MILVUM

Afstudeerverslag

Het ontwikkelen van een urenregistratie systeem

Auteur: Karim Ben Dahman

Student nummer: 11116730

Opleiding: Informatica

School: De Haagse Hogeschool

Eerste examinator: M.G. Maris

Tweede examinator: G.A. Mijnarends

Opdrachtgever: A. Jagesser

Bedrijfsbegeleider: A. Jagesser

Plaats: Den Haag

Bedrijf: Milvum

Datum: 21-03-2016

Referaat

Ben Dahman, Karim, Het ontwikkelen van een urenregistratie systeem, Den Haag, Milvum, 2015/2016

Afstudeerverslag van Karim Ben Dahman, geschreven in het kader van het afstuderen bij de opleiding Informatica aan de faculteit IT & Design aan de Haagse Hogeschool te Den Haag.

Het verslag behandelt het proces dat doorlopen is tijdens de afstudeerperiode van Karim Ben Dahman bij Milvum te Den Haag. Deze opdracht stond in het teken van het ontwikkelen van een urenregistratie systeem dat bestaat uit een Android applicatie en een webapplicatie.

Descriptoren:

- Scrum
- React Native
- JavaScript
- GraphQL
- Relay
- REST
- MySQL
- React
- Webapplicatie

Voorwoord

Dit afstudeerverslag is geschreven door Karim Ben Dahman. Dit verslag is geschreven in het kader van mijn afstuderen aan de opleiding Informatica aan de Haagse Hogeschool te Den Haag. De opdracht is gegeven door Milvum te Den Haag en is uitgevoerd in de periode van 16 november 2015 tot en met 18 maart 2016.

Ik wil graag Milvum bedanken voor het beschikbaar stellen van de afstudeeropdracht, ook gaat mijn dank uit naar dhr A. Jagesser voor de fijne begeleiding en ondersteuning tijdens deze periode. Tevens wil ik mijn examinatoren M.G. Maris en G.A. Mijnarends bedanken voor de gekregen feedback tijdens de stageperiode.

Karim Ben Dahman
Rijswijk, 21-03-2016

Inhoudsopgave

1 Inleiding

In dit afstudeerverslag word het proces beschreven van het uitvoeren van mijn afstudeeropdracht: Het ontwikkelen van een urenregistratie systeem. Het doel van dit verslag is dat het proces dat doorlopen is tijdens de opdracht beoordeeld kan worden door de examinatoren en de extern gecommitteerde.

Het verslag is opgedeeld in 3 delen. In het eerste deel, hoofdstuk 2 tot en met 5, van het verslag wordt achtergrond informatie van het bedrijf gegeven. Ook worden de opdracht, de aanpak en de voorbereidende werkzaamheden omschreven. In het tweede deel, hoofdstuk 6 tot en met 10, van het verslag wordt de uitvoering van de opdracht beschreven. In het laatste deel, hoofdstuk 11 tot en met 13, van het verslag worden de producten, het proces en de beroepstaken geëvalueerd.

2 Organisatie

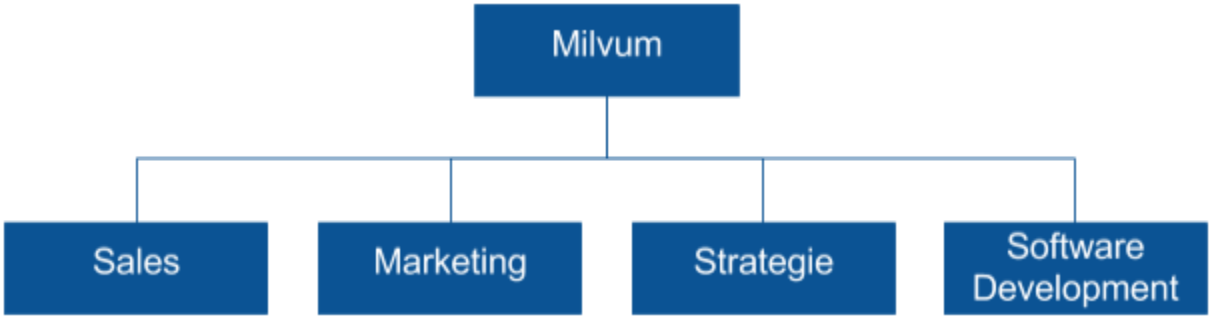
Milvum is een software ontwikkelingsbedrijf dat in 2012 opgericht is door een vriendengroep van vier personen. Het bedrijf zit in een fase van groei en is in een periode van 4 jaar uitgegroeid tot 12 medewerkers. Het bedrijf is gevestigd in Den Haag en richt zich vooral op het ontwikkelen van Business-to-Business(B2B) en Business-to-Employee(B2E) apps.

De missie van het bedrijf is om de omgeving van alle werknemers te beïnvloeden met digitale innovatieve oplossingen voor smart devices. De visie is een wereld waarin de dagelijkse taken van werknemers versoepeld zijn door smart devices, waarbij de echte wereld van een werknemer gemixt is met de digitale wereld.

De belangrijkste klanten van het bedrijf zijn I.T. Works, Radar, PostNL, MI Airline en het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties.

Figuur 1 Belangrijkste klanten van Milvum

De organisatiestructuur van het bedrijf is plat en het bedrijf kan als informeel getypeerd worden, de verschillende afdelingen zijn in het organogram hieronder te zien.



Figuur 1 Organogram

Als afstudeerder behoor ik tot de Software Development afdeling, de bedrijfsbegeleider zit tevens op deze afdeling. De afdeling gebruikt verschillende programmeertalen(waaronder Java en C#) om de applicaties van klanten te realiseren. In de toekomst wil Milvum gebruik maken van een cross-platform oplossing, waardoor de verschillende expertises niet meer nodig zijn. Één van de oplossingen waar het bedrijf veel potentie in ziet is React Native. Dit is een framework waarmee met JavaScript applicaties gemaakt kunnen worden voor iOS en Android.

Het bedrijf hanteert SCRUM als softwareontwikkelmethode. De meeste projecten hebben sprints met een duur van 2 weken en er wordt met teams van 3 medewerkers aan een product gewerkt. De teams houden elke dag samen met de SCRUM Master een daily stand-up. Hierdoor weet het team van elkaar wie waar mee bezig is, wat gedaan is en of er bottlenecks zijn. Aan het eind van een sprint wordt de review, retrospective en planning voor de volgende sprint gedaan.

3 Omschrijving van de opdracht

In dit hoofdstuk wordt de opdracht beschreven. Deze omschrijving komt niet overeen met het goedgekeurde afstudeerplan. Dit komt doordat er van project is veranderd na het afronden van de voorbereidende werkzaamheden. De verandering van project is gemaakt, omdat het nieuwe project (het ontwikkelen van een urenregistratie systeem) urgenter was voor het bedrijf dan het oude project (het ontwikkelen van een applicatie waarmee de waardes van huizen in de buurt bekeken kan worden). De verandering was mogelijk, omdat ik net klaar was met het uitvoeren van de voorbereidende werkzaamheden. Deze werkzaamheden bleven hetzelfde voor het nieuwe project.

3.1 Probleemstelling

Het registreren van de gewerkte uren van medewerkers van Milvum wordt handmatig gedaan in Google Spreadsheets. Hierdoor wordt er soms vergeten uren te registreren en worden fouten gemaakt bij het registreren van de uren. Ook is het lastig om een goed overzicht te behouden door het vele aantal spreadsheets, naar mate de tijd vordert wordt het steeds minder overzichtelijk. Verder vinden de medewerkers van Milvum het vervelend om elke dag hun uren te registreren, omdat het onnodig veel tijd kost. De opdrachtgever vindt de oplossingen die op de markt zijn te omslachtig en complex.

3.2 Doelstelling

Om de foutgevoeligheid van het handmatige urenregistratie systeem weg te nemen wil Milvum dat er een geautomatiseerd urenregistratie systeem komt. Het systeem bestaat uit twee delen: een Android applicatie en een webapplicatie. Op de Android applicatie moeten medewerkers kunnen in- en uitklokken voor projecten door middel van een badge met RFID tag. Op de webapplicatie moeten de geregistreerde uren ingezien en beheerd kunnen worden.

3.3 Resultaat

Een urenregistratie systeem waarbij medewerkers met een RFID tag op een Android toestel met NFC sensor kunnen in- en uitklokken voor projecten en een webapplicatie waarin een overzicht van de geregistreerde uren bekeken en beheerd kan worden. Uiteindelijk moet hierdoor tijd bespaard worden, het proces minder vervelend gemaakt worden voor medewerkers en het overzicht over de gewerkte uren behouden worden.

3.4 Scope

Eerst wordt de Android applicatie ontwikkeld, de applicatie moet met behulp van het React Native framework gemaakt worden. Bij het ontwikkelen van de applicatie wordt eerst gefocust op de core functionaliteit, het kunnen in- en uitklokken. Pas als de applicatie naar wens werkt mag er begonnen worden met het ontwikkelen van de webapplicatie. Bij de webapplicatie gaat het vooral om een overzicht van de gewerkte uren te kunnen zien en het beheren.

3.5 Huidige urenregistratie proces

Het huidige urenregistratie systeem is relatief simpel opgezet. Iedere werknemer noteert voor een bepaalde dag in een bepaalde maand in een persoonlijke Google Spreadsheet de datum, de begin tijd, de eindtijd en hoe lang er is gepauzeerd. Op basis hiervan worden de uren van een bepaalde dag uitgerekend en uiteindelijk het aantal gewerkte uren in een maand. Elk werkblad is een aparte maand. Het totaal aantal uur per maand per persoon wordt vervolgens door de administratie medewerker doorgestuurd naar de accountant. Zie het figuur hieronder voor een voorbeeld:

Datum	Begin	Eind	Pauze	#Uren
3/10/2013	9:15:00	18:00:00	0:30:00	8:15:00
8/10/2013	9:15:00	17:00:00	0:30:00	7:15:00
10/10/2013	9:15:00	17:45:00	0:30:00	8:00:00
			Totaal aantal uur	**23:30:00**

Figuur 2 Huidige urenregistratie

4 Aanpak en initiële backlog

In het plan van aanpak wordt beschreven wat de opdracht is, welke methodiek er gebruikt zal worden, wat de risicofactoren van het project zijn, wat de scope van het project is en een globale planning gemaakt. Hier worden de gemaakte keuzes besproken en de initiële backlog getoond.

4.1 Methodiek

Bij het maken van het afstudeerplan was duidelijk wat het uiteindelijke resultaat van de opdracht moest zijn, maar de precieze invulling van de opdracht was nog niet tot in detail bekend. Daarom was al duidelijk dat ik in ieder geval met een Agile methodiek te werk zou gaan. Bij een Agile methodiek kan namelijk op tijd bijgestuurd worden bij veranderingen aan de eisen van het project.

Omdat het bedrijf zelf SCRUM gebruikt voor projecten, heb ik ervoor gekozen om ook SCRUM te gebruiken. Hierdoor hoeft het bedrijf zich niet aan te passen en kan ik de theorie die mij geleerd is tijdens de studie in de praktijk brengen. Het is voor mij handig om meer ervaring met SCRUM op te doen, omdat het een ontwikkelingsmethodiek is die steeds populairder wordt. Daarnaast is het prettig om de voortgang dagelijks te bewaken en hierdoor op tijd bij te kunnen sturen bij veranderingen en problemen. Hoe de methode toegepast is, is te lezen in het plan van aanpak(bijlage I).

Elke sprint moest er getest worden, het rapporteren van de tests werd gedaan door het bedrijfstemplate hiervoor in te vullen op Google spreadsheets. Naast unit testing, is er exploratory testing gedaan.

4.2 Risicofactoren

Om te voorkomen dat de voortgang van het project geblokkeerd zou worden, is van te voren geanalyseerd wat de grootste risico's vormen en is een oplossing geformuleerd om deze risico's te vermijden.

In dit project moest ik werken met een voor mij onbekende programmeertaal. Daarnaast werd er gebruik gemaakt van een relatief nieuw framework en een aantal technieken waarmee ik geen ervaring had. Daarom is er tijd genomen om eerst kennis op te doen over de taal, het framework en andere technieken die gebruikt moesten worden.

4.3 Planning

De lengte van de sprints was voor dit project vastgesteld op 2 weken, dit kwam overeen met de sprints van andere projecten die op het moment in het bedrijf liepen en was daarom handig voor de product owner/SCRUM Master. De afstudeer stage had een duur van 17 weken, dus totaal waren er 8 sprints met 1 week uitlooptijd gepland. Eerst zijn er werkzaamheden uitgevoerd om voor te bereiden op het ontwikkeltraject. In de rest van de sprints werd ontworpen, ontwikkeld en getest. De eerste 3 sprints zijn gebruikt voor de Android applicatie, de overige sprints voor de webapplicatie. Zie bijlage I voor de complete planning.

4.4 Initiële backlog

Hieronder is de initiële backlog te zien. De user stories zijn met de product owner opgesteld en geprioriteerd met de MoSCoW-methode[1]. Voor het begin van elke sprint wordt er een sprint planning gehouden. Hierin wordt bepaald welke user stories er opgepakt worden en kunnen er nieuwe user stories bijkomen of aangepast worden. Ook worden de stories in taken opgesplitst en geschat hoe lang de taken zullen duren. Dit is dus niet een definitieve lijst.

Android app				
ID	Als een ...	Wil ik ...	Zodat ...	Prio
A0	gebruiker	met mijn RFID tag in- en uitklokken	ik niet meer zelf handmatig bij hoef te houden hoeveel uur ik heb gewerkt	Must have
A1	manager	dat medewerkers voor een bepaald project in- en uitklokken	per project bijgehouden kan worden hoeveel uur er aan gewerkt is	Must have
A2	gebruiker	een indicatie zien dat ik kan in- en uitklokken	ik weet wanneer ik mijn tag kan scannen	Must have
A3	gebruiker	bij het inklokken zien hoeveel uur ik nog gemiddeld per dag moet werken	ik weet of ik nog op schema loop	Must have
A4	gebruiker	bij het uitklokken zien hoeveel uur ik gewerkt heb	ik weet of ik genoeg gewerkt heb	Must have
A5	gebruiker	zo min mogelijk handelingen doen	ik niet veel tijd kwijt raak aan het registreren van uren	Must have
A6	gebruiker	bij het in- en uitklokken een inspirerende citaat zien	ik geïnspireerd raak	Must have
A7	gebruiker	een vrij aanvraag kunnen doen of ziekmelden	dit bij de manager bekend wordt	Should have
Web app				
ID	Als een ...	Wil ik ...	Zodat ...	Prio
W0	manager	projecten kunnen aanmaken	ik deze kan koppelen aan werknemers	Must have
W1	manager	projecten kunnen wijzigen	de projecten up-to-date gehouden kunnen worden	Must have
W2	manager	projecten kunnen verwijderen	geannuleerde projecten niet meer te zien zijn	Must have
W3	manager	werknemers kunnen aanmaken	ik deze kan koppelen aan projecten	Must have
W4	manager	werknemers kunnen wijzigen	de werknemers up-to-date gehouden kunnen worden	Must have
W5	manager	werknemers kunnen verwijderen	ik geen data meer zie van oud werknemers	Must have
W6	manager	werknemers aan projecten kunnen koppelen	ik kan zien hoeveel uur er is gewerkt per werknemer per project	Must have
W7	manager	de gewerkte uren per werknemer per project in een grafiek zien	ik hier overzicht over heb	Must have
W8	gebruiker	een overzicht van mijn gewerkte uren kunnen zien	ik dit kan controleren	Must have
W9	gebruiker	met google kunnen inloggen	ik mijn bedrijfsemail kan gebruiken	Must have
W10	manager	aan het eind van elke maand een mail ontvangen met de gewerkte uren per werknemer	ik dit naar de accountant kan sturen	Should have

Figuur 2 Initiële backlog

[1] https://nl.wikipedia.org/wiki/MoSCoW-methode

5 Voorbereidende werkzaamheden

Voor het beginnen van het ontwikkeltraject zijn een aantal voorbereidende werkzaamheden uitgevoerd. Het doel hiervan was om de benodigde kennis op te doen en de ontwikkelomgeving op te zetten. De volgende werkzaamheden zijn uitgevoerd:

- Tutorials volgen om kennis op te doen over JavaScript.
- De documentatie van React Native doornemen.
- De documentatie van GraphQL/Relay doornemen.
- Kennis opdoen over het Gitflow model.
- Het opzetten van de ontwikkelomgeving.
- Het in kaart brengen van de huidige situatie.

5.1 Kennis opdoen over JavaScript

Omdat ik nog nooit eerder met JavaScript had gewerkt heb ik eerst de tutorial op CodeCadamy[2] gevolgd. In de tutorial wordt de basis van JavaScript uitgelegd en leer je het tegelijkertijd door te coderen in de live editor waarbij je het resultaat ook gelijk kunt zien. Tijdens het volgen heb ik de verschillen met talen die ik ken en eigenaardigheden genoteerd zodat ik er later indien nodig naar kon refereren. Een voorbeeld van een eigenaardigheid is dat 1 == '1' true is. Dit komt doordat alleen de waarde wordt vergeleken met een dubbel gelijkteken. Om ook het type te vergelijken moet een drievoudig gelijkteken gebruikt worden(1 === '1' is false). Om in de toekomst problemen met vergelijkingen te voorkomen is kan dus beter een drievoudig gelijkteken gebruikt worden. Ik heb gekozen CodeCadamy voor de basis tutorials te gebruiken omdat ik de site eerder voor andere talen had gebruikt en het zeer gebruiksvriendelijk is. Na het leren van de basis heb ik gekeken naar de wat geavanceerdere onderwerpen die ik dacht nodig te hebben, zoals asynchrone acties(voor het ophalen van data) en het afhandelen van touchevents.

Om de code overzichtelijk te houden voor mijzelf en toekomstige ontwikkelaars die mijn code in moeten duiken, heb ik de JavaScript styling guidelines van Google[3] en Airbnb[4] doorgenomen. Bij het doornemen van de guidelines heb ik de verschillen tussen de twee genoteerd. Een van de dingen die mij opviel was dat de één(Airbnb) wel classes en bijvoorbeeld het keyword extends voor overving gebruikt en de ander(Google) niet. Ik was nieuwsgierig waarom dat soort verschillen er waren. Het blijkt dat het komt omdat browsers verschillende versies van ECMAScript(ES) ondersteunen en Google nog een oudere versie van ES, waar bepaalde features niet in zitten, gebruikt om zo veel mogelijk browsers te ondersteunen. Mijn persoonlijke voorkeur gaat uit naar de nieuwere style die Airbnb gebruikt, omdat het meer overeenkomt met de talen die ik ken. Er moest nog wel gekeken worden of React Native(zie paragraaf 5.2) de nieuwere versies van ES ondersteund.

Verder heeft het bedrijf nog eigen coding guidelines. Deze komen grotendeels overeen met de Clean code filosofie waar ik in de gaming minor van de opleiding al kennis over opgedaan had. Voorbeelden hiervan zijn: duidelijke variabele en methode namen en nested if statements opsplitsen in meerdere methodes.

[2] https://www.codecademy.com/
[3] https://google.github.io/styleguide/javascriptguide.xml
[4] https://github.com/airbnb/javascript

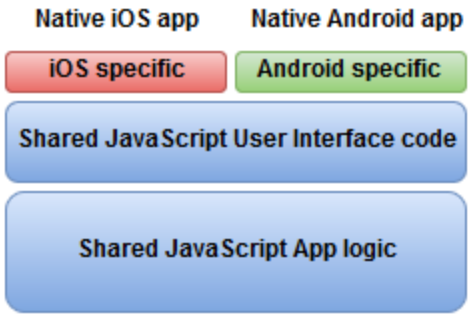
Het resultaat van het volgen van de tutorials en het doornemen van de styling/coding guidelines was dat ik een goede basis had om te beginnen met het ontwikkelen met JavaScript.

5.2 React Native

In deze paragraaf wordt beschreven wat React Native is, waarom hiervoor gekozen is, hoe het werkt en wat de voor- en nadelen zijn.

Wat is React Native?

React Native is een open-source framework, ontwikkeld door Facebook, om native applicaties voor iOS en Android te ontwikkelen met behulp van JavaScript en React(een JavaScript library voor web om UI's te maken).[5] Het framework is gebouwd met wat Facebook noemt “learn once, write anywhere” in gedachte. Dit betekent dat, hoewel het mogelijk is om een volledig cross-platform(iOS en Android) applicatie te ontwikkelen door alleen JavaScript en cross-platform componenten te gebruiken, dit niet de focus was bij het maken van het framework en meer gericht is op dat developers maar één taal/framework hoeven te leren en daarmee voor meerdere platforms kunnen ontwikkelen. Om de look en feel van een platform te behouden zal er voor de User Interface(UI) het een en ander voor de platformen specifiek gemaakt moeten worden. Zie het plaatje hieronder voor een globaal overzicht hoe de code verdeeld is.



Figuur 3 React Native code sharing

Waarom React Native?

