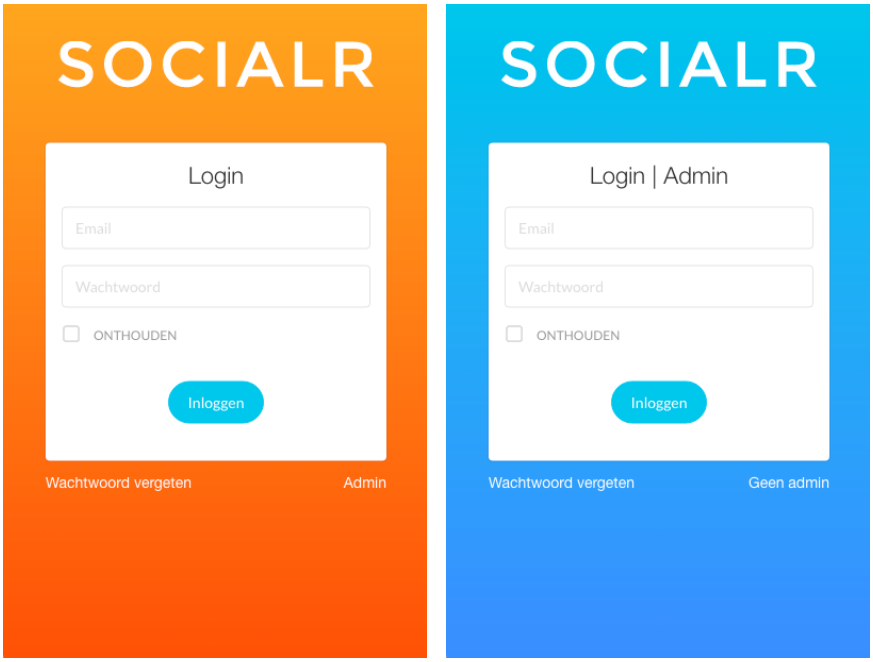
Figuur 10.5 Voorbeeld resultaat verplaatsen van widgets.

10.4.2 Design toepassen op de applicatie

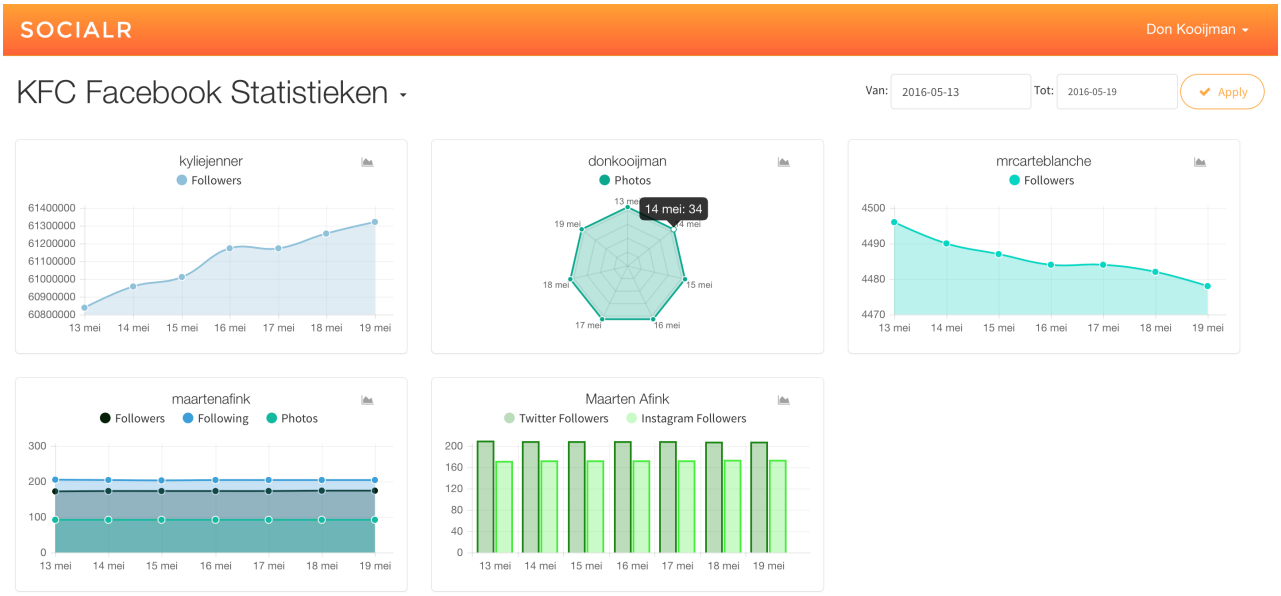
Als laatste heb ik mij gefocust op het toepassen van het design. Ik diende voor alle pagina's CSS te schrijven en de pagina's aan te passen.