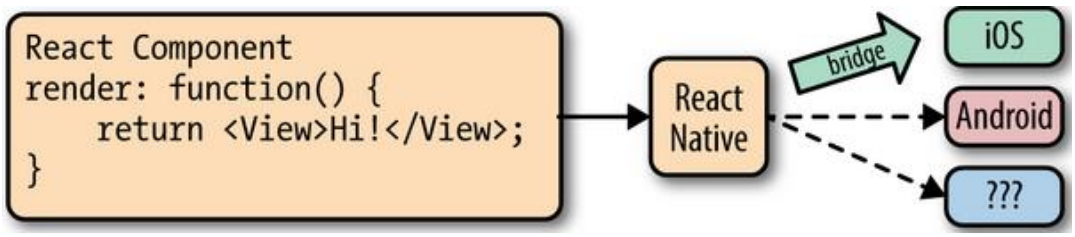
Een van de eisen aan het project was dat React-Native gebruikt moet worden. De reden hiervoor is dat het in tegenstelling tot veel alternatieven(bijvoorbeeld Cordova/PhoneGap) geen webviews gebruikt, maar native UI componenten, hierdoor kunnen native applicaties gemaakt worden met de look en feel van het betreffende platform. Er zijn echter alternatieven die eveneens geen webviews gebruiken zoals Xamarin, Appcelerator en NativeScript. Ik vroeg me af waarom de keus is gemaakt om React Native te gebruiken in plaats van een van deze alternatieven. De opdrachtgever wilde graag gebruik maken van een open-source oplossing, hierdoor vallen Xamarin en Appcelerator af. NativeScript en React Native zijn beide erg nieuw(maart 2015), als er gekeken wordt naar de statistieken op GitHub(waar deze twee open source projecten gehost worden) heeft React Native een grotere community. Zo heeft React Native op het moment van schrijven bijvoorbeeld 624[6] contributors en NativeScript 44[7], hier is de opdrachtgever erg enthousiast over.

[5] https://facebook.github.io/react-native/
[6] https://github.com/facebook/react-native

Hoe werkt het?

React Native gebruikt Objective-C APIs om iOS componenten te tekenen en Java APIs om Android componenten te tekenen. Dit is mogelijk door de React Native bridge. De React Native componenten worden door de bridge vertaald naar platform specifieke componenten. Op dit moment ondersteund React Native alleen iOS en Android. Om bijvoorbeeld Windows 10 Mobile te ondersteunen zal er een bridge ontwikkeld moeten worden die React Native componenten naar Windows 10 Mobile componenten vertaald. Facebook heeft aangegeven dat er op dit moment geen plannen voor zijn, dus zal dit door de community gedaan moeten worden.

Alle JavaScript code wordt op een aparte thread uitgevoerd, hierdoor wordt de main UI thread niet geblokkeerd door JavaScript. JavaScriptCore is de engine die gebruikt wordt om de JavaScript code uit te voeren. De communicatie tussen de JavaScript en Native code verloopt via de React Native bridge.



Figuur 5 React Native bridge[8]

De syntax die gebruikt wordt om views op te bouwen in React Native is JSX. JSX lijkt erg op XML en zorgt ervoor dat de code makkelijker te lezen is. [9] In het figuur hierboven wordt een React Native component met een <View> element weergegeven. Het <View> element wordt voor iOS vertaald naar een UIView en voor Android naar een android.view. [10]

Elk component heeft een render methode. Componenten kunnen ook state en props hebben. State en props kunnen gezien worden als model data voor het component[11]. State wordt gebruikt voor attributen die binnen het component kunnen veranderen, en props voor attributen die niet binnen het component kunnen veranderen. Bij een verandering van een state of prop wordt automatisch berekend wat er opnieuw getekend moet worden en vervolgens zo min mogelijk opnieuw getekend.

[7] https://github.com/NativeScript/NativeScript/
[8] Eisenman, B. (2015). *Learning React Native.* O'Reilly.
[9] https://facebook.github.io/react/docs/jsx-in-depth.html
[10] https://facebook.github.io/react-native/docs/view.html
[11] https://facebook.github.io/react/docs/thinking-in-react.html

Styling

Het stylen in React Native kan gedaan worden met behulp van JavaScript door op CSS gebaseerde stylesheets te definiëren. De lay-out kan geregeld worden met behulp van Flexbox. Dit ziet er als volgt uit:

```
var styles = StyleSheet.create({
    container: {
        flex: 1,
        flexDirection: 'column'
    },
    halfHeight: {
        flex: .5,
        backgroundColor: '#FF3366'
    },
    quarterHeight: {
        flex: .25,
        backgroundColor: '#000'
    }
});
```

Figuur 6 Javascript Stylesheet met Flexbox

Vervolgens kunnen de gedefinieerde styles gebruikt worden:

```
<View style={styles.container}>
    <View style={styles.halfHeight} />
    <View style={styles.quarterHeight} />
    <View style={[styles.quarterHeight, {backgroundColor: '#CCC'}]} />
</View>
```

Figuur 7 Styles toegewezen aan views

Figuur 8 Resultaat van de toegewezen styles

De drie views zitten in een container view(in dit geval het hele scherm) met een flex van 1 en een flexDirection column. Dit betekent dat de views van boven naar beneden neergezet worden. De eerste view in de container neemt de helft van de container in beslag omdat het een flex van .5 heeft. De andere 2 views nemen allebei een kwart van de container(flex .25). Bij de laatste view in de container wordt ook nog een backgroundColor toegewezen om te laten zien dat er niet per se gebruik gemaakt hoeft te worden van een stylesheet en dat een element meerdere styles kan hebben.

Packaging

Om te voorkomen dat er onnodige Android specifieke files worden gebundeld voor iOS en vice versa en alle gebruikte modules te bundelen is er een React Native packager. De React Native packager kijkt naar voor welk platform de code is. Dit kan bepaald worden door de naam van de files aan te vullen met .android of .ios(bijvoorbeeld Home.android.js en Home.ios.js). De file met android in de naam zal alleen voor Android gebruikt worden en de ios file voor iOS. Files zonder extra platform extensie worden voor beide gebruikt. De extra platform extensie hoeft dus alleen gebruikt te worden als er sprake is van platform specifieke componenten of modules.

Voor- en nadelen

Het ontwikkelen met React Native heeft voor- en nadelen, deze worden hier opgesomd.

Voordelen:

- Ontwikkelaars hoeven maar één taal/framework te leren om voor iOS/Android te ontwikkelen.
- Als je ervaring hebt met React voor web, kun je door de vele overeenkomsten nu ook native apps voor iOS/Android ontwikkelen.
- Tijdens het ontwikkelen kan Live Reloading gebruikt worden, waardoor elke change die je in de JavaScript code doet met een refresh meteen te zien is op het testdevice of de emulator.
- Vergeleken met veel alternatieven worden native UI elementen gebruikt en geen webviews. Hierdoor kunnen native apps gemaakt worden met de door de gebruiker verwachte look en feel.
- Met Flexbox kan de lay-out handig ingedeeld worden voor verschillende schermgroottes.

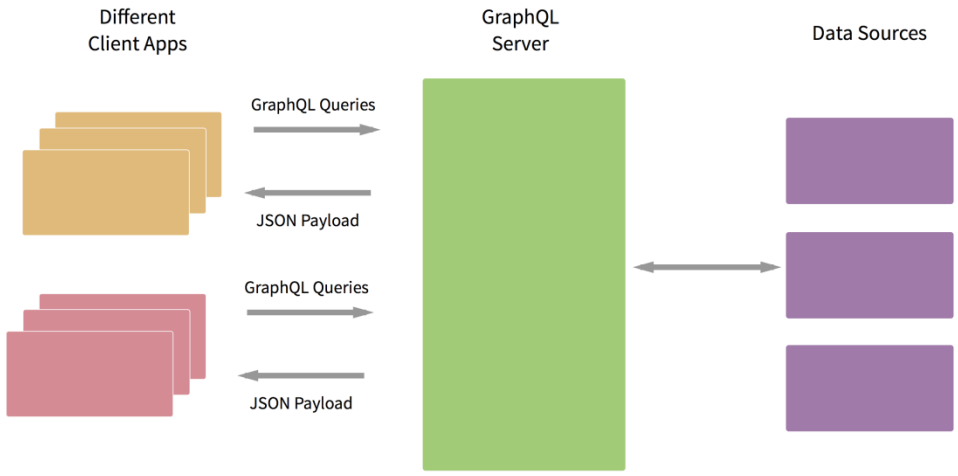
Nadelen:

- Het is relatief nieuw(ondersteuning voor iOS was in maart 2015 uitgebracht, voor Android in september 2015). Het gevolg hiervan is dat de documentatie niet compleet is(voor componenten zijn er bijvoorbeeld geen screenshots te vinden van hoe de componenten eruit zien) en maar een beperkt aantal componenten out-of-the-box beschikbaar is.
- Om de look en feel van de verschillende platformen te behouden zullen vaak aparte componenten gebruikt moeten worden per platform.

5.3 GraphQL/Relay

Relay is een framework door Facebook gemaakt die functionaliteit biedt om data van een GraphQL server op te halen. [12] Relay zorgt ervoor dat de queries om data op te halen in efficiënte groepen samengesteld worden. Doordat React componenten hun eigen data afhankelijkheden moeten specificeren krijgen de componenten precies de data waar om gevraagd wordt en kunnen de componenten geüpdatet worden als de data veranderd. Het specificeren van de data die nodig is gebeurd met GraphQL.

GraphQL is de query taal die Relay gebruikt om de data op te halen. Om dit te doen moet de server geconfigureerd zijn om GraphQL queries te accepteren. Dit kan worden gedaan door op de server een GraphQL schema te specificeren. Relay kan met een GraphQL server communiceren door een network layer. In het figuur hieronder is het verloop van een GraphQL request te zien.



Figuur 9 Overzicht GraphQL query request[13]

GraphQL wordt vergeleken met REST en heeft het voordeel dat het alle benodigde data in 1 request op kan halen(zou met REST ook kunnen maar dan zou de server aangepast moeten worden en er een extra endpoint komen). Bij GraphQL specificeert de client de benodigde data in plaats van erop vertrouwen dat de server de juiste data geeft. [14]

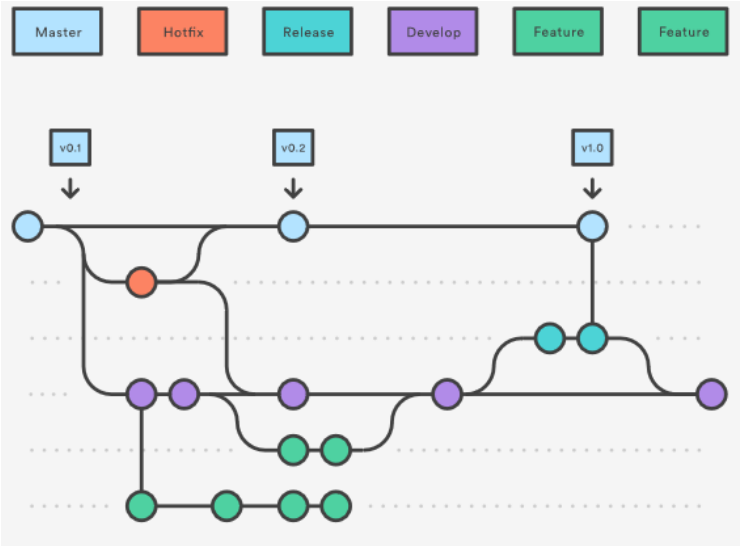
[12] https://facebook.github.io/relay/
[13] https://learngraphql.com/basics/introduction
[14] https://facebook.github.io/relay/docs/thinking-in-graphql.html

5.4 Gitflow

Gitflow is een branching model voor het werken met Git. Het bedrijf hanteert dit model en hier moest dus ook mee gewerkt worden. De volgende branches zijn van toepassing op het model:

- Master: de master wordt gebruikt voor officiële releases.
- Develop: develop wordt gebruikt als een verzamelingspunt voor features.
- Feature: voor elke nieuwe feature wordt een feature branch gemaakt, als een feature af is wordt deze naar develop gemerged.
- Release: als de develop branch genoeg features heeft voor een release, wordt een release branch aangemaakt. Hier mogen geen nieuwe features aan toegevoegd worden, alleen release georiënteerde bezigheden. Als een release klaar is wordt deze gemerged naar master en develop.
- Hotfix: hotfix branches worden gemaakt om bugs in releases die in productie genomen zijn te verhelpen. Dit zijn de enige branches die direct van master af komen. Als een hotfix af is wordt deze naar master en develop gemerged.

In het figuur hieronder is een voorbeeld van dit model weergegeven:



Figuur 10 Gitflow branching model[15]

De repository is op aangeven van het bedrijf aangemaakt op GitLab, hier kunnen privé repositories gratis gehost worden. [16]

5.5 Opzetten van de ontwikkelomgeving

Nadat de tutorials en documentatie doorgenomen was is begonnen aan het opzetten van de ontwikkelomgeving. Als eerst moest er een keus gemaakt worden op welk platform de ontwikkelomgeving opgezet wordt. De prioriteit van Facebook is om OS X als ontwikkelomgeving te ondersteunen, ondersteuning voor Linux en Windows is de verantwoordelijkheid van de community. De setup guide van Facebook richt zich vooral op OS X. Hierdoor leek mij het best om de ontwikkelomgeving op een OS X op te zetten. Macs waren echter beperkt beschikbaar binnen het bedrijf, dus om continu te kunnen ontwikkelen moest gekozen worden tussen Windows of Linux. Omdat

[15] https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
[16] https://about.gitlab.com/

Facebook op de documentatie aangeeft dat op Windows de React Native packager niet automatisch opstart is gekozen om de ontwikkelomgeving voor Linux op te zetten.
Nadat het operating systeem geïnstalleerd was, is de setup guide op de Facebook documentatie voor React Native gevolgd. Bij het volgen van de guide zijn veel fouten opgetreden. Dit kwam door een aantal missende afhankelijkheden die niet in de documentatie genoteerd zijn en eerst geïnstalleerd moesten worden. Om te voorkomen dat deze fouten in de toekomst weer optreden zijn de stappen die gezet zijn en alle benodigde commands genoteerd(Zie bijlage A). Hieronder volgt een lijst met de benodigde onderdelen:

- Git
- Node.js 4.0 of hoger
- JDK 7.0 of hoger
- Android SDK
- Een Android emulator
- React developer tools voor Chrome of Firefox
- React Native

Naast de benodigde onderdelen staan er ook een aantal handige optionele onderdelen op de documentatie die ook gebruikt worden. Deze onderdelen zijn Watchman, Flow en en Nuclide. Watchman kijkt of er files gewijzigd zijn en of de files herbouwd moeten worden, als dat het geval is herbouwd het de files ook. Flow is een typechecker die je helpt vroeg type errors te vinden in JavaScript code en runtime errors te voorkomen. Om flow te gebruiken moet er /* @flow */ bovenaan de JavaScript file staan.

Nuclide[17] is een IDE speciaal gemaakt voor React Native gebaseerd op Atom(een open-source tekst editor). Dit IDE is gebruikt omdat er verder geen alternatieven zijn voor React Native(behalve tekst editors). Na het opzetten van het IDE bleek het erg traag te werken. Het gebruikte 100+% van de CPU kracht. Na wat uitzoekwerk bleek het te komen door een aantal plug-ins die automatisch geïnstalleerd worden door de nuclide-installer. Omdat er op het internet niet te vinden was door welke plug-in het precies kwam zijn deze 1 voor 1 verwijderd om erachter te komen. Toen vastgesteld was welke plug-in het was is het IDE nogmaals geïnstalleerd en is alleen die plug-in verwijderd.

Unit testing wordt gedaan met behulp van Jest. De tests worden in de terminal uitgevoerd door de npm test command uit te voeren. Alle tests moeten in een folder gezet worden met de naam __tests__ en package.json(hierin staat alle metadata van het project, bijvoorbeeld de afhankelijkheden) van het project moet aangevuld worden met:

```
"scripts": {
  "test": "jest"
}
```

Waardes kunnen vergeleken worden met `expect(value).toBe(othervalue).`

Nadat alles opgezet was kon het project opgezet worden met react-native init <ProjectName>. Om het project vervolgens te runnen naar de folder navigeren van het project en react-native run-android uitvoeren. Als er een emulator of device is aangesloten wordt de applicatie erop geïnstalleerd en gestart.

[17] http://nuclide.io/

De ontwikkelomgeving is twee keer opgezet. Een keer op een werkstation en een keer op mijn persoonlijke laptop voor als het werkstation niet beschikbaar is.

5.6 Continuous Integration

Om bij elke push naar de remote repository automatisch de unit testen uit te voeren is continuous integration opgezet. Dit zorgt ervoor dat je kunt zien of alle tests nog slagen bij nieuwe features/aanpassingen aan de code en dus of hele applicatie nog goed werkt(Quality assurance). Deze tests worden in een geïsoleerde omgeving uitgevoerd door een runner. In deze geïsoleerde omgeving moeten wel dezelfde de dependencies aanwezig zijn, zodat er geen problemen met dependencies komen bij het testen.

Omdat er gebruik gemaakt wordt van GitLab moet de runner hierop geregistreerd worden, zodat bij een push de runner de tests kan uitvoeren. De runner hoeft hiervoor maar een command uit te voeren: npm test, hiermee worden alle tests automatisch uitgevoerd in de __tests__ folder van het project. De commands die de runner uitvoert moeten in een speciale file staan in de root van de projectfolder met de naam .gitlab-ci.yml. Je kunt de runner meer laten doen dan alleen het automatiseren van unit tests, maar voor dit project is dat het enige wat de runner hoeft te doen.
Nadat de continuous integration was opgezet is deze getest door een simpele unit test in de __tests__ folder te zetten en deze met het project te pushen naar de repository op GitLab. Op Gitlab kan dan bekeken worden wat de runner aan het doen is. Een build kan vier statussen hebben “pending”, “success“, “failed“ of “skipped“. Als de build pending is kan bekeken worden waar de runner mee bezig is. Als een van de tests gefaald is zal de status van de build op failed komen te staan en als alle tests geslaagd zijn op success. De skipped status zal alleen verschijnen als er specifiek wordt aangegeven bij een build om de runner niet uit te voeren.

5.7 Resultaat

Het resultaat van de voorbereidende werkzaamheden was dat er kennis is opgedaan om te kunnen beginnen met het ontwikkelen met React Native, een opgezette ontwikkelomgeving en een eerste uitgevoerde React Native applicatie. De feedback op het uitgevoerde proces was erg positief. Na de feedback werd verteld door de stakeholders dat zij een ander project hadden dat meer waarde voor hun had en dus urgenter was. Er werd gevraagd of ik kon wisselen van project, voor het project moesten dezelfde tools/technieken gebruikt worden. Omdat de voorbereidende werkzaamheden voor het nieuwe project niet zouden veranderen, was de impact van het veranderen van project erg laag. Ik vond het hierom niet erg om te wisselen van project, mits de school het toe zou laten. Hierop is contact opgenomen met de begeleider vanuit de school. Vervolgens is een nieuw afstudeerplan opgestuurd. Na het akkoord vanuit de school is het plan van aanpak aangepast op het nieuwe project.

Hierna kon ik beginnen met de sprints.

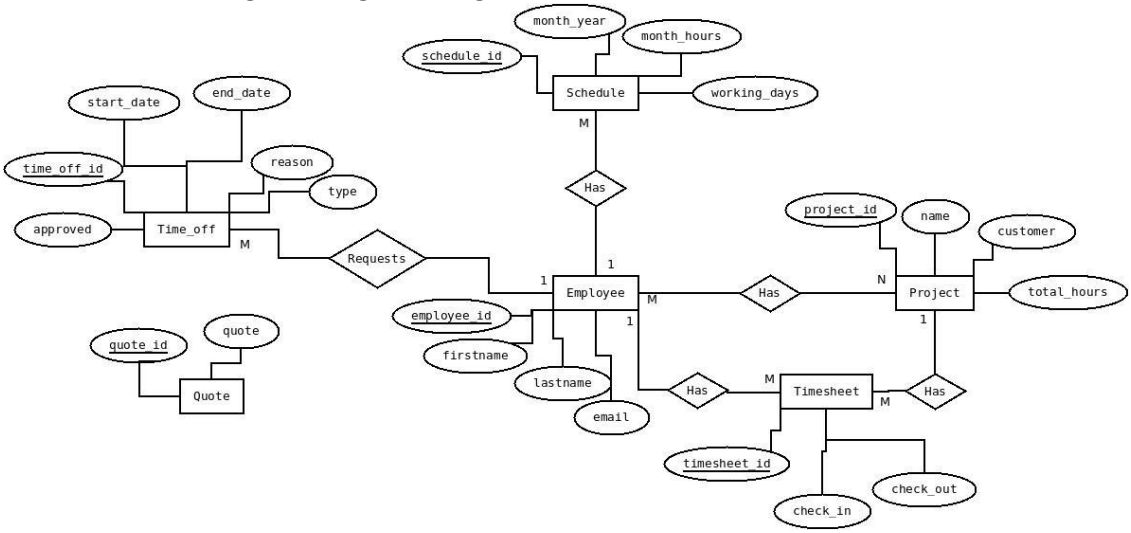
6 Sprint 1 Prototype van de applicatie

De bedoeling van deze sprint was om een prototype van de applicatie te maken. In deze sprint zijn aan de volgende user stories gewerkt:

- Als manager wil ik dat medewerkers voor een bepaald project in- en uitklokken, zodat per project bijgehouden kan worden hoeveel uur er aan gewerkt is.
- Als gebruiker wil ik een indicatie zien dat ik kan in- en uitklokken, zodat ik weet wanneer ik mijn tag kan scannen.
- Als gebruiker wil ik bij het inklokken zien hoeveel uur ik nog gemiddeld per dag moet werken, ik weet of ik nog op schema loop.
- Als gebruiker wil ik bij het in- en uitklokken een inspirerende citaat zien, zodat ik geïnspireerd raak.

6.1 Entity Relationship Diagram(ERD)

Voordat ik ben begonnen met het ontwikkelen van de applicatie heb ik de database ontworpen en opgezet zodat ik data uit de database kan gebruiken bij het ontwikkelen van de applicatie. Bij het ontwerpen van het ERD is onder andere rekening gehouden met het huidige systeem en de user stories uit de initiële backlog. Het volgende diagram is het resultaat:



Figuur 11 Entity Relationship Diagram

Hieronder een korte uitleg van het diagram:

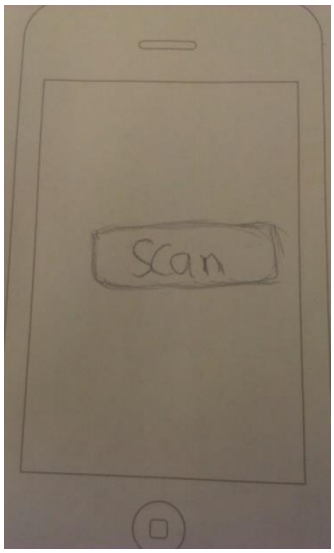
- Employee: hier worden gegevens opgeslagen van de werknemer op basis van een tagId. Het tagId is het ID van de tag die de werknemer bij zich draagt.
- Time_off: hier worden de verzoeken voor het vrij vragen van werknemers in opgeslagen. Bij het vrij vragen moet een startdatum, einddatum en een reden ingevoerd worden. De administrator kan het verzoek vervolgens goed of afkeuren.
- Schedule: hier wordt per maand per werknemer opgeslagen hoeveel uren er gewerkt moet worden en hoeveel werkdagen er gemiddeld in een week gewerkt wordt. Dit is onder andere nodig om te berekenen hoeveel uur er nog te gemiddeld te gaan is per dag en per maand.
- Project: hier worden de projecten opgeslagen met naam, klant en het totaal aantal uur.
- Timesheet: hier wordt het in- en uitklokken van werknemers voor een bepaald project opgeslagen.
- Quote: hierin staan de citaten opgeslagen die gezien moeten worden bij het inklokken.

6.2 Opzetten database

Voor het opzetten van de database was ik vrij om het database management systeem te kiezen, mits het open-source was. Mijn keus is uitgegaan naar MySQL, omdat ik hier al ervaring mee had. Nadat de database was opgezet moest er een GraphQL server met een GraphQL schema opgezet worden die met de database kan verbinden, zodat later data via de GraphQL server opgehaald kon worden. Om dit te doen heb ik eerst het voorbeeld van de documentatie gevolgd. Dit voorbeeld verbind echter niet met een externe database en gebruikt een in-memory database. Er staat ook niet beschreven hoe je met een externe database kan verbinden. Ook was er weinig over te vinden op het internet. Uiteindelijk had ik dus wel een opgezette GraphQL server, maar geen manier om de server met de MySQL database te verbinden. Omdat ik hier veel tijd aan heb besteedt, heb ik besloten om tijdelijk een in-memory database te gebruiken en te beginnen met het implementeren van de UI.

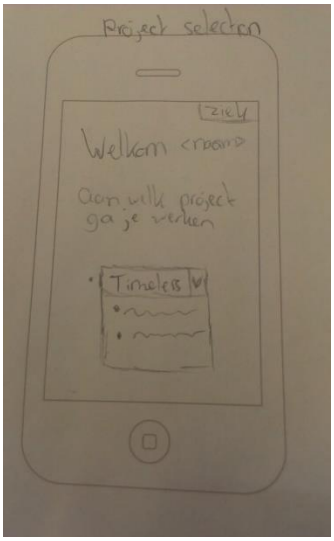
6.3 Implementeren van de UI

De UI is gebaseerd op wireframes, de gebruikte wireframes voor het implementeren van de UI worden hier getoond met de beschrijving:



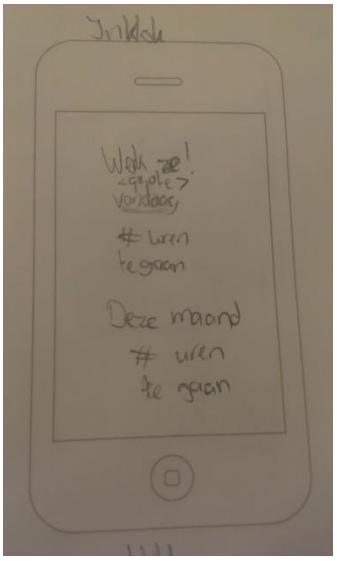
Figuur 12 Scan scherm WF

Op het eerste scherm moet een indicatie komen dat er gescand wordt. De bedoeling is dat op dit scherm een RFID-tag gescand kan worden, als er een tag gescand is wordt er naar het volgende scherm genavigeerd.



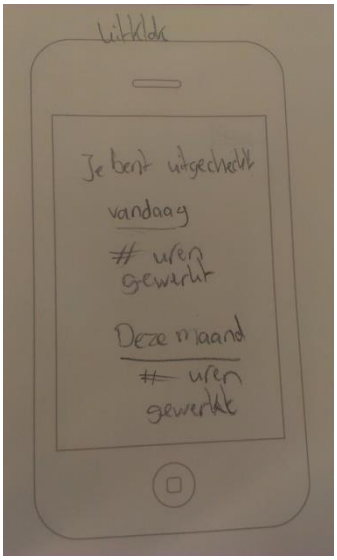
Figuur 13 Project selecteer scherm

Op dit scherm moet een welkomstbericht komen met de naam van de persoon die een tag gescand heeft. Bovenaan het scherm is een knop om vrij te vragen of ziek te melden. Onder het welkomstbericht is een lijst te zien met de projecten waarvoor ingeklokt kan worden. Als een project uit de lijst gekozen wordt, wordt naar het inklok scherm genavigeerd. Als de vrij aanvraag/ziekmeld knop gebruikt wordt, wordt er naar het vrij aanvraag/ziekmeld scherm genavigeerd.



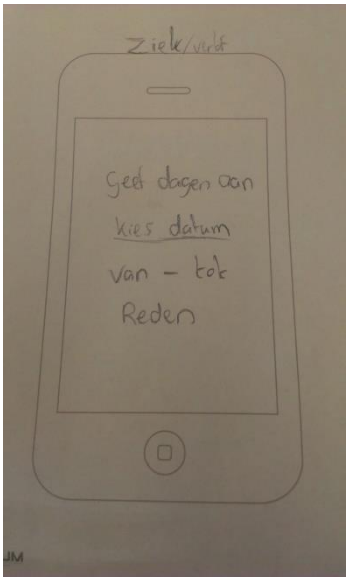
Figuur 14 Inklok scherm WF

Op dit scherm wordt een indicatie dat er is ingeklokt, hoeveel uur er gemiddeld per dag te gaan is en hoeveel uur er nog in de maand te gaan is getoond.



Figuur 15 Uitklok scherm WF

Op dit scherm wordt een indicatie dat er is uitgeklokt, hoeveel uur er gewerkt is op de dag en hoeveel uur gewerkt is in de maand getoond.



Figuur 16 Vrij vraag scherm WF

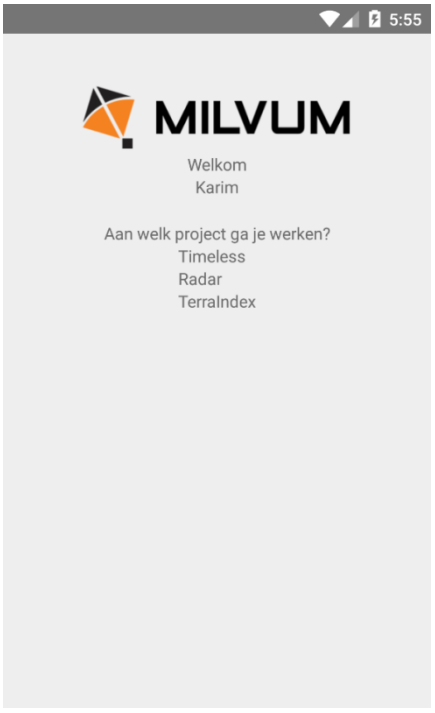
Op dit scherm kan de datum ingevuld worden van wanneer tot wanneer er vrij gevraagd wordt, met daarbij de reden.

In React Native hebben alle componenten een render methode, hierin kan gespecificeerd worden wat er op het scherm getekend moet worden. Dit is dus de plek waar alle code voor de UI hoort. De UI wordt gespecificeerd met JSX syntax(Zie figuur 7).

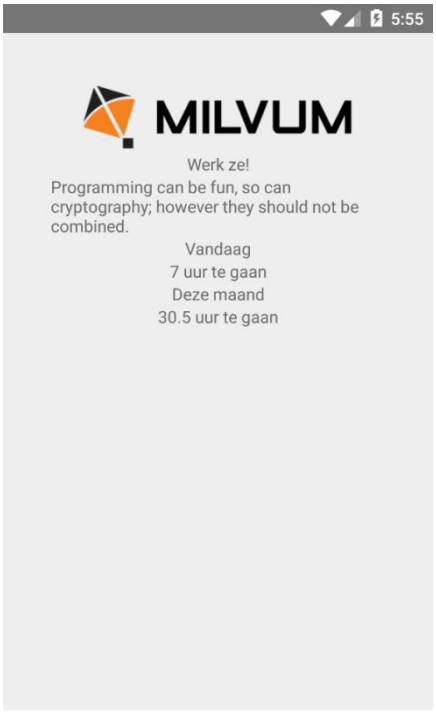
Om van schermen te wisselen moest er een Navigator component geïmplementeerd worden, de documentatie over hoe dit te doen is erg onduidelijk. Uiteindelijk is het gelukt door hulp van iemand op Reactiflux, een chatroom voor alles wat met React te maken heeft. Hier kunnen dus ook vragen gesteld worden over React Native in een aparte channel. Er moet een initialRoute prop aangegeven worden bij het aanmaken van een Navigator. Hier kan aangegeven worden wat het eerste scherm is dat getoond moet worden. Vervolgens kan naar een volgend scherm genavigeerd worden door de navigator.push(<ComponentNaam>) methode te gebruiken en terug genavigeerd worden door de navigator.pop() methode te gebruiken.

Omdat het design nog niet ontworpen was heb ik de schermen minimalistisch in elkaar gezet, hier volgt het resultaat:

Figuur 17 Scan scherm v1



Figuur 18 Project selecteer scherm v1



Figuur 19 In/Uitklok scherm v1

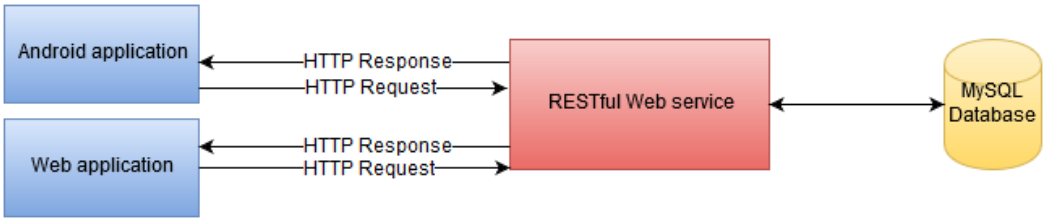
Bij deze schermen mist het vrij aanvraag/ziekmeld scherm. Dit komt doordat het mij beter leek om het vrij vragen/ziekmelden op de webapplicatie te doen. Het device waarop in- en uitgeklokt moet worden staat namelijk op het kantoor. Als iemand zich ziek wilt melden is de persoon meestal niet op het kantoor, maar op de plaats waar hij/zij woonachtig is. Ik heb geadviseerd om dit in plaats van in de Android applicatie, het in de webapplicatie te plaatsen. Zo blijft de functie van de applicatie puur voor het in- en uitklokken en kan de rest op de webapplicatie gedaan worden. De stakeholders vonden dit een goed idee en het scherm hoefde dus niet geïmplementeerd te worden.

6.4 Data ophalen

Toen de schermen geïmplementeerd waren, werd het tijd om de mock data te vervangen met data uit de database. Het ophalen van data wilde niet lukken. Naast het probleem van het verbinden van de GraphQL server met de database kwam ik er ook achter dat Relay nog niet out-of-the-box ondersteund is in React Native en alleen gebruik kan worden door een downgrade van React Native en een aantal hacks uit te voeren. Tijdens het zoeken kwam ik erachter dat het React Native team toevallig een reddit AMA(Ask Me Anything) deed. Hier konden live vragen gesteld worden aan het React Native team. Omdat de opdracht was om Relay te gebruiken heb ik de vraag gesteld wanneer Relay out-of-the-box ondersteund wordt. Als antwoord kreeg ik te horen dat ze er wel mee bezig zijn, maar nog geen datum hebben.

Vervolgens heb ik gekeken naar de alternatieven van een GraphQL/Relay setup. GraphQL wordt door Facebook op de documentatie een direct alternatief voor REST genoemd. Hierna heb ik de opdrachtgever geadviseerd om geen GraphQL/Relay te gebruiken vanwege de hacks die uitgevoerd moeten worden om het werkend te krijgen en de geringe documentatie, maar een RESTful web service te gebruiken. De opdrachtgever was het eens dat het geen goed idee was om GraphQL/Relay te gebruiken door de hacks die uitgevoerd moeten worden en heeft mijn advies opgevolgd.

Vervolgens heb ik gezocht naar een snelle manier om een RESTful web service op te bouwen, omdat er veel tijd gegaan is naar het proberen van GraphQL/Relay werkend te krijgen. Ik ben erachter gekomen dat met Netbeans en Glassfish server heel makkelijk een RESTful web service(met de Java API for RESTful Web Services) gegenereerd kan worden uit een bestaande MySQL database. Nadat ik dit had gedaan heb ik een aantal aanpassingen gedaan om ervoor te zorgen dat de service aan de wensen van de stakeholders kan voldoen. Bij het testen van de web service kwam ik erachter dat de service alleen data in XML formaat ondersteunde, terwijl gespecificeerd was dat het ook JSON moest kunnen ondersteunen. Dit bleek te komen door een probleem in een van de modules die de GlassFish server gebruikte. Toen deze module aangepast was kon de service ook JSON formaat ondersteunen. De Android aplicatie en web applicatie kunnen beide gebruik maken van de web service om resources uit de database op te halen:



Figuur 20 Beide applicaties maken gebruik van de RESTful web service

6.5 Resultaat

Het resultaat van de sprint was een database, een prototype van de applicatie en een RESTful web service. Bij het tonen van het prototype bij de sprint review waren de reacties van de stakeholders erg positief. Een van de stakeholders dacht dat de applicatie bijna af was. Aan deze stakeholder heb ik uitgelegd dat de applicatie gebruik maakt van een in-memory database met testdata en dat de opgezette RESTful web service om de database te bevragen nog niet gebruikt werd. De product owner was erg tevreden over hoe ik omgegaan ben met het probleem dat GraphQL/Relay niet out-of-the-box werkt met React Native.

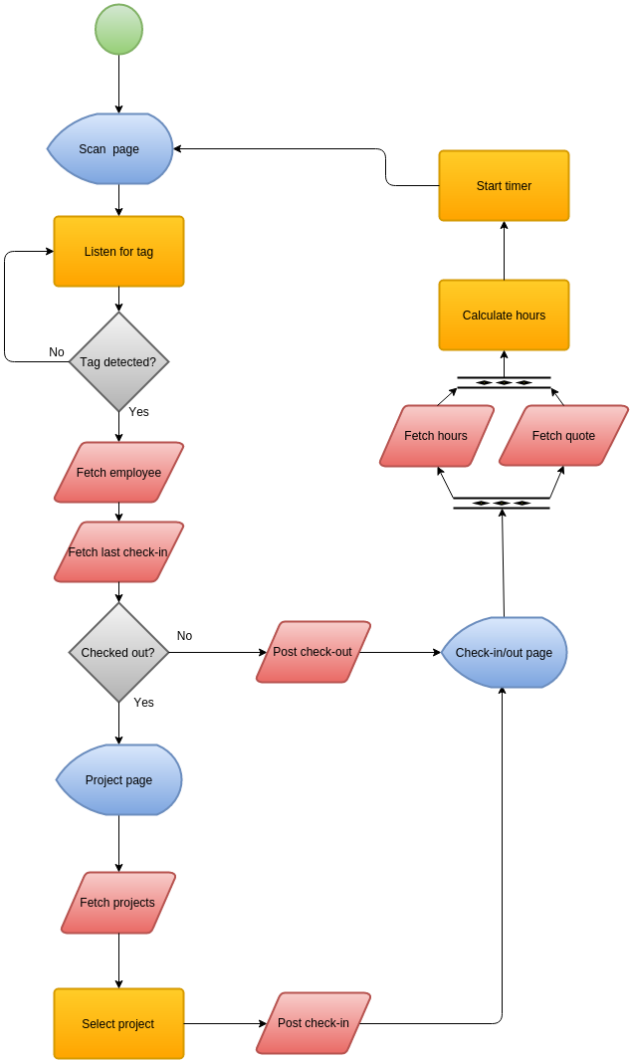
7 Sprint 2 Ontwerpen en bouwen van de mobiele applicatie

Het doel van deze sprint was om de applicatie te ontwerpen, een nieuw design te implementeren en de applicatie af te maken. In deze sprint zijn aan de volgende user stories gewerkt:

- Als gebruiker wil ik bij het uitklokken zien hoeveel uur ik gewerkt heb, zodat ik weet of ik genoeg gewerkt heb.
- Als gebruiker wil ik met mijn RFID tag in- en uitklokken, zodat ik niet meer zelf handmatig bij hoef te houden hoeveel uur ik heb gewerkt.
- Als gebruiker wil ik zo min mogelijk handelingen doen, zodat ik niet veel tijd kwijt raak aan het registreren van uren.

7.1 Flow diagram

Het flow diagram hieronder laat zien hoe de applicatie werkt(zie de volgende pagina voor uitleg):



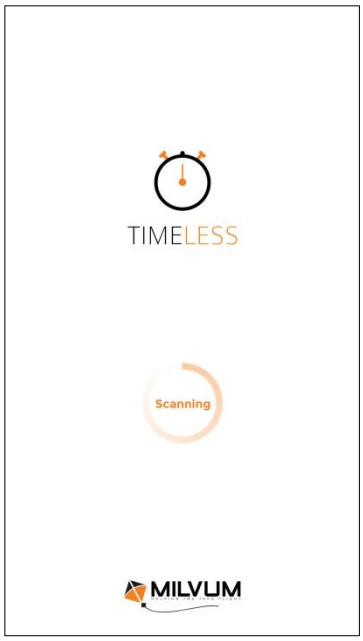
Figuur 21 Flow diagram mobile app

Als de applicatie opgestart wordt kom je op de Scan page terecht. Op de Scan page wordt er gekeken of er een tag gedetecteerd is, deze check blijft doorgaan tot er een tag gedetecteerd is. Als er een tag gedetecteerd is wordt de medewerker die bij de gedetecteerde tag hoort uit de database opgehaald. Vervolgens wordt de laatste keer dat de medewerker heeft ingeklokt uit de database gehaald(timesheet). Bij de opgehaalde timesheet wordt gekeken of er een waarde is ingevuld voor de uitkloktijd. Nu zijn er twee mogelijkheden:

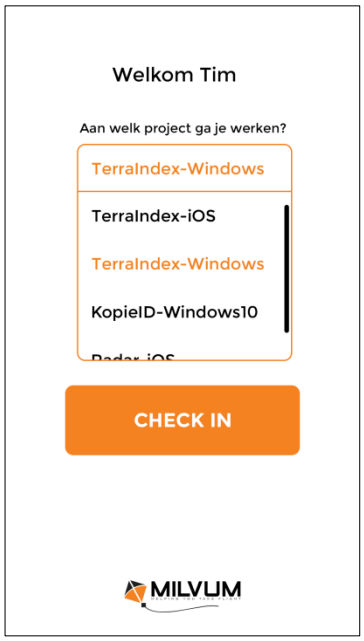
1. Als dit niet het geval is wordt de opgehaalde timesheet aangevuld met de huidige tijd als uitklok waarde en in de database geüpdatet. Vervolgens wordt er naar de Check-out page genavigeerd. Op deze page wordt een quote en de timesheets van de werknemer in een maand gewerkt opgehaald. Uit de opgehaalde timesheets wordt berekend hoeveel uur de werknemer op de dag heeft gewerkt en op de maand. Dit wordt op het scherm getoond. Zodra de berekening af is wordt een timer gestart die na afloop naar de Scan page navigeert.
2. Er wordt naar de Project page genavigeerd en de projecten waaraan de werknemer gekoppeld is worden opgehaald en in een lijst getoond, met bovenaan de lijst het project waar voor het laatst ingeklokt is. Vervolgens kan de werknemer selecteren voor welk project ingeklokt moet worden. Als voor het laatst ingeklokte project ingeklokt moet worden hoeft er niks geselecteerd worden, omdat deze standaard al geselecteerd staat. Na selectie wordt een nieuwe timesheet entry naar de database verzonden met de huidige tijd, het id van de werknemer en het id van het geselecteerde project. Vervolgens wordt naar de Check-in page genavigeerd, een quote en timesheets van de huidige maand opgehaald. Uit de timesheets wordt berekend hoeveel uur er per dag gemiddeld nog te gaan is in de maand. Zodra de berekening af is wordt een timer gestart die na afloop naar de Scan page navigeert.