Allereerst heb ik mij gefocust op de pagina's die voor klanten zichtbaar zijn. Mocht ik niet genoeg tijd hebben om alle pagina's te doen, dan is het geen probleem als de admin pagina's nog niet gedaan zijn. Deze zijn immers alleen zichtbaar voor mensen van SOCIALR.

Naar mijn mening is het resultaat erg mooi, ik laat u een aantal voorbeelden zien.



Figuur 10.6 - Inlog pagina's. Links gebruiker, rechts admin.



Figuur 10.7 - Dashboard voor gebruiker. De titel van het dashboard is slechts ter demonstratie.

10.5 Testen

10.5.1 Uitgevoerde tests

#	Naam test	Beschrijving
1	admin_can_change_position_widget	Er wordt getest of een admin de positie van een widget kan wijzigen.
2	user_can_change_position_widget	Er wordt getest of de gebruiker de positie van een widget kan wijzigen
3	can_handle_twitter_data	Er wordt getest of de applicatie dummy data van Twitter API kan verwerken. Er wordt getest of de juiste datamodels aangemaakt worden.
4	cronjob_retrieves_twitter_data	Er wordt getest of de job die automatisch twitter data ophaalt naar behoren functioneert.
5	admin_can_add_twitter_to_widget_template	Er wordt getest of de admin Twitter kan toevoegen aan een widget template.
6	admin_can_attach_twitter_username_to_widget	Er wordt getest of de admin een Twitter gebruikersnaam kan koppelen aan een widget.

10.6 Reflectie

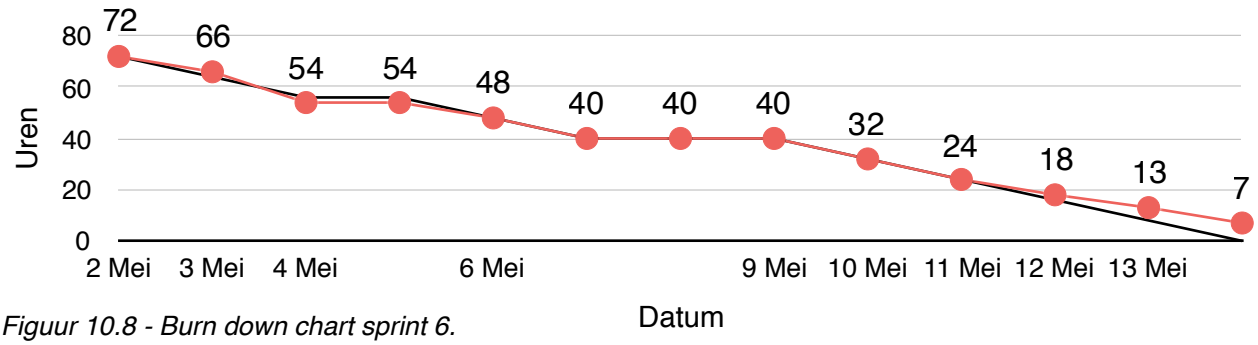
De zesde en tevens laatste sprint is goed verlopen. In tegenstelling tot de eerste sprints heb ik hier niet veel tijd besteed aan het koppelen van een API. Het toevoegen van de Twitter API ging erg goed, het kwam overeen met de Instagram Pages API. Met deze koppeling zijn al de API's die de opdrachtgever met een M had geprioriteerd gedaan.