7.2 Nieuw design

Voor het bepalen van het design van de applicatie is een design sessie met de designer van het bedrijf gehouden. Tijdens de sessie is gekeken wat wel en wat niet mogelijk is. Na de sessie is de designer aan het werk gegaan met de designs en heeft hij deze aan mij geleverd. Een onderdeel van het design was een drop down box om projecten te selecteren. Omdat facebook geen drop down component beschikbaar heeft gemaakt voor React Native heb ik gezocht naar een drop down component op React.parts. Hier kunnen componenten gevonden worden die door de community zijn gemaakt. Ik heb hier een drop down component kunnen vinden. Het installeren van een component gaat erg gemakkelijk met npm. Npm maakt het makkelijk voor JavaScript ontwikkelaars om code van elkaar te herbruiken of met elkaar te delen. [18] Het enige wat gedaan moest worden om het component te gebruiken was het npm install react-native-dropdown command uit te voeren in de terminal en vervolgens het component in de React Native code te importeren. Dit component werkte echter niet goed voor Android. Vervolgens heb ik dit gemeld aan de designer en is het design aangepast om een scrollable Listview met header te gebruiken in plaats van een drop down. Hier volgt het resultaat van het design:



Figuur 22 Scan scherm v2



Figuur 23 Project selecteer scherm v2

Figuur 24 Inklok scherm v2

Figuur 25 Uitklok scherm v2

Op het projecten selecteer scherm(Figuur 23) is het de bedoeling dat het project waarvoor het laatst ingeklokt is standaard in de header komt en geselecteerd is.

[18] https://docs.npmjs.com/getting-started/what-is-npm

7.3 Tag detectie

Om tags te kunnen scannen wordt er gebruik gemaakt van Near Field Communication(NFC). NFC is de techniek die bijvoorbeeld wordt gebruikt voor het contactloos betalen en het in en uitchecken met de OV-chipkaart. Om met een NFC device contact te maken is meestal een afstand van 4cm of minder nodig tussen het NFC device en de tag waarmee je contact wil maken met het device. [19] NFC wordt gebruikt om taken te automatiseren. Voor dit project is het de bedoeling dat medewerkers kunnen in- en uitklokken met hun tag.

De tagdetectie werkt als volgt: De native code kijkt of er een tag gedetecteerd is, De JavaScript code luistert naar het “tagId” event. Als er een tag gedetecteerd wordt, wordt het tagId uit de tag opgehaald en een “tagId” event naar de JavaScript code gestuurd. Vervolgens wordt het event ontvangen en kan het tagId in de JavaScript code gebruikt worden.

Omdat er nog geen standaard module in React Native beschikbaar is om de NFC sensor van een Android device te gebruiken, heb ik deze zelf in native Android moeten implementeren. Hiervoor heb ik de volgende stappen moeten nemen:

- In de AndroidManifest.xml file aangeven dat NFC gebruikt wordt met:
  ```
  <uses-permission android:name="android.permission.NFC" />
  ```
- Bij de activity in de AndroidManifest.xml file een intent filter aangeven, zodat deze op kan vangen wanneer een tag gedetecteerd wordt en het data type van de tag weet:
  ```
  <intent-filter>
      <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
      <category android:name="android.intent.category.DEFAULT"/>
      <data android:mimeType="text/plain"/>
  </intent-filter>
  ```
- De volgende onderdelen importeren:
  ```
  import android.nfc.NfcAdapter;
  import android.nfc.Tag;
  import com.facebook.react.bridge.ReactContext;
  import com.facebook.react.modules.core.DeviceEventManagerModule;
  ```
- Een methode schrijven die intents afhandeld van de nfc sensor en via de React Native bridge het tagId van de gedetecteerde tag kan sturen(voor verschillende type tags):
  ```
  private void handleNFCIntent(Intent intent) {
    String action = intent.getAction();
    ReactContext reactContext = mReactInstanceManager.getCurrentReactContext();
    if(NfcAdapter.ACTION_TAG_DISCOVERED.equals(action) || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)){
        Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
        int tagId = ByteBuffer.wrap(tag.getId()).getInt();
        if(reactContext != null){
          reactContext.getJSModule(DeviceEventManagerModule.RCTDeviceEventEmitter.class).emit("tagId", tagId);
        }
    }
  }
  ```
  Het sturen van het tagId gaat via de React Native bridge met behulp van de RCTDeviceEventEmitter.

[19] http://developer.android.com/guide/topics/connectivity/nfc/index.html

- Het opvangen van het verstuurde tagId in de JavaScript code en de bijbehorende werknemer ophalen:
```
DeviceEventEmitter.addListener('tagId', (id) =>this.fetchEmployee(id));
```

Het uitzoeken hoe iets van de native code naar de JavaScript code gestuurd kon worden duurde langer dan gedacht omdat de React Native documentatie het maar heel kort aan de orde brengt.

7.4 Flushmode

In plaats van de in-memory database te gebruiken, werd nu alle data opgehaald via de RESTful web service. Bij het testen van het in- en uitklokken kwam ik erachter dat niet het laatste project waarvoor ingeklokt was werd getoond bij het projecten selecteer scherm. Dit kwam door de flushmode van de web service. De web service werd niet na elke toevoeging, wijziging of verwijdering in de database gesynchroniseerd. Hierdoor werd er data getoond werd dat niet up-to-date met de database was. Ik heb de flushmode aangepast, zodat er elke keer als in of uitgeklokt wordt er gesynchroniseerd wordt.

7.5 Resultaat

Het resultaat van deze sprint was een werkende applicatie met een nieuw geïmplementeerd design. Er is alleen gebruik gemaakt van cross-platform componenten, hierdoor zal de app op de tagdetectie na ook voor iOS werken. Tijdens het ontwikkelen van de applicatie liep ik hier en daar tegen wat problemen aan die lastig waren om op te lossen vanwege de geringe documentatie en hoe nieuw het React Native framework is. Uiteindelijk heb ik alle problemen kunnen oplossen, maar heeft het wel langer geduurd dan gedacht. Bij de sprint review kwam ik erachter dat in tegenstelling tot eerder, meerdere keren per dag voor hetzelfde project ingeklokt moet kunnen worden. Ook wilden de stakeholders foutmeldingen zien als er iets fout gaat. Verder werd voor de in en uitkloktijd de tijd van het device gebruikt. Dit was niet wenselijk omdat als er iets mis gaat met de tijd van het device, de in en uitkloktijden niet meer kloppen. De oplossing hiervoor was om de servertijd te gebruiken. De foutafhandeling, het meerdere keren per dag voor hetzelfde project in kunnen klokken en de servertijd gebruiken voor het in- en uitklokken zijn naar de volgende sprint als nieuwe user stories meegenomen.

8 Sprint 3 Ontwerpen en bouwen webapplicatie

Het doel van deze sprint was om de mobiele applicatie af te maken en een begin aan de webapplicatie te maken. De volgende user stories waren van toepassing op deze sprint:

- Als gebruiker wil ik meerdere keren per dag voor hetzelfde project kunnen inklokken, zodat als ik tussendoor voor een ander project een werkzaamheid moet uitvoeren, ik weer kan inklokken voor het project waar ik eerst mee bezig was.
- Als manager wil ik dat de tijd van de server wordt gebruikt voor het in- en uitklokken, zodat als er iets mis gaat met de tijd van het device, deze tijden nog kloppen.
- Als gebruiker wil ik een foutmelding zien als er iets misgaat in de applicatie, zodat ik weet wat er fout is gegaan en ik dit door kan geven of kan corrigeren.
- Als manager wil ik werknemers kunnen aanmaken, zodat ik deze kan koppelen aan projecten.
- Als manager wil ik projecten kunnen aanmaken, zodat ik deze kan koppelen aan werknemers.

8.1 Mobiele applicatie afmaken

Om de mobiele applicatie af te maken moest nog aan een aantal user stories, die er bij de sprint review aan het einde van sprint 2 bijgekomen zijn, worden voldaan. Deze worden hier besproken.

Meerdere keren in- en uitklokken per dag voor een project

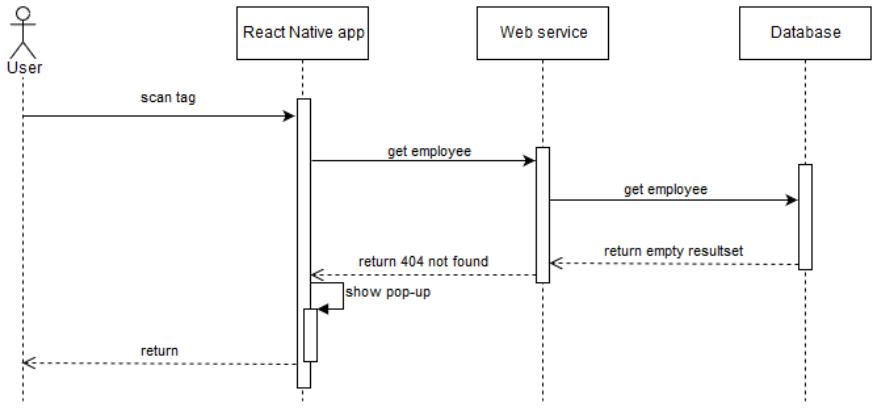
Het meerdere keren per dag kunnen in- en uitklokken voor hetzelfde project was niet lastig om aan te passen. Om ervoor te zorgen dat er maar een keer per project per dag ingeklokt kan worden had ik in de database aangegeven dat de verzameling van employee_id, project_id en check_in datum uniek moest zijn. Het enige wat ik moest doen om aan de user story te voldoen was deze restrictie verwijderen.

Het gebruiken van de servertijd

Om de servertijd te gebruiken in plaats van de tijd van het Android device heb ik een endpoint gemaakt op de web service. Deze endpoint geeft de huidige tijd terug van de server. In de applicatie roep ik de endpoint aan bij het in- en uitklokken en stuur ik de tijd die teruggeven wordt mee met de in- of uitklok request. Zo wordt de servertijd in als in- en uitklok tijd gebruikt.

Foutafhandeling

Om foutafhandeling in te bouwen wordt eerst de internetconnectie van het device gecontroleerd als er geen internetconnectie is wordt een pop-up getoond met “No connection, please connect to the internet”. Voor de overige fouten moesten de status codes die de web service meestuurt met zijn responses geïnterpreteerd worden en vervolgens een pop-up getoond worden met informatie over wat er mis is. Bij het scannen van een tag die niet gekoppeld is aan een medewerker krijg je bijvoorbeeld een pop-up te zien met “Employee not found, please contact your manager”. Dit ziet er als volgt uit:



Figuur 26 Foutafhandeling geen employee

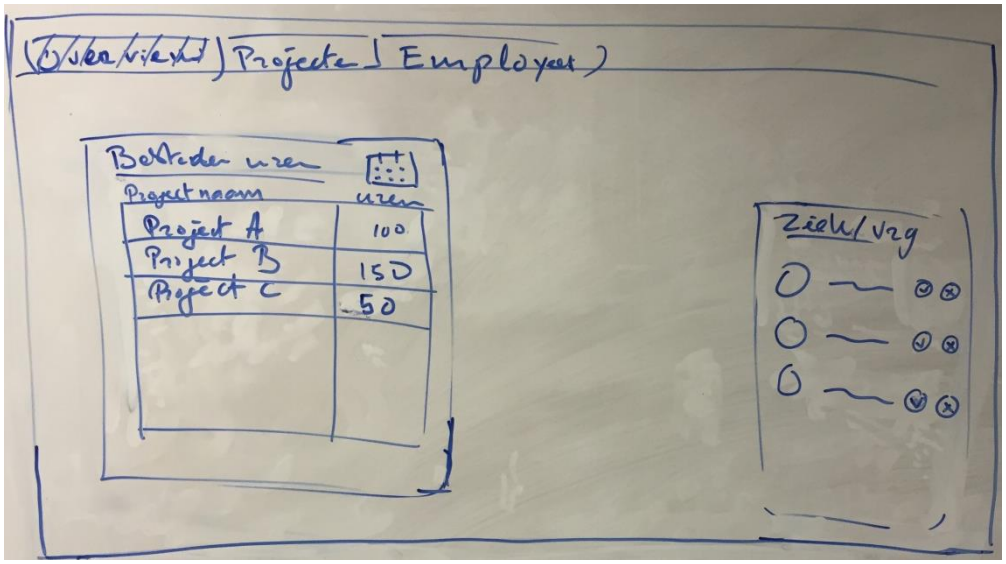
Na deze toevoegingen is nog een kleine design toepassing doorgevoerd op aanvraag van de designer. Alle user stories voor de Android applicatie waren nu afgerond. Na het testen van de applicatie met de nieuwe toevoegingen kon ik beginnen aan de webapplicatie.

8.2 Begin webapplicatie

Aangezien de Android applicatie met React Native gemaakt is en dit gebaseerd is op React is ervoor gekozen om de webapplicatie met React te maken. Qua syntax zijn er geen verschillen tussen de twee. Hierdoor zou er code uit de Android applicatie herbruikt kunnen worden voor de webapplicatie.

Wireframes

Net als bij de Android applicatie zijn er voor de webapplicatie eerst met de designer wireframes gemaakt. Deze worden hieronder beschreven:



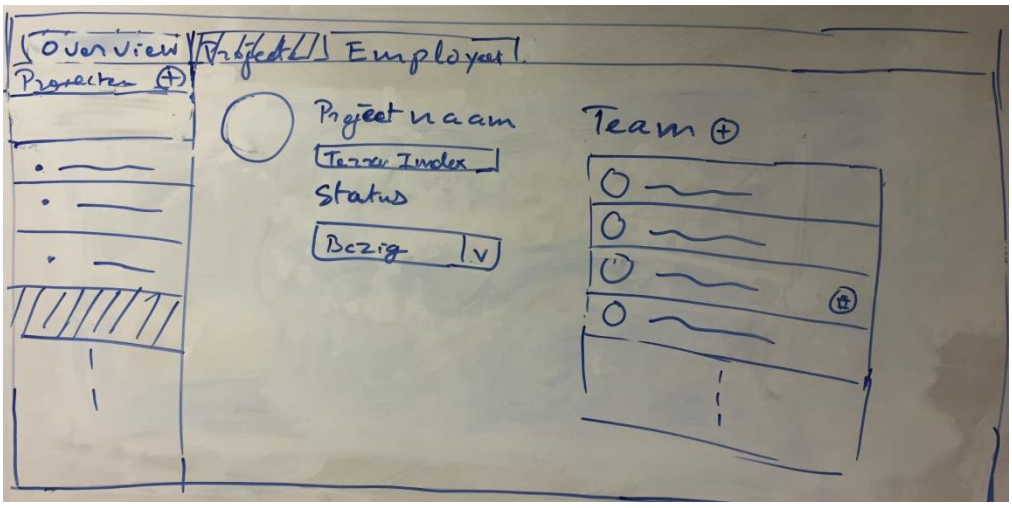
Figuur 27 Wireframe overview scherm

Het Overview scherm is het eerste scherm waar een manager beland als er ingelogd is. Hier moet een overzicht te zien kunnen zijn van de projecten en de besteedde uren daaraan. Als er op het kalender icoon geklikt wordt, moet een tijdsbereik gespecificeerd kunnen worden waarvan de uren van de projecten getoond moeten worden. Ook moeten de ziekmeldingen en vrij aanvragen met de naam van de medewerker te zien kunnen zijn, zodat deze goed of afgekeurd kunnen worden. Dit scherm is gebaseerd op twee nieuwe user stories voor de web applicatie:

- Als manager wil ik de besteedde uren binnen een gespecificeerd bereik kunnen zien, zodat ik hier overzicht over heb.
- Als manager wil ik de ziekmeldingen en/of vrij aanvragen kunnen zien, zodat ik dit goed of af kan keuren.

Figuur 28 Wireframe projecten scherm 1

Het projecten scherm bestaat uit twee delen. Op het eerste deel van het projecten scherm moeten projecten geselecteerd kunnen worden. Van het geselecteerde project moet vervolgens een grafiek te zien zijn op jaar, maand, week of dag basis met de totale gewerkte uren van een project en de gewerkte uren per werknemer. Rechts van de grafiek is het team dat aan het project gekoppeld is te zien. Als er op een van de teamleden wordt geklikt moet de lijn in de grafiek van het teamlid benadrukt worden. Als er op het plusteken bij het team geklikt wordt, moeten er nieuwe teamleden aan het geselecteerde project toegevoegd kunnen worden. Als er op het plusteken of het wijzigteken bij projecten wordt geklikt komt deel 2 van het projectenscherm tevoorschijn.

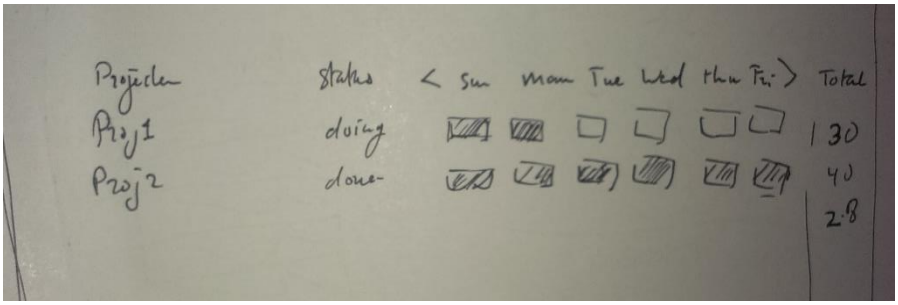


Figuur 29 Wireframe projecten scherm 2

Op het tweede deel van het projecten scherm moeten projecten aangemaakt of gewijzigd kunnen worden. Ook moeten er teamleden aan een project gekoppeld kunnen worden. Voor de projectschermen gelden de volgende user stories:

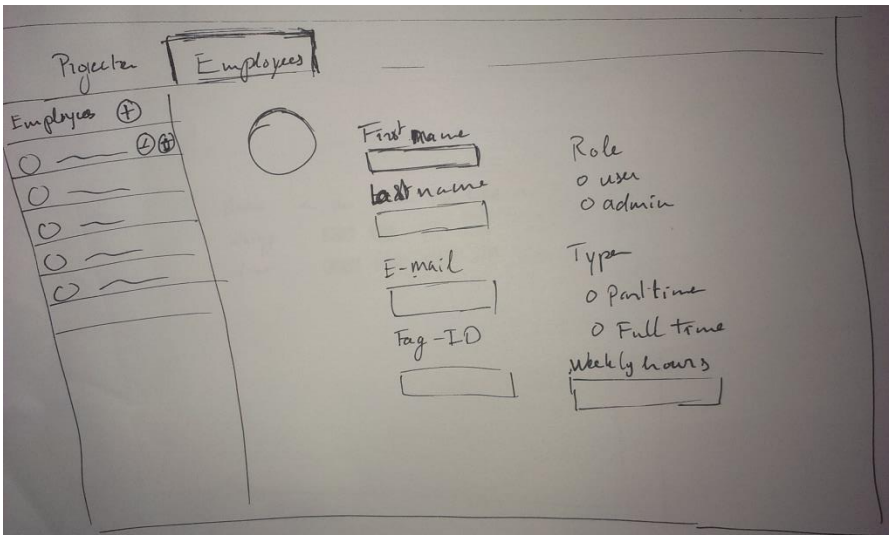
- Als manager wil ik de totale gewerkte uren en de uren per werknemer op jaar, maand, week en dag basis per project in een grafiek zien, zodat ik hier overzicht over heb.

- Als manager wil ik projecten kunnen aanmaken, zodat ik deze kan koppelen aan werknemers.
- Als manager wil ik projecten kunnen wijzigen, zodat de projecten up-to-date gehouden kunnen worden.
- Als manager wil ik projecten kunnen verwijderen, zodat geannuleerde projecten niet meer te zien zijn.
- Als manager wil ik werknemers aan projecten kunnen koppelen, zodat ik kan zien hoeveel uur er is gewerkt per werknemer per project.
- Als manager wil ik de geselecteerde werknemer benadrukt in de grafiek zien, zodat ik dit makkelijker kan bekijken.



Figuur 30 Wireframe employee scherm 1

Dit scherm geldt voor normale medewerkers en managers. Het enige verschil bij het scherm van de managers is dat zij nog kunnen selecteren voor welke medewerker zij de tabel willen zien/aanpassen en knoppen hebben om medewerkers toe te voegen, wijzigen of verwijderen. Op dit scherm is een tabel te zien met de uren dat gewerkt is per project in een bepaalde week. Als er op de pijl naar links wordt geklikt worden de uren van een week terug getoond, als er op de pijl naar rechts worden de gegevens getoond van een week verder. In de tabel kan er op een cel geklikt worden om de uren te wijzigen, dit is handig als iemand bijvoorbeeld vergeten is in of uit te klokken via de Android applicatie. De restrictie bij het wijzigen van uren is dat alleen de uren van de huidige salarisperiode aangepast kunnen worden.



Figuur 31 Wireframe employee scherm 2

Op dit scherm moeten werknemers aangemaakt of gewijzigd kunnen worden. De volgende user stories gelden voor de employee schermen:

- Als gebruiker wil ik een overzicht van mijn gewerkte uren per week kunnen zien, zodat ik dit kan controleren.
- Als gebruiker wil ik mijn gewerkte uren in de huidige salarisperiode kunnen aanpassen, zodat ik dit kan corrigeren als ik vergeten ben in te klokken, of voor het verkeerde project ingeklokt heb.
- Als manager wil ik werknemers kunnen aanmaken, zodat ik deze kan koppelen aan projecten.
- Als manager wil ik werknemers kunnen wijzigen, zodat de werknemers up-to-date gehouden kunnen worden.
- Als manager wil ik werknemers kunnen verwijderen, zodat ik geen data meer zie van oud werknemers.

Projecten en employees aanmaken

Nadat de wireframes af waren ben ik aan de slag gegaan met de functionaliteit om projecten en werknemers toe te voegen. Hierbij is nog geen aandacht besteedt aan het design, omdat de designer nog bezig was met het definitieve design. Eerst heb ik de navigatie in elkaar gezet zodat er tussen de overview/project/employee schermen genavigeerd kan worden. Vervolgens heb ik de invoervelden en knoppen op de projecten pagina voor het aanmaken van een project en op de employee pagina voor het aanmaken van een employee geplaatst. Daarna heb ik de functionaliteit toegevoegd die een project/werknemer in de database zet en een pop-up laat zien als dit gelukt is. Hiervoor gebruikt de webapplicatie, zoals te zien in figuur 20, dezelfde web service als de Android applicatie. Dit verliep er soepel. Nadat de functionaliteit geïmplementeerd was is het getest.

8.3 Resultaat

Het doel van deze sprint is behaald. De nieuwe user stories voor de Android applicatie zijn verwerkt. Er is een ontwerp gemaakt voor de webapplicatie en het navigeren, projecten aanmaken en werknemers aanmaken is geïmplementeerd en getest. Bij de sprint review waren de stakeholders positief over de nieuwe toevoegingen aan de Android applicatie en het begin aan de webapplicatie. Een van de stakeholders vertelde mij dat de salarisperiode binnen het bedrijf is verandert. Deze verandering heeft impact op mijn project. De veranderingen die ik hierdoor moest maken zijn naar de volgende sprint meegenomen.

Bij de sprint retrospective is afgesproken dat ik de Scrum master elke dag een bericht stuur voor het houden van de daily stand-up, zodat dit niet vergeten wordt. Deze afspraak is gemaakt omdat naar mate de tijd vorderde de Scrum master er steeds minder daily stand-ups werden gehouden.

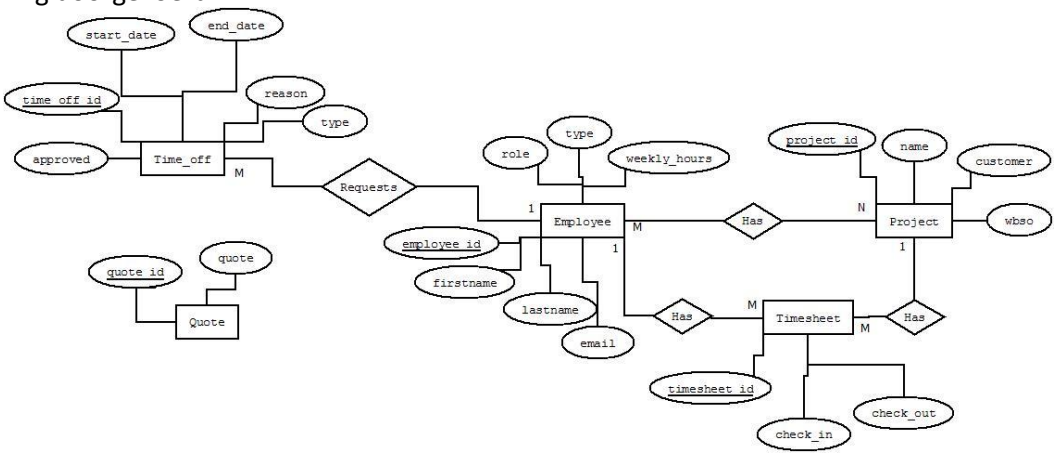
9 Sprint 4 Uitbreiden webapplicatie

Het doel van deze sprint was om de veranderingen die het gevolg waren van de verandering van salarisperiode binnen het bedrijf door te voeren en de functionaliteit van de webapplicatie uit te breiden. De volgende user stories waren van toepassing op deze sprint:

- Als manager wil ik onderscheid kunnen maken tussen parttimers en fulltimers, zodat ik bij de fulltimers kan aangeven hoeveel contracturen zij werken.
- Als manager wil ik een rol kunnen aangeven bij het aanmaken van werknemers, zodat op basis hiervan de juiste pagina geladen kan worden.
- Als fulltimer wil ik bij het inklokken zien hoeveel uur ik nog gemiddeld per dag moet werken, zodat ik weet of ik nog op schema loop.
- Als gebruiker wil ik met google kunnen inloggen, zodat ik mijn bedrijfsemail kan gebruiken.
- Als gebruiker wil ik een overzicht van mijn gewerkte uren kunnen zien, zodat ik dit kan controleren.

9.1 Aanpassing salarisperiode

Door de verandering van salarisperiode moesten de database, de web service en de Android applicatie aangepast worden. Eerst was de salarisperiode een bepaalde maand(bijvoorbeeld januari, februari, maart etc.). Per maand werd dan aangegeven hoeveel uur er totaal per werknemer gewerkt moest worden. Door de verandering werd dit een periode vanaf de 21ste van een maand tot en met de 20ste van de daarop volgende maand(bijvoorbeeld 21 december t/m 20 januari) en zal ook niet meer aangegeven worden per werknemer per maand hoeveel gewerkt moest worden. Bij de fulltimers wordt dit bepaald door het aantal contracturen en bij de parttimers is dit niet bekend. In het volgende ERD is de verandering doorgevoerd:



Figuur 32 Aangepast ERD

Als we dit ERD vergelijken met het ERD in figuur 11 dan is te zien dat de Schedule entiteit ontbreekt. Dit komt omdat hier eerst de aantal te werken uren per maand per werknemer ingezet werden. De bedoeling is dat dit nu niet meer gedaan wordt. Ook zijn er drie nieuwe attributen bij de Employee entiteit gekomen en wordt bij de Project entiteit het totaal aantal uur niet meer bijgehouden, in plaats daarvan is WBSO toegevoegd. Hieronder een korte beschrijving van de nieuwe attributen:

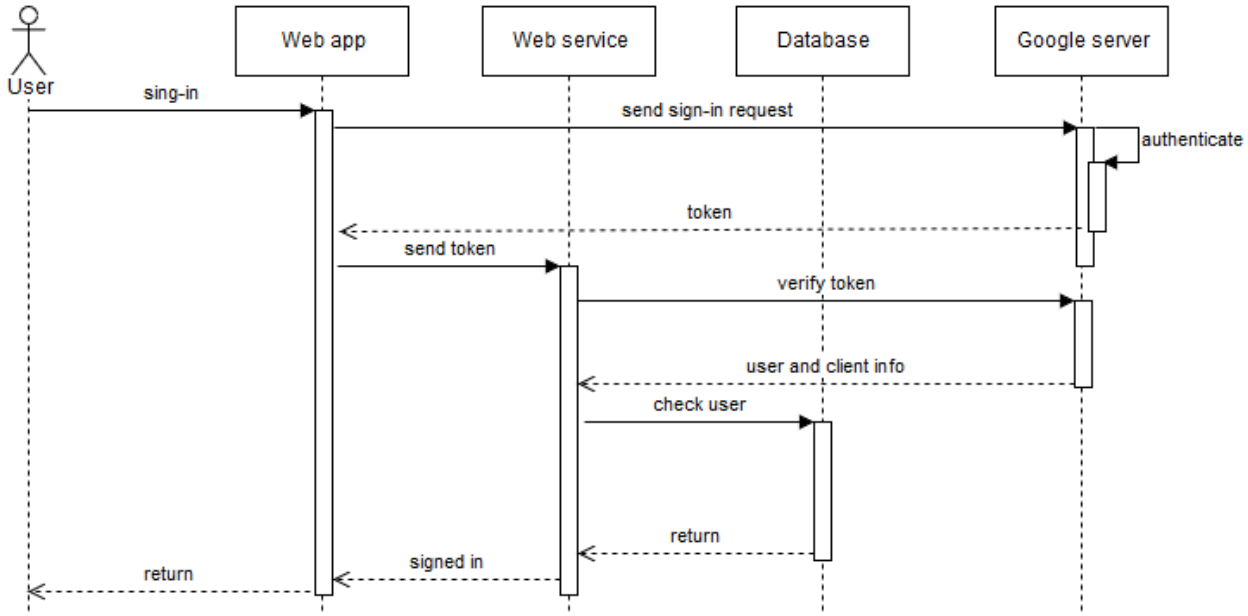
- role: hierin staat de rol van een werknemer, dit kan manager of een gewone medewerker zijn, op basis hiervan kan bij het inloggen op de webapplicatie de juiste pagina geladen worden.
- type: het type kan parttime of fulltime zijn, fulltime medewerkers zien op de Android applicatie hoeveel uur er gemiddeld per dag te gaan is, parttimers niet.

- weekly_hours: hier staan de contracturen van fulltime medewerkers.
- wbso: geeft aan of WBSO van toepassing is op een project, hiermee kan bepaald worden voor welke projecten een subsidie geldt.

Naast deze aanpassing van de database is ook de web service aangepast op deze verandering(Schedule klasse is verwijderd en role, type en weekly_ hours zijn als properties bij de Employee klasse gekomen). Verder is in de Android applicatie de berekening van het gemiddeld aantal uur te gaan op een dag en de bijbehorende unit tests aangepast. Ook is er een check op het type medewerker toegevoegd bij het inklok scherm, voor fulltimers wordt wel het gemiddeld aantal uur te gaan op een dag berekend en getoond, maar voor parttimers niet(omdat zij geen vast aantal contracturen hebben en dit dus niet berekend kan worden).

9.2 Google sign-in

Het inloggen in de webapplicatie is met Google sign-in geregeld. Dit is gedaan zodat de werknemers niet een nieuwe account hoeven aan te maken om op de webapplicatie in te loggen omdat alle werknemers al een Google bedrijfsemail hebben. Het implementeren is gedaan met behulp van de guide van Google.[20] Het inloggen ziet er als volgt uit:



Figuur 33 Google sign-in sequence diagram

De gebruiker klikt op inloggen op de webapplicatie, vervolgens wordt de login aanvraag naar de Google server gestuurd om daar geauthentiseerd te worden. Als de gebruiker geauthentiseerd is wordt er een token teruggestuurd. Deze token wordt naar de web service gestuurd, zodat de web service kan verifiëren met de Google server of deze token bedoeld is voor de webapplicatie. Als dit zo is wordt de gebruiker uit de database gehaald op basis van de gebruikte email bij het inloggen. Vervolgens krijgt de gebruiker toegang tot de webapplicatie.

[20] https://developers.google.com/identity/sign-in/web/devconsole-project

9.3 Functionaliteit webapplicatie

Nadat het inloggen van de Google sign-in geïmplementeerd en getest was is er aan de functionaliteit van de urenoverzicht tabel gewerkt. Deze tabel moet door alle gebruikers van de webapplicatie te zien zijn. In de tabel staan de uren die in een week(zondag t/m zaterdag) gewerkt zijn per project gesorteerd, met het totaal aantal uur zie figuur 30. Ook moeten de uren gezien kunnen worden van andere weken, door een van de pijlen te klikken bij de tabel.

Om de tabel te vullen wordt eerst opgehaald welke datum het is. Vervolgens wordt uit de datum de dag nummer gehaald(0=zondag 6=zaterdag). Met deze informatie wordt bepaald wat de eerste dag van de week is en wat de laatste dag van de week is en wordt er een request naar de web service gestuurd om de gewerkte uren op te halen vanaf de begin datum tot en met einddatum van de week. De response van de web service wordt per project, per dag gefilterd en op de juiste plek in de tabel gezet.

Omdat er hierna nog wat tijd over was is er ook aan de volgende user stories gewerkt:

- Als manager wil ik werknemers kunnen wijzigen, zodat de werknemers up-to-date gehouden kunnen worden.
- Als manager wil ik werknemers kunnen verwijderen, zodat ik geen data meer zie van oud werknemers.
- Als manager wil ik projecten kunnen wijzigen, zodat de projecten up-to-date gehouden kunnen worden.
- Als manager wil ik projecten kunnen verwijderen, zodat geannuleerde projecten niet meer te zien zijn.
- Als manager wil ik werknemers aan projecten kunnen koppelen, zodat ik kan zien hoeveel uur er is gewerkt per werknemer per project.
- Als manager wil ik de koppeling tussen een werknemer er project verwijderen, zodat als ik toch besluit dat de werknemer beter niet aan het project kan werken, ik dit kan corrigeren.

Het implementeren van deze user stories ging redelijk voorspoedig, omdat de functionaliteit om werknemers en projecten te kunnen wijzigen/verwijderen al op de web service gegenereerd waren. Hiervoor hoefde ik dus alleen nog knoppen te plaatsen waarmee de requests naar de web service worden verzonden.

Tijdens het testen kwam ik erachter dat bij het verwijderen van de koppeling tussen een werknemer en project, het project en de werknemer zelf ook verwijdert werden. Dit is natuurlijk niet de bedoeling. Het bleek te komen doordat bij het genereren van de web service automatisch aan is gegeven dat alle operaties die op de koppeling werknemer-project gedaan worden, ook op de betreffende werknemer en het project gedaan moeten worden(CascadeType.ALL). Nadat ik dit heb gecorrigeerd werkte alles zoals verwacht.

9.4 Resultaat

Het resultaat van deze sprint was dat de benodigde aanpassingen die voort zijn gekomen uit de verandering van salarisperiode waren doorgevoerd, de Google sign-in was geïmplementeerd en de basisfunctionaliteit van de webapplicatie af was. Hiermee waren alle doelen van de sprint behaald. De sprint review verliep goed, de stakeholders waren tevreden met de voortgang. Door de maatregel die genomen was aan het eind van de vorige sprint werden de daily stand-ups weer vaker gehouden.

10 Sprint 5 Afronden webapplicatie

Het doel van deze sprint was om de webapplicatie af te maken. Inmiddels waren de designs van de webapplicatie af en kon ik hier ook aandacht aan besteden. Omdat dit in de vorige sprints niet is gedaan heb ik hier een groot deel van de tijd aan moeten besteden. De volgende user stories waren van toepassing op deze sprint:

- Als gebruiker wil ik mijn gewerkte uren in de huidige salarisperiode kunnen aanpassen, zodat ik dit kan corrigeren als ik vergeten ben in te klokken, of voor het verkeerde project ingeklokt heb.
- Als manager wil ik de totale gewerkte uren en de uren per werknemer op jaar, maand, week en dag basis per project in een grafiek zien, zodat ik hier overzicht over heb.
- Als manager wil ik aan het eind van elke maand een mail ontvangen met de gewerkte uren en WBSO uren per werknemer, zodat ik dit naar de accountant kan sturen.

10.1 Functionaliteit webapplicatie afmaken

Het kunnen aanpassen van de uren van de huidige salarisperiode was een belangrijke functionaliteit. Hierdoor zouden mensen die vergeten zijn via de Android applicatie in- en/of uit- te klokken alsnog hun uren kunnen invullen of corrigeren. Dit aanpassen gebeurd in dezelfde tabel als waar zij hun overzicht op weekbasis per project kunnen zien. Eerst wordt de huidige salarisperiode bepaald. Vervolgens worden alle cellen uit de tabel die binnen deze periode vallen klikbaar gemaakt. Als er op een van deze cellen wordt geklikt komt er een pop-up tevoorschijn met een invoerveld voor het aantal uur en kan dit ingevuld worden.

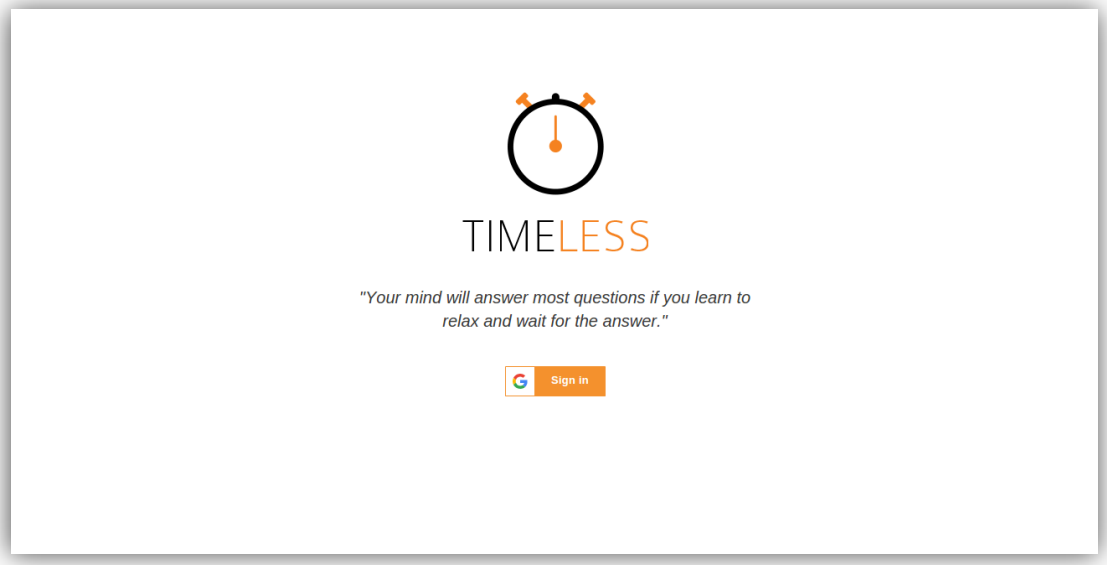
Het maken van de grafiek is gedaan met behulp van Chart.js, een JavaScript library waarmee je simpele grafieken kunt maken. Ik heb voor Chart.js gekozen omdat bij het oriënteren op hoe de verschillende mogelijkheden geïmplementeerd kunnen worden deze het makkelijkst voor mij te volgen was. Toch ben ik tegen een probleem aangelopen tijdens de implementatie. De grafiek werd niet geüpdatet als ik andere data gebruikte. Omdat andere mensen dit probleem ook hebben gehad, was het makkelijk om op te lossen. De oplossing was dat er expliciet aangegeven moet worden dat de grafiek moet updaten. Verder moest de data voor de grafiek eerst gefilterd worden, zodat de juiste data bij de juiste label van de grafiek terecht kwam. Het filteren duurde een tijdje om uit te vogelen, maar is uiteindelijk gelukt.

Omdat er niet veel tijd meer over was is er met de product owner besloten om het overview scherm te laten vallen. Op dit scherm moesten ziekmeldingen en vrij aanvragen goed of afgekeurd kunnen worden en een overzicht van de totale besteedde uren van de projecten in een gespecificeerd bereik gezien kunnen worden. Aangezien de totale gewerkte uren voor een project op jaar, maand en weekbasis in de grafiek gezien kunnen worden bleef die functionaliteit wel bestaan maar op een andere manier. Wel is de functionaliteit voor het ziekmelden/vrij aanvragen hierdoor uiteindelijk niet geïmplementeerd in de applicatie.

De laatste user story die geïmplementeerd is, is het elke maand versturen van een e-mail met de gewerkte uren en het gedeelte ervan dat gewerkt is aan een WBSO project per werknemer in een salarisperiode. Hiervoor is een TimerTask opgezet die na elke salarisperiode uitgevoerd wordt. Om de mail te sturen wordt de JavaMail API gebruikt. In de TimerTask wordt de mail opgesteld en verzonden naar een e-mail adres.