Toen we deze sprint begonnen heeft de opdrachtgever aangegeven dat de applicatie een beter uiterlijk moest krijgen om klanten ermee te laten werken. De helft van deze sprint heb ik besteedt aan het implementeren van dat design.

Er is één pagina niet gelukt, de dashboard pagina voor admins. Dat is de pagina waarop admins dashboards voor klanten samenstellen. Ik heb mij eerst gefocust op de pagina's voor de klanten. De klanten waren tenslotte de reden dat dit design geïmplementeerd moest worden.

Deze laatste sprint heb ik presentatie van het product gedaan aan alle collega's. De reacties waren erg positief. De opdrachtgever heeft aangegeven dat het geen probleem is dat de admin dashboard pagina niet het zelfde design heeft. De applicatie zal mij stage periode getest worden met een aantal klanten, wanneer de ervaringen en feedback positief zijn, besluit de opdrachtgever Cyfe helemaal te vervangen of niet. Hij heeft aangegeven dat het er nu naar uit ziet, dat hij zeker Cyfe zal vervangen.

In onderstaande figuur is de laatste burn down chart te zien.



Figuur 10.8 - Burn down chart sprint 6.

11. Evaluatie

11.1 Productevaluatie

10.5.1 Applicatie

Ik ben erg tevreden over het product. Uiteindelijk zijn er veel requirements die niet in het product verwerkt zitten. Echter alle requirements die met een Must geprioriteerd waren zijn dat wel.

Het systeem is zo ontworpen en gebouwd dat je bij het toevoegen van een nieuwe API geen bestaande code dient aan te passen. Er moeten alleen nieuwe klassen toegevoegd worden. Zelf heb ik daar ook profijt van gehad, omdat ik gedurende het project steeds API's heb toegevoegd.

Uitbreidbaarheid is iets dat ons op school geleerd wordt, nu ik het in de praktijk in zo een groot project heb toegepast geeft dat mij een te gek gevoel.

Uiteindelijk durf ik te zeggen dat het doel van het project behaald is. Alle noodzakelijke features zijn in het systeem aanwezig. Hierdoor kan de opdrachtgever overstappen van Cyfe naar deze eigen op maat gemaakte applicatie. De applicatie is nog niet live. Het bedrijf wil de applicatie zelf in de komende weken online zetten en dan eerst experimenteren met een beperkt aantal klanten.

10.5.2 Requirements

Tijdens dit project heb ik een grote lijst met requirements opgesteld. Ik heb ze niet allemaal kunnen doen dit project maar ze kunnen handig van pas komen om de applicatie in de toekomst uit te breiden.

De requirements heb ik een beetje op me eigen manier aangepakt. Ik heb ze bij bespreken direct in user story notatie geformuleerd. Vervolgens heb ik over die user stories use case beschrijvingen gemaakt. Achteraf gezien ben ik hier wel tevreden mee, er is namelijk nooit onduidelijkheid geweest over wat de opdrachtgever wilde hebben. Dit is hoofdzakelijk te wijten aan het feit dat we de requirements steeds gezamenlijk opstelde en de user story notatie is goed te begrijpen voor beide partijen.

11.2 Procesevaluatie

Tijdens dit project heb ik op aandringen van het bedrijf scrum gevolgd. Scrum is verdeeld in verschillende sprints en het is de bedoeling dat je iedere sprint iets geheel afmaakt. Het is niet de bedoeling dat er zaken door elkaar lopen.

Aan het begin van het project liepen er bij mij echter wel onderdelen door elkaar. Ik ging eerst de Facebook Ads API bekijken, de sprint daarna wilde ik de koppeling daarmee gaan bouwen. Toen besloot ik toch maar eerst Google Analytics API te bekijken. Doordat ik die extra onderdelen van Google Analytics had ingepland ging het bouwen van de Facebook Ads API ook weer verspreid over twee sprints.

De reden dat dit bij mij is gebeurd, is omdat ik weinig tot geen kennis had over de verschillende API's. Om een goed ontwerp te kunnen maken wilde ik eerst de API's bekijken en beschrijven, zodat ik een ontwerp kon maken dat voor beide werkte. Achteraf gezien had ik misschien een langere aanloop tijd nodig gehad. In die periode had ik de API's beter kunnen bestuderen, waardoor ik tijdens de uitvoeringsfase een meer gestructureerde planning had kunnen aanhouden.

Ondanks dit ben ik toch tevreden met scrum. Aan het begin van het verslag vertel ik namelijk dat scrum uitstekend werkt wanneer de requirements van te voren niet goed duidelijk zijn. Doordat je in iteraties werkt heeft de opdrachtgever aan het eind van iedere sprint de mogelijkheid om requirements te wijzigen.

In de eerste 4 sprints wijzigde er constant requirements. Wat grappig is om te zien, en waardoor je kunt zien dat scrum goed tot zijn recht is gekomen. Is het feit dat vaak wanneer er nieuwe requirements bij kwamen, dit ook direct de requirements waren die in de eerst volgende sprint waren ingepland. Eind sprint 3 kwam er bijvoorbeeld de requirement voor filters bij, deze vond de opdrachtgever direct belangrijker dan de eerder opgestelde requirements. De filters werden dus voor de eerste volgende sprint (sprint 4) ingepland.

Wanneer ik een methodiek had aangehouden waarbij requirements niet constant gewijzigd konden worden, dan had ik de requirements in de eerste weken opgehaald. De filters waren dan bijvoorbeeld nooit bekend geworden.

Ondanks dat de zaken in de eerste sprints door elkaar hebben gelopen, ben ik tevreden met scrum. Het heeft ervoor gezorgd dat de juiste requirements bekend zijn geworden en de gewenste applicatie is gebouwd.

11.3 Evaluatie beroepstaken

11.3.1 Uitvoeren analyse door definitie van requirements (BT 1.4)

Niveau 3 (lastig / zelfstandig)

Tijdens de eerste sprint heb ik veel requirements opgehaald. Er waren enorm veel ideeën die beschreven zijn als user story. Het was uiteindelijk belangrijk om samen met de opdrachtgever uit te vogelen welke onderdelen essentieel waren voor de applicatie. De opdrachtgever heeft dit gedaan door de onderdelen te prioriteren.

Tijdens elke sprint meeting zijn de requirements opnieuw bekeken. Vooral in de eerste paar sprints kwamen er daardoor steeds nieuwe requirements bij. Het feit dat de opdrachtgever de mogelijkheid heeft gehad om steeds requirements toe te voegen of te wijzigen is een goed punt. Per sprint heb ik beschreven welke requirements erbij komen en waarom. Daarnaast is er een overzichtelijke lijst met uiteindelijke requirements toegevoegd in de bijlage.

Ik denk dat ik de beroepstaak op niveau 3 heb behaald doordat ik de benodigde documenten heb opgeleverd. Iedere requirement is beschreven als user story of item. Indien nodig zijn de user stories ook uitgeschreven in use case beschrijvingen. Dit heeft er in geresulteerd dat er weinig tot geen misverstanden hebben bestaan omtrent de inhoud van de requirements.

11.3.2 Ontwerpen, bouwen en bevragen van een database (BT 2.2)

Niveau 3 (lastig / zelfstandig)

Het ontwerpen van de database is incrementeel verlopen. Iedere sprint heb ik indien nodig nieuwe klassen toegevoegd. Die heb ik vervolgens vertaald naar de database via een RRM en RIM.

De applicatie zou steeds uitgebreid worden met verschillende API's. De gegevens van deze API's moeten worden opgeslagen in de database. De hoeveelheid gegevens en de namen van de gegevens verschillen per API. Doordat het bedrijf vereiste dat ik met MySQL werkte, vormde dit een probleem. Het zou betekenen dat ik voor nieuwe API's de database zou moeten aanpassen. De oplossing die ik heb gevonden is het opslaan van JSON in een MySQL database. Hierdoor kan ik meerdere gegevens opslaan in één column.