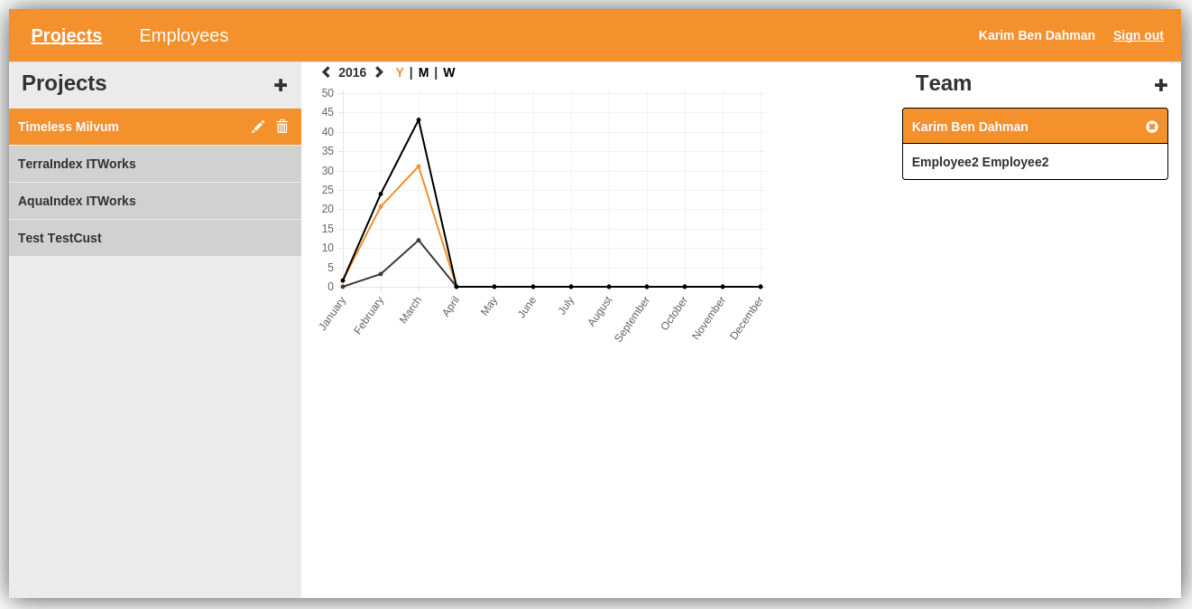
10.2 Styling webapplicatie

De styling van een React applicatie gaat met behulp van CSS. Hier had ik nog geen ervaring mee, maar omdat het sterk op de manier van stylen leek in React Native kon ik dit snel oppakken. Het design dat voor de webapplicatie is gemaakt is te vinden in bijlage D. Het stylen met CSS was erg prettig, omdat je bijvoorbeeld met Chrome developer tools in de browser de styling kan wijzigen en als dit naar wens is, je dit over kan nemen naar je applicatie. Uiteindelijk was het stylen moeilijker dan gedacht. Bij het selecteren van een project moesten bijvoorbeeld 2 knoppen bij het geselecteerde project komen(zie figuur 35). De projectenlijst staat opgeslagen in een array, om de knoppen bij het project te krijgen moest de plek waarop het geselecteerde project stond in de array uitgewisseld worden met hetzelfde project plus 2 knoppen. Dit leek voor mij erg omslachtig. Zie hieronder voor het uiteindelijke resultaat van het geïmplementeerde design:



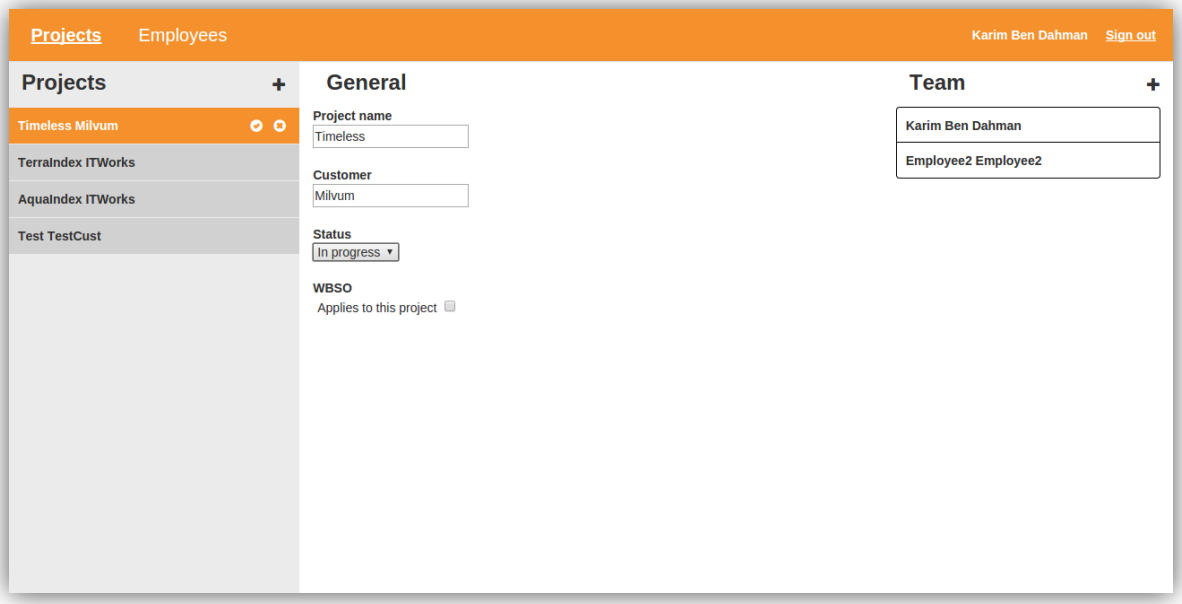
Figuur 34 Login scherm

Dit is het scherm waar je beland als je naar de webapplicatie gaat, hierop staat het logo van de applicatie, een citaat en een Google sign-in knop.



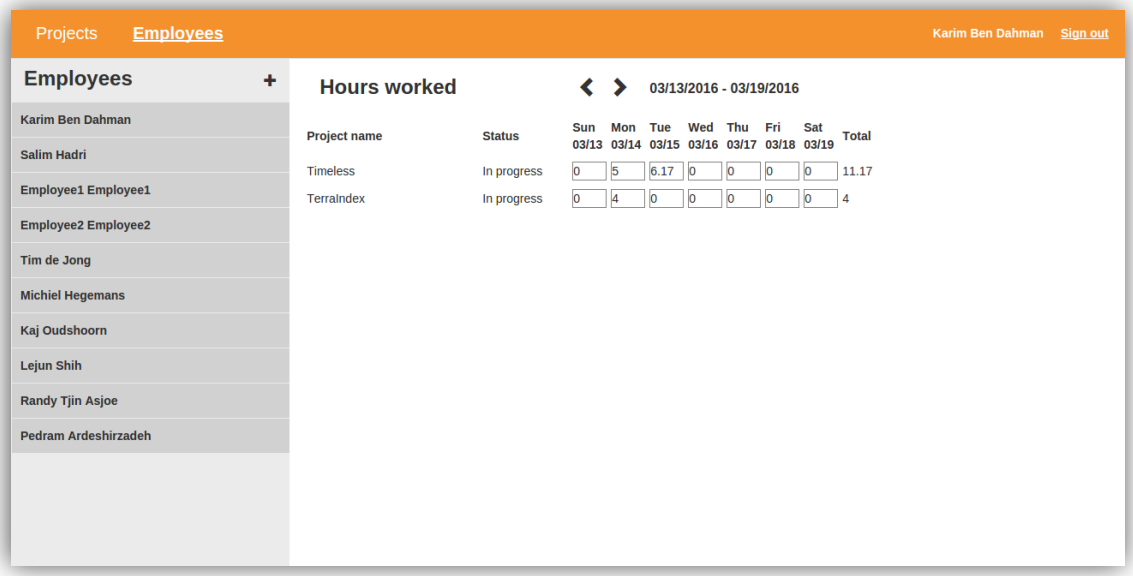
Figuur 35 Projecten scherm 1

Op dit scherm kunnen de gewerkte uren per werknemer per project gezien worden op jaar maand of weekbasis. De lijn van de geselecteerde werknemer wordt in het oranje benadrukt.



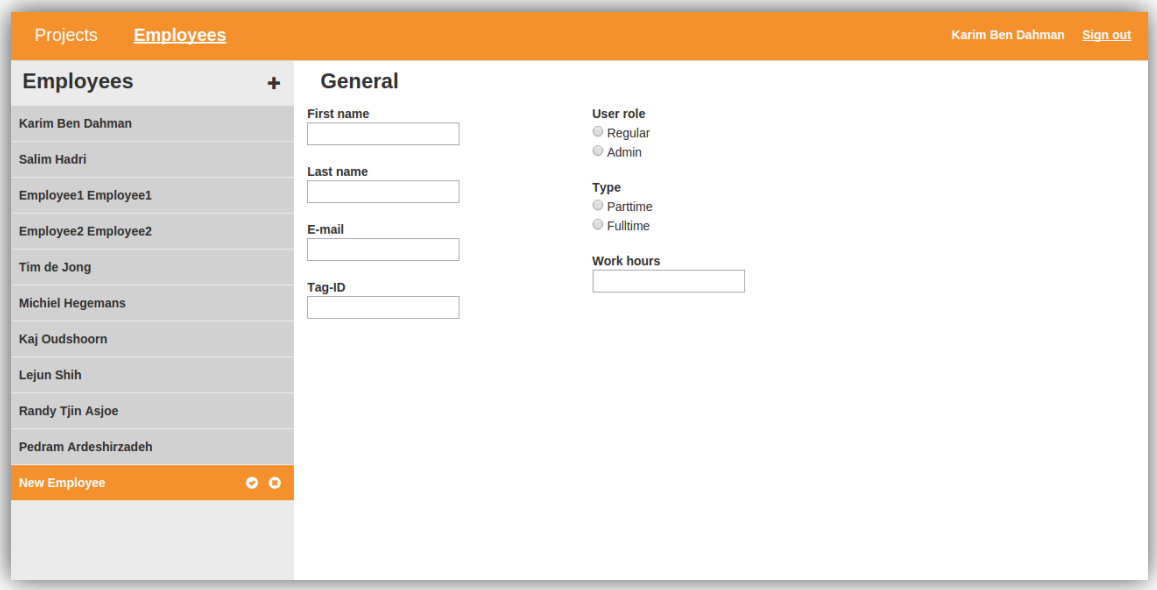
Figuur 36 Projecten scherm 2

Op dit scherm kunnen projecten gewijzigd worden, het scherm om projecten aan te maken ziet er hetzelfde uit.



Figuur 37 Employee scherm 1

Op dit scherm kunnen de uren die gewerkt zijn per werknemer, per week, per project gezien worden. Als er op een cel geklikt waarvan de datum zich in de huidige salarisperiode bevind kan het aantal uur gewijzigd worden. Dit scherm geldt ook voor werknemers die geen manager zijn, alleen kunnen zij alleen hun eigen uren zien en aanpassen.



Figuur 38 Employee scherm 2

Op dit scherm kunnen nieuwe werknemers toegevoegd worden, het scherm voor het wijzigen van werknemers ziet er hetzelfde uit.

10.3 Resultaat

Het resultaat van deze sprint was dat de uren in het urenoverzicht tabel aangepast kunnen worden in de huidige salarisperiode. Verder kan een overzicht van de gewerkte uren per werknemer gezien worden op jaar, maand en week basis in een grafiek, wordt er elke maand een e-mail gestuurd met het aantal gewerkte uur per werknemer en is het design verwerkt. Wegens de beperkte tijd die over was, is er besloten om de ziekmeld/vrij aanvraag functionaliteit niet te bouwen. Ondanks dit waren de stakeholders erg blij met wat ik deze sprint heb kunnen doen.

11 Productevaluatie

Er is een Android applicatie gemaakt waarmee het uren registreren voor de medewerkers een stuk prettiger gaat, het enige wat zij hoeven te doen is een tag scannen en een project te selecteren voor het inklokken. Voor het uitklokken hoeft alleen een tag gescanned te worden. Dit is een flinke verbetering van de urenregistratie die eerst handmatig ging.

De applicatie is behalve de tagdetectie, volledig gemaakt met cross-platform componenten en code. Hierdoor kan de applicatie in de toekomst makkelijk ontwikkeld worden voor iOS en eventueel andere platformen die ondersteund zullen worden door React Native. De code is door een medewerker gereviewd. Bij de review kwamen een aantal punten naar voor waarop de code nog verbeterd kon worden. De code is aangepast op het commentaar, zodat het aan de standaarden van het bedrijf voldoet.

Hoewel het mogelijk is om met React Native applicaties te maken is het nog erg jong(versie 0.21), hierdoor zijn er nog aantal problemen met performance en is de documentatie naar mijn mening van slechte kwaliteit. Het verbeteren van de documentatie staat wel op de to-do lijst van de React Native community, dus zij zijn zich hier wel van bewust. Ik heb geadviseerd om nog te wachten met het ontwikkelen met React Native totdat de huidige problemen zijn opgelost en er fatsoenlijke documentatie is.

De webapplicatie maakt het makkelijk voor de managers om het overzicht over de uren te behouden. Ook kunnen hier de uren aangepast worden indien er vergeten is met de Android applicatie in- en/of uit- te klokken. Bij de webapplicatie is geen code review gedaan door een ander, hierdoor kan de kwaliteit van de code minder zijn dan die van de Android applicatie. De enige user story die ik niet heb kunnen doorvoeren was het ziekmelden/vrij aanvragen. Wel staat de web service en database al klaar voor deze functionaliteit.

Op dit moment is het niet veilig om de web service extern te laten benaderen, omdat er geen veiligheidsmaatregelen geïmplementeerd zijn. Ik heb het advies gegeven om eerst alles intern te laten draaien en pas extern te zetten als de web service is beveiligd.

Ik ben tevreden over de opgeleverde producten, ze dienen waarvoor ze bedoelt zijn en maakt het proces van het registreren van uren voor werknemers en het beheren van uren voor managers makkelijker.

12 Procesevaluatie

In de eerste sprint is een plan van aanpak opgesteld waarin onder andere de ontwikkelmethode, de risicofactoren en de globale planning beschreven staan. De grootste risicofactoren waren dat ik nog geen ervaring had met JavaScript, React Native en GraphQL. Daarom is er aan het begin tijd besteedt om over deze onderwerpen te leren. Ervaring opdoen met JavaScript was geen probleem, omdat er veel documentatie over te vinden is op het web en er ook goede tutorials voor zijn. Echter voor React Native en GraphQL/Relay is de documentatie niet heel uitgebreidt en zijn ook niet veel voorbeelden over te vinden. Dit komt mede omdat React Native en GraphQL/Relay zeer nieuw zijn. React Native voor Android is bijvoorbeeld pas in september 2015 open-sourced. Binnen het bedrijf was hier ook nog geen ervaring mee. Hierdoor heeft het ontwikkelen van de applicatie langer geduurd dan gedacht. Het is wel erg leerzaam geweest.

In de planning stonden 8 sprints gepland. Uiteindelijk heb ik in plaats hiervan 5 sprints kunnen uitvoeren. Dit kwam doordat ik de voorbereidende werkzaamheden die gedaan moesten worden om het ontwikkeltraject te beginnen ook gepland had in sprints, hiervoor was anderhalf sprint gebruikt. Bij het tussentijds assessment kreeg ik te horen dat dit beter voor de sprints geplaatst kon worden. Dit advies heb ik opgevolgd. Een andere factor was de ziekte van de product owner. Door de ziekte van de product owner kon de planning voor de volgende sprint niet gehouden worden. Tijdens deze periode heb ik verder gewerkt aan features die op de backlog stonden en collega's geholpen met React Native. Verder werd bij het tussentijds assessment ook verteld dat ik beter alle overige tijd aan het verslag kon besteden. Ik heb de werkzaamheden waar ik mee bezig was afgerond en dit advies ook opgevolgd.

Van te voren heb ik gepland dat ik elke vrijdag aan mijn afstudeerverslag zou zitten. In de realiteit is dit vaak niet het geval geweest en ben ik bezig geweest met het verder uitwerken van de sprints. Door de focus op het project werd het maken van het afstudeerverslag soms "vergeten". Hierdoor heb ik uiteindelijk minder tijd aan mijn afstudeerverslag kunnen besteden dan ik had gepland.

Het werken met SCRUM was zeer prettig en geschikt voor dit project. Eisen aan het project zijn meerdere keren veranderd. Door SCRUM kon ik deze veranderingen makkelijk oppakken en doorvoeren. De daily scrums waren ook zeer handig om de voortgang bij te houden. Aan het begin werden de daily scrums elke dag gehouden. Na verloop van tijd werd op steeds minder dagen de daily scrum gehouden. Dit kwam door een druk schema van de Scrum Master. Om dit op te lossen is de sprint retrospective van sprint 3 de afspraak gemaakt dat ik de Scrum Master elke dag herinner aan de daily scrum. Nadat deze afspraak gemaakt was, werden de daily scrums weer consistent gehouden. De sprint reviews waren ook handig. Hierdoor kwam ik te weten bij het laten zien van het product of ik op de juiste weg was en konden aanpassingen duidelijk gemaakt worden.

Over het algemeen ben ik tevreden over hoe het proces verlopen is. Wat ik de volgende keer anders zou doen is niet blind ervan uitgaan dat iets te gebruiken is omdat het onderdeel is van de opdracht, maar eerst uitzoeken of het wel mogelijk is. In dit geval was ging het om Relay, ik ging ervan uit dat het zou werken, omdat het onderdeel was van de opdracht. Ik heb redelijk wat tijd besteedt aan het uitzoeken hoe het dan zou moeten werken en ben uiteindelijk tot de conclusie gekomen dat het niet ondersteund werd. Als ik van tevoren had gekeken of het wel te gebruiken was met React Native, dan was er op dit gebied tijd bespaard. Het was een fout van mij om mij niet aan de planning te houden omtrent het schrijven van het afstudeerverslag. Hierdoor heb ik minder tijd kunnen besteden aan het verslag dan gepland. Dit zou ik een volgende keer ook anders doen. Het verslag is namelijk belangrijker dan het product. Uiteindelijk heb ik het verslag wel af kunnen krijgen.

13 Beroepstaken evaluatie

Hier wordt per gekozen beroepstaak besproken waarom ik denk eraan voldaan te hebben.

13.1 Uitvoeren analyse door definitie requirements

In dit project zijn de user stories opgesteld door in gesprek te gaan met de stakeholders en de schermontwerpen te analyseren. De user stories zijn(met een ID voor traceerbaarheid) samen met de product owner opgesteld en geprioriteerd. Voor elke sprint zijn de user stories gepland, ingeschat en verdeeld in taken. Tijdens het project zijn op verschillende momenten user stories aangepast, erbij gekomen of afgevallen. Ook is kritisch gekeken naar het nut van user stories, indien ik het nut van een user story betwijfelde of dacht dat het beter anders kon is dit met de stakeholders overlegd en eventueel veranderd(bijvoorbeeld het ziekmelden/vrij aanvragen in de Android applicatie). Deze beroepstaak is hiermee naar mijn idee voldoende uitgevoerd.

13.2 Ontwerpen systeemdeel

Bij het maken van het afstudeerplan is met de examinatoren afgesproken dat het opstellen van een gegevensmodel voor de database en het bouwen/bevragen hiervan. Er is een ERD opgesteld dat veel op veel en een op veel relaties bevat. Dit model is omgezet naar een MySQL database waarop bevragingen via een web service gedaan worden die betrekking hebben op meerdere tabellen. Voor het opstellen van de maandelijkse e-mail met het aantal gewerkte uren per werknemers en het aantal uur wat daarvan aan een WBSO project is gewerkt zijn drie tabellen nodig, Employee, Project en Timesheet.

Naast het ontwerpen en bevragen van de database zijn er schermontwerpen opgesteld aan de hand van user stories en is er een flow diagram gemaakt om het verloop van de Android applicatie weer te geven. Door het opstellen van het ERD, bouwen en bevragen van de database in combinatie met het ontwerpen van de applicaties, denk ik voldaan te hebben aan deze beroepstaak.

13.3 Bouwen applicatie

Er zijn twee applicaties gebouwd, de Android applicatie en de webapplicatie. Het maken van de Android applicatie is gedaan met het React Native framework en de webapplicatie met behulp van React. Bij het maken van de Android applicatie is er zo veel mogelijk gebruik gemaakt van cross-platform componenten zodat, indien gewenst, de applicatie later makkelijk gemaakt kan worden voor iOS. Ook heb ik functies uit de Android applicatie kunnen hergebruiken in de webapplicatie(bijvoorbeeld het ophalen van projecten waaraan iemand gekoppeld is of het klokken van uren). Ook is er een web service gemaakt waarvan beide applicaties gebruik maken. De versiebeheer van de applicaties is gedaan met behulp van git, waarbij de repository op gitlab is gehost. Bij aanvang van het project had ik nog geen ervaring met React Native, React en JavaScript. Ondanks de geringe documentatie over React Native heb ik toch een applicatie hiermee kunnen bouwen en de ontwikkelomgeving kunnen opzetten. Hierdoor heb ik aan deze beroepstaak voldaan.

13.4 Uitvoeren van en rapporteren over het testproces

Tijdens het bouwen van de applicaties zijn er per sprint unit tests opgesteld met behulp van Jest en is er veel exploratory testing gedaan. Doordat er aan het begin continuous integration is opgesteld werden de unit tests bij elke push naar de remote repository automatisch opnieuw getest(Zie 5.6). De uitgevoerde testen zijn per sprint gedocumenteerd met de user story waaraan ze gelinkt zijn. Hiermee denk ik voldaan te hebben aan deze beroepstaak.

Literatuurlijst

Boeken

Eisenman, B. (2015). *Learning React Native.* O'Reilly.