Ik denk dat ik de applicatie op niveau 3 heb behaald doordat ik de benodigde stappen heb doorlopen om van een concept diagram naar een database te gaan. Daarnaast heb ik database ontworpen die compatible is met verschillende API's door het slimme gebruik van JSON velden. Het is daardoor niet nodig om de database aan te passen of te wijzigen wanneer de applicatie wordt uitgebreid met een nieuwe API. Hier heb ik op een creatieve manier een dynamische database gemaakt maar me toch aan de eis van de opdrachtgever gehouden om MySQL te gebruiken.

11.3.3 Ontwerpen systeemdeel (BT 3.2)

Niveau 3 (lastig / zelfstandig)

Bij het ontwerpen van de applicatie heb ik rekening gehouden met de uitbreidbaarheid van nieuwe API's. De ontwerpen heb ik zichtbaar gemaakt in UML diagrammen.

De verantwoordelijkheid van klassen heb ik goed gescheiden gehouden. Daarnaast heb ik ervoor gezorgd dat er minimale wijziging in code nodig is door het gebruik van de Simple Factory design pattern. Dat dit heeft gewerkt, heb ik tijdens de verschillende sprints gemerkt.

Ik denk dat ik de beroepstaak op niveau 3 en gedeeltelijk op niveau 4 heb afgerond. Het systeem is makkelijk uit te breiden door de scheiding van verantwoordelijkheden en het gebruik van een design pattern.

11.3.4 Bouwen applicatie (BT 3.3)

Niveau 4 (complex / zelfstandig)

Ik heb vanuit het niets een applicatie ontwikkelt. Hierbij heb ik gebruik gemaakt van het Laravel framework. Om de benodigde onderdelen te bouwen heb ik verschillende technieken van het framework gebruikt, bijvoorbeeld: caching en traits.

Om data te kunnen weergeven moest mijn applicatie met verschillende API's communiceren. Hierbij heb ik gebruik gemaakt van verschillende libraries / SDK's. Voorbeelden zijn Facebook Ads SDK en Google Analytics SDK.

De data van deze verschillende SDK's en libraries kwamen in verschillende structuren. Ik heb complexe code moeten schrijven om de data geschikt te maken voor het weergeven en opslaan in mijn applicatie.

Ik heb ook een aantal verschillende libraries gebruikt voor de frontend van de applicatie en het geheel heb ik bijgehouden in versiebeheer, git.

Vooral het feit dat ik met een nieuw framework en veel verschillende SDK's en libraries heb moeten werken toont aan dat ik deze beroepstaak op niveau 4 heb behaald.

12. Bronnenlijst

12.1 SDK's, packages & libraries

Laravel Socialite (v2.0)
https://github.com/laravel/socialite

Facebook Ads SDK (v2.5.0)
https://github.com/facebook/facebook-php-ads-sdk

Google Analytics SDK (v2.0.0)
https://github.com/google/google-api-php-client

Facebook Pages SDK (v5.0)
https://github.com/facebook/facebook-php-sdk-v4

Instagram pages scraping library (v0.1.0)
https://github.com/raiym/instagram-php-scraper

Twitter scraping library (v1.0)
https://github.com/Saleh7/ScrapeTwitter

ChartJS (v1.0.2)
Frontend grafieken
http://www.chartjs.org

X-editable (v1.5.0)
Frontend aanpassen namen widgets en sources
https://vitalets.github.io/x-editable/

JQuery UI (v1.12.0)
Frontend verplaatsen volgorde van widgets
https://jqueryui.com

12.2 Referenties

RF01 Facebook Ads API limieten
https://developers.facebook.com/docs/marketing-api/api-rate-limiting

RF02 Google Analytics API limieten
https://developers.google.com/analytics/devguides/reporting/core/v4/limits-quotas#core_reporting

RF03 Facebook Pages API limieten
https://developers.facebook.com/docs/graph-api/advanced/rate-limiting