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

Websites

MoSCoW-methode. (2015, 6 oktober). Geraadpleegd van https://nl.wikipedia.org/wiki/MoSCoW-methode

Codecadamy. (2015). Geraadpleegd van https://www.codecademy.com/

Google JavaScript Style Guide. (z.d.). Geraadpleegd van https://google.github.io/styleguide/javascriptguide.xml

Airbnb JavaScript Style Guide. (2015). Geraadpleegd van https://github.com/airbnb/javascript

React Native. (2015). Geraadpleegd van https://facebook.github.io/react-native/

NativeScript. (2015). Geraadpleegd van https://github.com/NativeScript/NativeScript/

JSX in Depth. (2015, 22 december). Geraadpleegd van https://facebook.github.io/react/docs/jsx-in-depth.html

View. (2015, 27 november). Geraadpleegd van https://facebook.github.io/react-native/docs/view.html

Thinking in React. (2015, 21 december). Geraadpleegd van https://facebook.github.io/react/docs/thinking-in-react.html

Relay. (2015). Geraadpleegd van https://facebook.github.io/relay/

Intoduction to GraphQL. (2015). Geraadpleegd van https://learngraphql.com/basics/introduction

Thinking in GraphQL. (2015, 6 oktober). Geraadpleegd van https://facebook.github.io/relay/docs/thinking-in-graphql.html

Gitflow Workflow. (z.d.). Geraadpleegd van https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

GitLab. (z.d.). Geraadpleegd van https://about.gitlab.com/

Nuclide. (z.d.). Geraadpleegd van http://nuclide.io/

What is npm. (2016, 3 maart). Geraadpleegd van https://docs.npmjs.com/getting-started/what-is-npm

Near Field Communication. (z.d.). Geraadpleegd van
http://developer.android.com/guide/topics/connectivity/nfc/index.html

Google Sign-In for Websites. (z.d.). Geraadpleegd van
https://developers.google.com/identity/sign-in/web/devconsole-project

Bijlagen

Bijlage A Gebruikte tools en technieken

Tool/Techniek	Beschrijving	Link
SCRUM	De gebruikte agile ontwikkelingsmethodiek	http://www.scrumguides.org/
MoSCoW	Manier om prioriteiten op te stellen	https://nl.wikipedia.org/wiki/MoSCoW-methode
JavaScript	De programmeertaal die gebruikt is om de applicaties te maken	https://www.javascript.com/resources
React Native	Framework om native apps te bouwen m.b.v. JavaScript en React, dit is gebruikt voor de Android applicatie	https://facebook.github.io/react-native/
React	Een JavaScript library voor het bouwen van UI, dit is gebruikt voor de Android applicatie en de webapplicatie	https://facebook.github.io/react/
Git	Versiebeheer systeem voor o.a. source code	https://git-scm.com/
Gitflow	Een branching model voor het werken met Git	https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
Watchman	Kijkt of er files verandert zijn en kan in dat geval vervolgstappen nemen, gebruikt met React Native	https://facebook.github.io/watchman/
Flow	Kan fouten vinden in JavaScript code vinden en deze vroegtijdig aan de ontwikkelaar bekend maken	http://flowtype.org/
Nuclide	IDE dat gebruikt is om te ontwikkelen voor React Native	http://nuclide.io/
Jest	Hiermee kunnen JavaScript unit tests opgesteld worden	https://facebook.github.io/jest/
Continuous integration	Hiermee kan o.a. als er telkens bij een push naar de repository getest worden of alles nog werkt	https://en.wikipedia.org/wiki/Continuous_integration
React developer tools	Browser extensie waarmee je React elementen kunt inspecteren	https://github.com/facebook/react-devtools
Dia	Tekenprogramma om gestructureerde diagrammen te tekenen	https://en.wikipedia.org/wiki/Dia_(software)
MySQL	Het gebruikte database management systeem	https://www.mysql.com/
REST	Architectuur stijl dat gebruik maakt van http om tussen client-server te communiceren	http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
Netbeans	IDE gebruikt om een RESTful web service te genereren vanuit de MySQL database	https://netbeans.org/
JAX-RS	Java API for RESTful Web Services, dit is de API die is gebruikt bij het genereren van de web service	https://jax-rs-spec.java.net/
JavaMail API	Framework voor het bouwen van mail applicaties	http://www.oracle.com/technetwork/java/javamail/index.html

Bijlage B Product backlog

Android app				
ID	Als een ...	Wil ik ...	Zodat ...	Prio
A0	gebruiker	met mijn RFID tag in- en uitklokken	ik niet meer zelf handmatig bij hoef te houden hoeveel uur ik heb gewerkt	Must have
A1	manager	dat medewerkers voor een bepaald project in- en uitklokken	per project bijgehouden kan worden hoeveel uur er aan gewerkt is	Must have
A2	gebruiker	een indicatie zien dat ik kan in- en uitklokken	ik weet wanneer ik mijn tag kan scannen	Must have
A3	fulltimer	bij het inklokken zien hoeveel uur ik nog gemiddeld per dag moet werken	ik weet of ik nog op schema loop	Must have
A4	gebruiker	bij het uitklokken zien hoeveel uur ik gewerkt heb	ik weet of ik genoeg gewerkt heb	Must have
A5	gebruiker	zo min mogelijk handelingen doen	ik niet veel tijd kwijt raak aan het registreren van uren	Must have
A6	gebruiker	bij het in- en uitklokken een inspirerende citaat zien	ik geïnspireerd raak	Must have
A7	gebruiker	een vrij aanvraag kunnen doen of ziekmelden	dit bij de manager bekend wordt	Should have
A8	gebruiker	meerdere keren per dag voor hetzelfde project kunnen inklokken	als ik tussendoor voor een ander project een werkzaamheid moet uitvoeren, ik weer kan inklokken voor het project waar ik eerst mee bezig was	Must have
A9	manager	dat de tijd van de server wordt gebruikt voor het in- en uitklokken	als er iets mis gaat met de tijd van het device, deze tijden nog kloppen	Must have
A10	gebruiker	een foutmelding zien als er iets misgaat in de applicatie, zodat	ik weet wat er fout is gegaan en ik dit door kan geven of kan corrigeren	Must have
Web app				
ID	Als een ...	Wil ik ...	Zodat ...	Prio
W0	manager	projecten kunnen aanmaken	ik deze kan koppelen aan werknemers	Must have
W1	manager	projecten kunnen wijzigen	de projecten up-to-date gehouden kunnen worden	Must have
W2	manager	projecten kunnen verwijderen	geannuleerde projecten niet meer te zien zijn	Must have
W3	manager	werknemers kunnen aanmaken	ik deze kan koppelen aan projecten	Must have
W4	manager	werknemers kunnen wijzigen	de werknemers up-to-date gehouden kunnen worden	Must have
W5	manager	werknemers kunnen verwijderen	ik geen data meer zie van oud werknemers	Must have
W6	manager	werknemers aan projecten kunnen koppelen	ik kan zien hoeveel uur er is gewerkt per werknemer per project	Must have
W7	manager	de totale gewerkte uren en de uren per werknemer op jaar, maand, week en dag basis per project in een grafiek zien	ik hier overzicht over heb	Must have
W8	gebruiker	een overzicht van mijn gewerkte uren kunnen zien	ik dit kan controleren	Must have
W9	gebruiker	met google kunnen inloggen	ik mijn bedrijfsemail kan gebruiken	Must have
W10	manager	aan het eind van elke maand een mail ontvangen met de gewerkte uren en WBSO uren per werknemer	ik dit naar de accountant kan sturen	Should have
W11	manager	onderscheid kunnen maken tussen parttimers en fulltimers	ik bij de fulltimers kan aangeven hoeveel contracturen zij werken	Must have
W12	manager	een rol kunnen aangeven bij het aanmaken van werknemers	op basis hiervan de juiste pagina geladen kan worden	Must have
W13	gebruiker	met google kunnen inloggen	ik mijn bedrijfsemail kan gebruiken	Must have
W14	manager	de koppeling tussen een werknemer er project verwijderen	als ik toch besluit dat de werknemer beter niet aan het project kan werken, ik dit kan corrigeren	Must have
W 15	gebruiker	mijn gewerkte uren in de huidige salarisperiode kunnen aanpassen	dit kan corrigeren als ik vergeten ben in te klokken, of voor het verkeerde project ingeklokt heb	Must have

Bijlage C Commands voor het opzetten van React Native voor Linux

Hier worden de commands die in de terminal uitgevoerd zijn opgesomd, de commands worden door een $ aangeduid.

Stappen opzetten react-native linux:

- Git installeren door in terminal te typen: $ sudo apt-get install git
- Node.js 4.0 of hoger en React Native installeren door:
  - $ git clone https://github.com/creationix/nvm.git ~/.nvm && cd ~/.nvm && git checkout `git describe --abbrev=0 --tags`
  - $. ~/.nvm/nvm.sh
  - $ nvm install 5.0
  - $ nvm use 5.0
  - $ npm install -g npm@2
  - $ npm install -g react-native-cli
- Watchman installeren om node file watching bugs te voorkomen:
  - $ git clone https://github.com/facebook/watchman.git
  - $ sudo apt-get install python-dev
  - $ sudo apt-get install automake
  - $ sudo apt-get install autoconf
  - $ cd watchman
  - $ ./autogen.sh
  - $ ./configure
  - $ make
  - $ sudo make install
- Java JDK 7.0 of hoger
  - zelf installeren. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - of de volgende commands in terminal
    - $ sudo add-apt-repository ppa:webupd8team/java
    - $ sudo apt-get update
    - $ sudo apt-get install oracle-java8-installer
- Android SDK tools installeren(downloaden van android site en extracten in directory van waar je het wil runnen).http://developer.android.com/sdk/index.html#Other
- ANDROID_HOME variabele naar Android SDK laten wijzen:
  - in terminal type $ gedit ~/.bashrc en voeg de volgende line toe:
  - export ANDROID_HOME=/home/username/Documents/android-sdk-linux
- naar de tools directory van je ge-extracte android sdk, vervolgens om de sdk manager te starten in terminal ./android sdk
- de volgende onderdelen instaleren:
  - Android SDK Build-tools version 23.0.1,
  - Android 6.0 (API 23)
  - Android Support Repository
- react developer tools installeren voor chrome/firefox.

Google emulator:

- virtual machine acceleration voor snellere emulator:
  - check of je processor acceleration support met
  - $ egrep -c '(vmx|svm)' /proc/cpuinfo
  - als er een waarde van 1 of hoger uitkomt, kvm-ok typen
    - als de output is dat kvm-ok niet geinstalleerd is, $ sudo apt-get install cpu-checker
    - vervolgens weer kvm-ok in de terminal uitvoeren en checken of de output is: INFO: /dev/kvm exists KVM acceleration can be used
    - kvm installeren $ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils
- het volgende command is nodig om de emulator te starten en mergedebugresource fout op te lossen $ sudo apt-get install lib32stdc++6 lib32z1 lib32z1-dev
- om de emulator met acceleration te runnen $ emulator -avd <avd_name> -qemu -m 512 -enable-kvm
- om normaal te starten ./android avd in de tools directory

Project aanmaken:

$ react-native init AwesomeProject
$ cd AwesomeProject
$ react-native start <- om een lokale server te starten om de apps te runnen.
$ react-native run-android <- om het project te runnen op emulator

- Open index.android.js om de app aan te passen.
- menu button en Reload JS drukken om verandering te zien, of Enable Live reload om veranderingen direct te zien.
- Run $ adb logcat *:S ReactNative:V ReactNativeJS:V in terminal om de log van je app te zien.

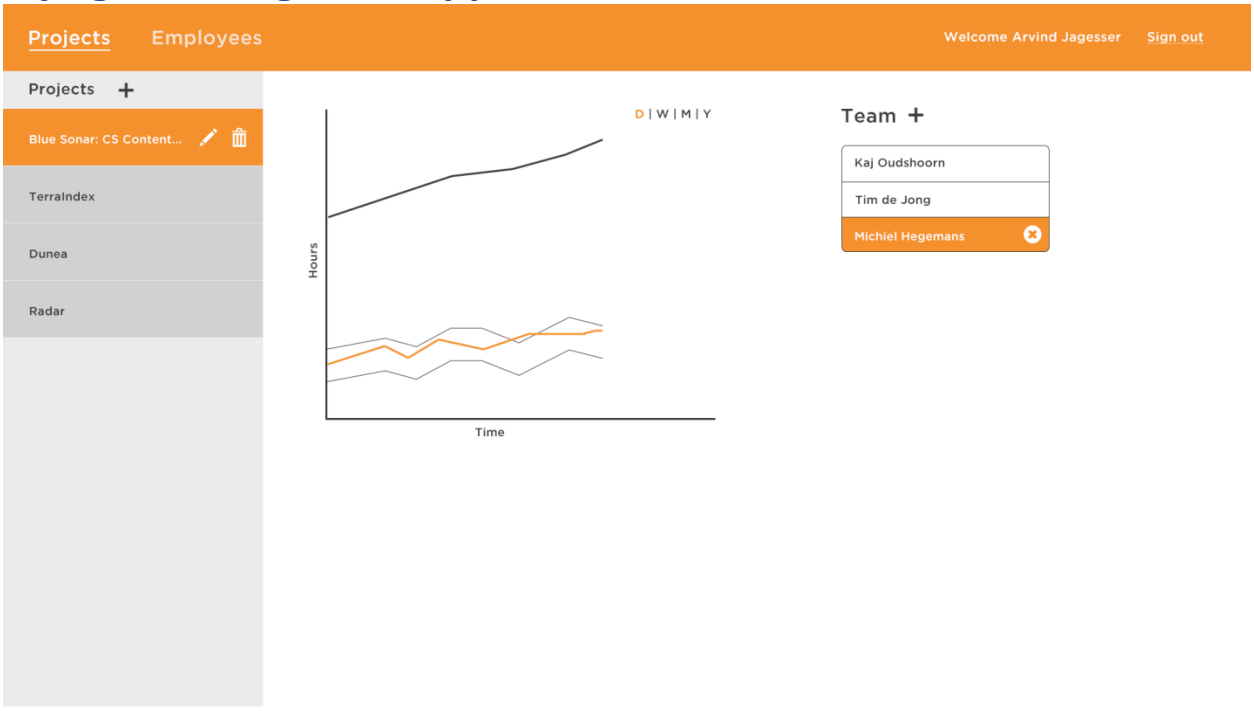
Flow als typechecker

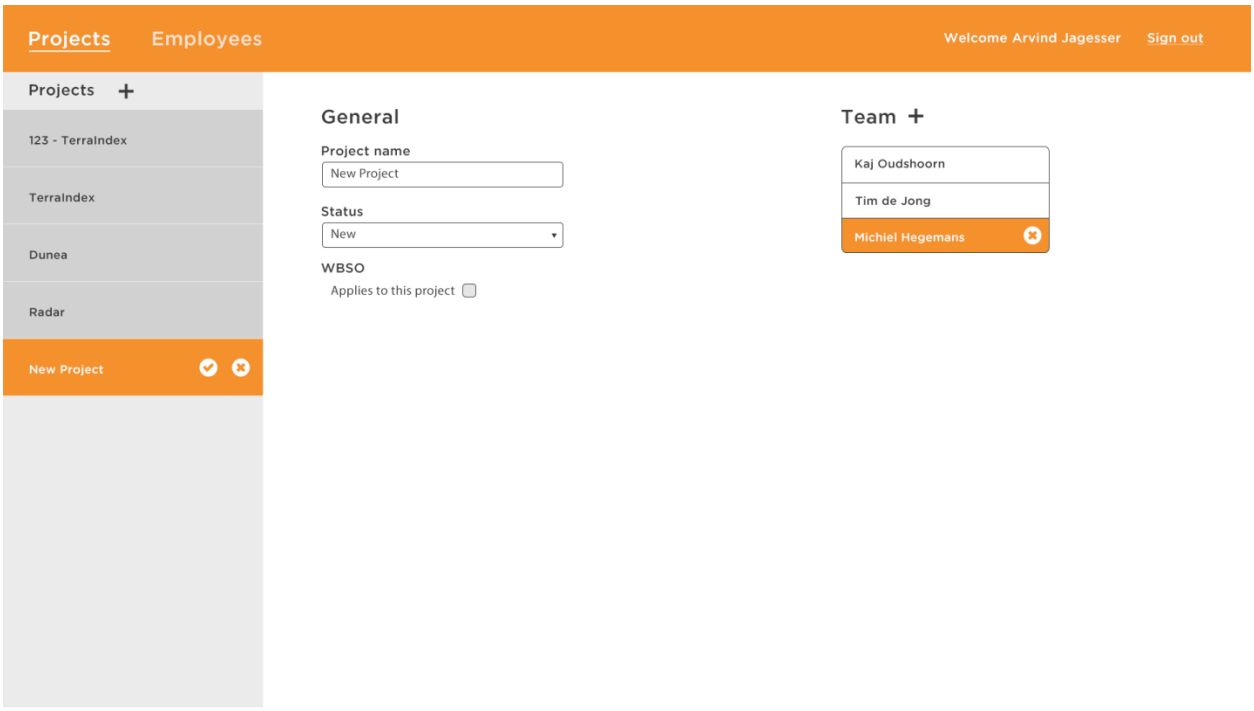
- Flow downloaden van: http://flowtype.org/docs/getting-started.html#_
- Unzippen en naar de flow directory navigeren.
- $ echo -e "\nPATH=\"\$PATH:$(pwd)/\"" >> ~/.bashrc && source ~/.bashrc <- zodat je flow overal kan runnen

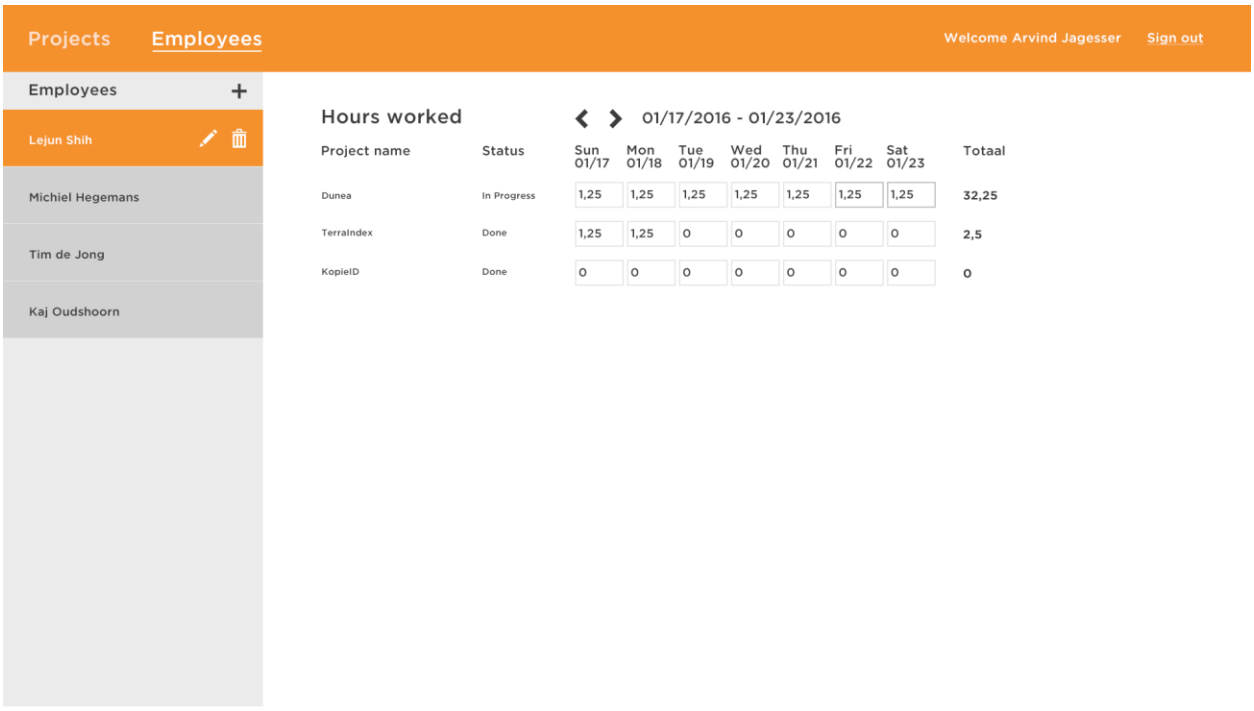
Nuclide als IDE(flow support)

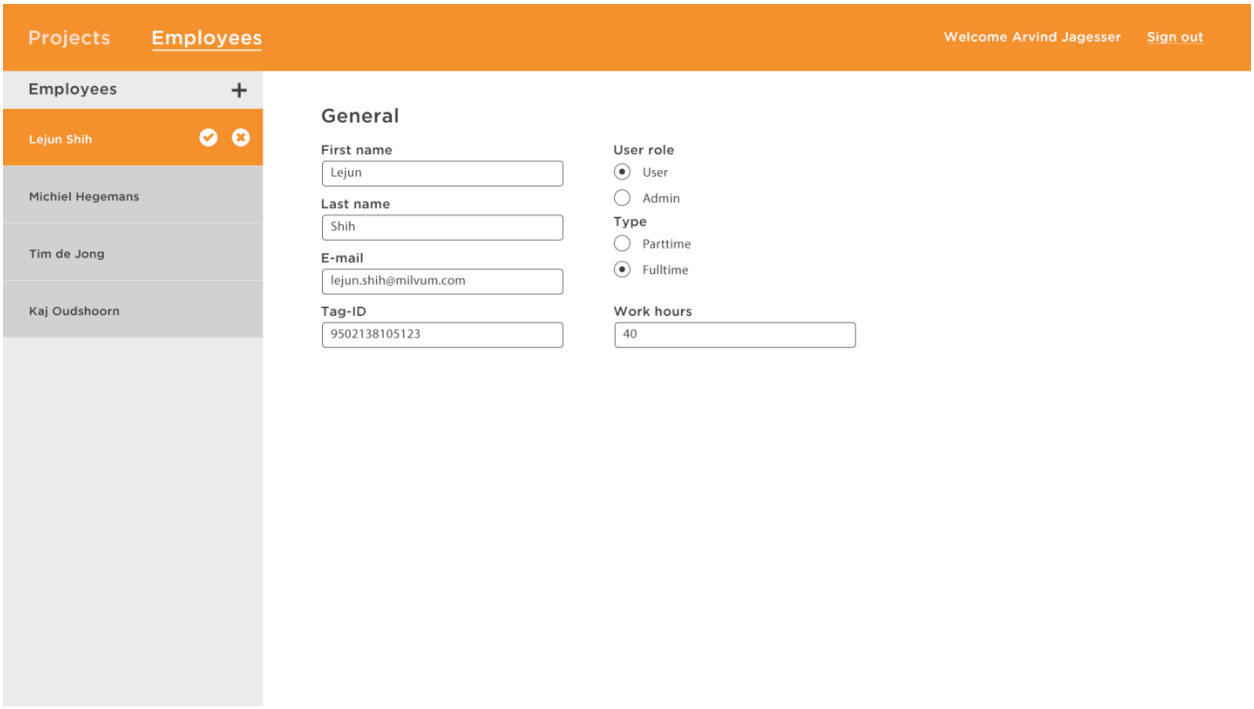
- Atom deb downloaden: https://github.com/atom/atom/releases/download/v1.2.4/atom-amd64.deb
- $ sudo apt-get install alien dpkg-dev debhelper build-essential
- $ sudo dpkg -i packagename.deb
- $ atom <- om atom te starten
- in atom de nuclide-installer package instaleren en evt andere packages naar voorkeur
- open de project folder om te beginnen met editten

Bijlage D Design webapplicatie






Projects
Employees
Welcome Arvind Jagesser
Sign out
Employees
Lejun Shih
Michiel Hegemans
Tim de Jong
Kaj Oudshoorn
Hours worked
01/17/2016 - 01/23/2016
Project name
Status
Sun 01/17
Mon 01/18
Tue 01/19
Wed 01/20
Thu 01/21
Fri 01/22
Sat 01/23
Totaal
Dunea
In Progress
1,25 1,25 1,25 1,25 1,25 1,25 1,25
32,25
TerraIndex
Done
1,25 1,25 0 0 0 0 0
2,5
KopieID
Done
0 0 0 0 0 0 0
0


Projects
Employees
Welcome Arvind Jagesser
Sign out
Employees
Lejun Shih
Michiel Hegemans
Tim de Jong
Kaj Oudshoorn
General
First name
Lejun
Last name
Shih
E-mail
lejun.shih@milvum.com
Tag-ID
9502138105123
User role
User
Admin
Type
Parttime
Fulltime
Work hours
40

Bijlage E Uitgevoerde tests

Sprint 1		
Passed	**User Story/ Case**	**Check**
Yes	A1	Welkomstbericht is te zien met naam
Yes	A1	Projecten zijn te zien in de lijst
Yes	A1	Project kan geselecteerd worden
Yes	A1	Er kan niet naar een volgend scherm genavigeerd worden zonder een project te selecteren
Yes	A1	Er word naar het inklokscherm genavigeerd als er een project geselecteerd word
Yes	A2	Er is een indicatie te zien
Yes	A3	Er is een indicatie te zien dat er is ingeklokt
Yes	A6	Er is een citaat te zien op het inklok scherm
Yes	A6	Het citaat is willekeurig
Yes	A3	Het gemiddeld aantal uur per dag is te zien
Yes	A3	De berekening voor het gemiddeld aantal uur per dag klopt
Yes	WS	Werknemer kan opgehaald worden op id
Yes	WS	Timesheet kan opgehaald worden op werknemer
Yes	WS	Projecten kunnen opgehaald worden
Yes	WS	Random citaat kan opgehaald worden
Yes	WS	Timesheet kan toegevoegd worden
Yes	WS	Timesheet kan aangepast worden
Sprint 2		
Passed	**User Story/ Case**	**Check**
Yes	A4	Er is te zien hoeveel uur er gewerkt is
Yes	A4	De berekening voor aantal gewerkte uren klopt
Yes	A6	Er is een citaat te zien op het uitklokscherm
Yes	A6	Het citaat is willekeurig
Yes	A0	Tag wordt gedetecteerd
Yes	A0	TagID kan uitgelezen worden
Yes	A0	TagID word naar de javascript code gestuurd
Yes	A0	Werknemer die bij het tagID hoort wordt opgehaald
Yes	A0	Gegevens wanneer werknemer voor het laatst was ingeklokt(timesheet) worden opgehaald
Yes	A0	Als bij de opgehaalde timesheet een uitklok tijd staat wordt er genavigeerd naar de projectpagina
Yes	A0	Als bij de opgehaalde timesheet geen uitklok tijd staat wordt er uitgeklokt met de huidige tijd
Yes	A1	Projecten worden opgehaald
Yes	A0/A1	Bij het selecteren van een project wordt er ingeklokt(nieuwe timesheet aangemaakt)
Yes	A5	Na 5 seconden op het inklok scherm wordt er genavigeerd naar het scan scherm

Yes	A5	Na 5 seconden op het uitklok scherm wordt er genavigeerd naar het scan scherm
Yes	A5	Bij het scannen van een tag wordt er automatisch genavigeerd naar het volgende scherm
Sprint 3		
Passed	**User Story/ Case**	**Check**
Yes	A8	Er kan meerdere keren voor hetzelfde project ingeklokt worden op een dag
Yes	A9	De servertijd kan opgehaald worden
Yes	A9	De servertijd wordt gebruikt bij het in- en uitklokken
Yes	A10	Er wordt een foutmelding getoond als er geen connectie is
Yes	A10	Er wordt een foutmelding getoond als er gescanned wordt met een tag die niet in de database staat
Yes	W0	Er kan een project aangemaakt worden
Yes	W0	Er wordt een pop-up getoond als een project is aangemaakt
Yes	W3	Er kan een werknemer aangemaakt worden
Yes	W3	Er wordt een pop-up getoond als een werknemer is aangemaakt
Sprint 4		
Passed	**User Story/ Case**	**Check**
Yes	W11	Er kan een werknemer aangemaakt worden met type
Yes	W12	Er kan een werknemer aangemaakt worden met rol
Yes	A3	Gemiddeld aantal uren te gaan per dag wordt getoond voor fulltimer bij het inklokken
Yes	A3	Bij parttimer wordt geen gemiddeld aantal uur te gaan per dag getoond bij het inklokken
Yes	W13	Er kan ingelogd worden met het bedrijfsemail via Google sign-in
Yes	W13	Er kan uitgelogd worden
Yes	W8	De gewerkte uren in een week kunnen opgehaald worden
Yes	W8	De gewerkte uren in een week worden in de tabel getoond op de juiste plek bij het project
Yes	W8	Er kan gebladerd worden tussen weken, de juiste data wordt opgehaald en op dejuiste plek gezet
Yes	W1	Projecten kunnen aangepast worden
Yes	W2	Projecten kunnen verwijderd worden
Yes	W4	Werknemers kunnen aangepast worden
Yes	W5	Werknemers kunnen verwijderd worden

Yes	W14	Werknemers kunnen aan projecten gekoppeld worden
Yes	W15	De koppeling tussen een werknemer en project kan verwijderd worden
Sprint 5		
Passed	**User Story/ Case**	**Check**
Yes	W8	Normale werknemers zien alleen de uren tabel als ze inloggen
Yes	W15	De uren in de tabel kunnen alleen in de huidige salarisperiode aangepast worden
Yes	W7	De gegevens zijn in de grafiek per werknemer te zien en het totale aantal voor een project, er kan terug of verder gebladerd worden per jaar maand of week
Yes	W10	Er wordt een mail gestuurd
Yes	W10	De juiste gegevens staan in de mail
Yes	Design/login	Op het inlogscherm is een het logo, een citaat en de google sing-in knop te zien
Yes	Design/projects/employees	Er is een Dashboard te zien met projects en employees, de naam van de ingelogde werknemer en een sign-out button
Yes	Design/projects	Op het projecten scherm is een projectenlijst te zien
Yes	Design/projects	Als op een project uit de projectenlijst geklikt wordt, verandert het aangeklikte project van design(wordt oranje met witte text en een edit/remove button) en wordt de grafiek voor dat project getoond
Yes	Design/projects	Als er bij de grafiek op jaar wordt getoond, wordt de grafiek op jaarbasis getoond, maand op maandbasis en week op weekbasis
Yes	Design/projects	Als er bij de grafiek op Y wordt geklikt, wordt de grafiek op jaarbasis getoond, M op maandbasis en W op weekbasis
Yes	Design/projects	Als een project wordt geselecteerd wordt het team getoond
Yes	Design/projects	Als team member wordt aangeklikt wordt de lijn in de grafiek van de aangeklikte werknemer oranje en komt er een x bij de werknemer te staan om te ontkoppelen
Yes	Design/projects	Als er op de + geklikt wordt bij Team komt er een pop-up met een werknemerslijst
Yes	Design/projects	Als er op de edit knop bij een geselecteerd project geklikt of plusteken bij projects worden de invoervelden getoond voor het aanmaken of aanpassen van een project en veranderd de edit/remove knop in confirm/cancel

Yes	Design/employees	Op het employees scherm is een werknemerslijst te zien
Yes	Design/employees	Als op een werknnemer uit de werknemers lijst geklikt wordt, verandert de aamgeklikte werknemer van design(wordt oranje met witte text en een edit/remove button) krijg je de tabel van de aangeklikte werknemer te zien
Yes	Design/employees	Als er bij de tabel geklikt wordt op een cel binnen de huidige salarisperiode verschijnt er een pop-up om uren te wijzigen
Yes	Design/employees	Als er op de edit knop bij een geselecteerde werknemer geklikt of plusteken bij employees worden de invoervelden getoond voor het aanmaken of aanpassen van een werknemer en veranderd de edit/remove knop in confirm/cancel
Yes	Design/projects	Als een project wordt geselecteerd wordt het team getoond
Yes	Design/projects	Als team member wordt aangeklikt wordt de lijn in de grafiek van de aangeklikte werknemer oranje en komt er een x bij de werknemer te staan om te ontkoppelen
Yes	Design/projects	Als er op de + geklikt wordt bij Team komt er een pop-up met een werknemerslijst

Bijlage F Goedgekeurd afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2015-2.2 (start uiterlijk 16 november 2015)
Startdatum uitvoering afstudeeropdracht: 16 november 2015
Inleverdatum afstudeerdossier volgens jaarrooster: 21 maart 2016

Studentnummer: 11116730
Achternaam: dhr Ben Dahman
Voorletters: K.
Roepnaam: Karim
Adres: Minister Talmalaan 141
Postcode: 2285EE
Woonplaats: Rijswijk
Telefoonnummer: -
Mobiel nummer: 0614574173
Privé emailadres: karim_bdm@hotmail.com

Opleiding: Informatica
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: G.M. Tuk
Naam begeleidend examinator: M. Maris
Naam tweede examinator: G.A. Mijnarends

Naam bedrijf: Milvum
Afdeling bedrijf: Ontwikkelteam
Bezoekadres bedrijf: Regulusweg 5
Postcode bezoekadres: 2516AC
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Den Haag
Telefoon bedrijf: +31 (0)70 – 20 55 711
Telefax bedrijf: -
Internetsite bedrijf: www.milvum.com

Achternaam opdrachtgever: dhr Jagesser
Voorletters opdrachtgever: V.
Titulatuur opdrachtgever: Dhr. BSc
Functie opdrachtgever: Managing Partner
Doorkiesnummer opdrachtgever: -
Email opdrachtgever: viresh@milvum.com

Achternaam bedrijfsmentor: dhr Jagesser
Voorletters bedrijfsmentor: A.
Titulatuur bedrijfsmentor: Dhr. BSc
Functie bedrijfsmentor: Managing Partner
Doorkiesnummer bedrijfsmentor: -
Email bedrijfsmentor: arvind@milvum.com

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Ontwikkelen van een huizencheck applicatie bij Milvum.

Opdrachtomschrijving

1. Bedrijf

Milvum is een commercieel bedrijf dat zich richt op het ontwikkelen van Business-to-Business(B2B), Business-to-Consumer(B2C) en Business-to-Employee(B2E) software. Het bedrijf bestaat uit 12 personen en zit in een fase van groei. De organisatiestructuur van het bedrijf is horizontaal.

Voorbeelden van klanten en applicaties:

- Voor het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties is een app(KopieID) ontwikkeld die het veilig maakt om identiteitsbewijzen te delen.

- Voor het tv-programma Radar zijn apps ontwikkeld die de samenwerking tussen de redactie en consumenten versoepeld. De redactie kan co-creatie stimuleren door oproepen naar consumenten sturen met behulp van push notificaties. Vervolgens kunnen de consumenten de oproep beantwoorden door ervaringen met producten of diensten via een applicatie met de redactie te delen. Deze ervaringen kunnen eventueel gebruikt worden voor een uitzending.

2. Probleemstelling

Het is moeilijk voor consumenten om achter de waarde van huizen te komen, vooral onderweg. De opdrachtgever wil dit consumentenprobleem oplossen. Op dit moment is er een app voor op de markt, de prijzen voor het aanvragen van informatie in de app zijn naar mening van de opdrachtgever echter aan de hoge kant.

3. Doelstelling van de afstudeeropdracht

Om het consumentenprobleem op te lossen moet er een applicatie ontwikkeld worden waarbij gebruikers in een oogopslag kunnen zien hoeveel woningen in de buurt waard zijn en door middel van een in-app aankoop de koophistorie van een woning bekijken.

4. Resultaat

Een releasebare iOS App, waarin je in een oogopslag kunt zien wat de waarde van woningen om je heen is en met een in-app aankoop de koophistorie van woningen kan bekijken.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

De softwareontwikkelmethode die gebruikt zal worden is SCRUM. De lengte van de sprints is vastgesteld op 2 weken.

Uit te voeren werkzaamheid	Aantal werkdagen
Plan van aanpak schrijven	4
User stories vaststellen	7
Bekend worden met het react-native framework	5

• Kennis opdoen over JavaScript/css	
Kennis opdoen over GraphQL en Relay	5
Ontwerpen van de database	4
Bouwen van de database	3
Ontwerpen van de applicatie	7
Bouwen van de applicatie	26
Testen	5
Implementatie en overdracht	4
Opbouwen afstudeerdossier	15

6. Op te leveren (tussen)producten

Rapport met gebruikte tools/koppelingen
Rapport met user stories(product backlog)
Klassendiagram van de applicatie
Flow diagram
ER diagram van de database.
De iOS applicatie.

7. Te demonstreren competenties en wijze waarop

Uitvoeren analyse door definitie requirements(Niveau 3):
Het opstellen en bijhouden van de requirements, de requirements zullen opgesteld worden door stakeholders te interviewen. Het gaat om meerdere stakeholders en de requirements worden in de loop van het project aangevuld en kunnen veranderen.

Ontwerpen systeemdeel(Niveau 3):
Het gaat om het ontwerpen van een object georiënteerde applicatie, waarbij rekening gehouden moet worden met wijzigingen en hergebruik. Daarnaast zal er een database ontworpen en gebouwd moeten worden.

Bouwen applicatie(Niveau 4):
Het maken van de huizencheck applicatie voor iOS. Bij het maken van de applicatie moet rekening worden gehouden met hergebruik en uitbreidbaarheid, zodat in de toekomst meer features toegevoegd kunnen worden en de applicatie ook voor Android gemaakt kan worden.

Uitvoeren van en rapporteren over het testproces(Niveau 3):
Er zullen unit tests opgesteld worden om de functionaliteit van de applicatie herhaaldelijk te kunnen testen.

Bijlage G Aangepast afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2015-2.2 (start uiterlijk 16 november 2015)
Startdatum uitvoering afstudeeropdracht: 16 november 2015
Inleverdatum afstudeerdossier volgens jaarrooster: 21 maart 2016

Studentnummer: 11116730
Achternaam: dhr Ben Dahman
Voorletters: K.
Roepnaam: Karim
Adres: Minister Talmalaan 141
Postcode: 2285EE
Woonplaats: Rijswijk
Telefoonnummer: -
Mobiel nummer: 0614574173
Privé emailadres: karim_bdm@hotmail.com

Opleiding: Informatica
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: G.M. Tuk
Naam begeleidend examinator: M. Maris
Naam tweede examinator: G.A. Mijnarends

Naam bedrijf: Milvum
Afdeling bedrijf: Ontwikkelteam
Bezoekadres bedrijf: Regulusweg 5
Postcode bezoekadres: 2516AC
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Den Haag
Telefoon bedrijf: +31 (0)70 – 20 55 711
Telefax bedrijf: -
Internetsite bedrijf: www.milvum.com

Achternaam opdrachtgever: dhr Jagesser
Voorletters opdrachtgever: V.
Titulatuur opdrachtgever: Dhr. BSc
Functie opdrachtgever: Managing Partner
Doorkiesnummer opdrachtgever: -
Email opdrachtgever: viresh@milvum.com

Achternaam bedrijfsmentor: dhr Jagesser
Voorletters bedrijfsmentor: A.
Titulatuur bedrijfsmentor: Dhr. BSc
Functie bedrijfsmentor: Managing Partner
Doorkiesnummer bedrijfsmentor: -
Email bedrijfsmentor: arvind@milvum.com

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Ontwikkelen van een urenregistratie systeem bij Milvum.

Opdrachtomschrijving

1. Bedrijf

Milvum is een commercieel bedrijf dat zich richt op het ontwikkelen van Business-to-Business(B2B), Business-to-Consumer(B2C) en Business-to-Employee(B2E) software. Het bedrijf bestaat uit 12 personen en zit in een fase van groei. De organisatiestructuur van het bedrijf is horizontaal.

Voorbeelden van klanten en applicaties:

- Voor het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties is een app(KopieID) ontwikkeld die het veilig maakt om identiteitsbewijzen te delen.

- Voor het tv-programma Radar zijn apps ontwikkeld die de samenwerking tussen de redactie en consumenten versoepeld. De redactie kan co-creatie stimuleren door oproepen naar consumenten sturen met behulp van push notificaties. Vervolgens kunnen de consumenten de oproep beantwoorden door ervaringen met producten of diensten via een applicatie met de redactie te delen. Deze ervaringen kunnen eventueel gebruikt worden voor een uitzending.

2. Probleemstelling

Op het moment wordt het registreren van gewerkte uren bij milvum handmatig bijgehouden in Google Spreadsheets. Dit zorgt ervoor dat er uren vergeten worden en soms fouten worden gemaakt. Ook is het lastig om een goed overzicht te behouden door het vele aantal spreadsheets. De opdrachtgever vindt de oplossingen van de markt te omslachtig en complex.

3. Doelstelling van de afstudeeropdracht

Om de foutgevoeligheid van het handmatige urenregistratie systeem weg te nemen wil Milvum een geautomatisseerd urenregistratie systeem. In dit systeem kunnen medewerkers in- en uitklokken op een Android toestel door middel van een badge met RFID tag. De geregistreerde uren moeten ingezien kunnen worden in een webapplicatie.

4. Resultaat

Een urenregistratie systeem bestaande uit een Android applicatie voor het in- en uitklokken en een webapplicatie voor het inzien van de uren.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

De softwareontwikkelmethode die gebruikt zal worden is SCRUM. De lengte van de sprints is vastgesteld op 2 weken.

Uit te voeren werkzaamheid	Aantal werkdagen
Plan van aanpak schrijven	4
User stories vaststellen	7

Bekend worden met het react-native framework • Kennis opdoen over JavaScript/css	5
Kennis opdoen over GraphQL en Relay	5
Ontwerpen van de database	4
Bouwen van de database	3
Ontwerpen van de applicatie	7
Bouwen van de applicatie	26
Testen	5
Implementatie en overdracht	4
Opbouwen afstudeerdossier	15

6. Op te leveren (tussen)producten

Rapport met gebruikte tools/koppelingen
Rapport met user stories(product backlog)
Klassendiagram van de applicatie
Flow diagram
ER diagram van de database.
De Android applicatie.
De webapplicatie.

7. Te demonstreren competenties en wijze waarop

Uitvoeren analyse door definitie requirements(Niveau 3):
Het opstellen en bijhouden van de requirements, de requirements zullen opgesteld worden door stakeholders te interviewen. Het gaat om meerdere stakeholders en de requirements worden in de loop van het project aangevuld en kunnen veranderen.

Ontwerpen systeemdeel(Niveau 3):
Het gaat om het ontwerpen van een object georiënteerde applicatie, waarbij rekening gehouden moet worden met wijzigingen en hergebruik. Daarnaast zal er een database ontworpen en gebouwd moeten worden.

Bouwen applicatie(Niveau 4):
Het maken van het urenregistratie systeem voor Android inclusief webapplicatie. Bij het maken van het systeem moet rekening worden gehouden met hergebruik en uitbreidbaarheid, zodat in de toekomst meer features toegevoegd kunnen worden.

Uitvoeren van en rapporteren over het testproces(Niveau 3):
Er zullen unit tests opgesteld worden om de functionaliteit van de applicatie herhaaldelijk te kunnen testen.

Bijlage H Beoordeling tussentijds assessment

Bespreking concept	**Tussentijds assessment**	Eerste beoordeling

Formulier tussentijds assessment

Student: Karim Ben Dahman **Studentnummer:** 11116730

Datum: 29-02-2016 **eerste / tweede TTA:** Eerste

Tijdens het tussentijds assessment is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	X	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	X	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	

Aanpak	**O**	**T**	**V**	**G**
Passend		x		
Theoretisch verantwoord		x		
Samenhang uitvoering beroepstaken	x			

Beroepstaken op afgesproken niveau uitgevoerd?		**O**	**T**	**V**	**G**
1	*Uitvoeren analyse door definitie requirements(Niveau 3)*		x		
2	*Ontwerpen systeemdeel(Niveau 3)*		x		

3	*Bouwen applicatie(Niveau 4)*		x		
4	*Uitvoeren van en rapporteren over het testproces(Niveau 3)*	*X*			

Producten	**O**	**T**	**V**	**G**
Tussenproducten		x		
Eindproducten		x		

Effectief communiceren	**O**	**T**	**V**	**G**
Binnen afstudeerbedrijf			X	
Afstudeerdossier	X			

Reflectie	**O**	**T**	**V**	**G**
Inzicht in eigen functioneren		X		
Inzicht in eigen leerproces		X		

<u>Toelichting per beoordelingscriterium</u>

Aanpak
Geen goede verslagstructuur, te matig het proces en de resultaten beschreven in het rapport. Werkwijze van SCRUM kom je niet goed tegen in de beschrijving van de sprints

Beroepstaken op afgesproken niveau uitgevoerd?
Uitvoeren analyse door definitie requirements(Niveau 3) Er zijn geen duidelijke requirements op dit moment.
Ontwerpen systeemdeel(Niveau 3) Het ontwerp ontbreekt grotendeels.

Bouwen applicatie(Niveau 4) Applicatie is wel gebouwd doch het ontwikkelproces is matig beschreven.
Uitvoeren van en rapporteren over het testproces(Niveau 3) Geen tests gerapporteerd.

Producten
De bijlage dienen geven de indruk dat er substantieel werk verricht is. Echter is de status van de scriptie zeer matig en geeft geen goed beeld van het gedane afstudeerwerk. Ontwerp- en Testrapport ontbreken. Mondeling geeft de student aan dat er werkende app is. Schriftelijk is een groot deel nog niet beschreven.

Effectief communiceren
De student luistert goed naar gegeven feedback en probeert deze te integreren in zijn werk. Het afstudeerverslag is nog niet af en beschrijft veel punten onvoldoende of te oppervlakkig. Hierdoor zijn het eindproduct en de beroepstaken eigenlijk niet goed te beoordelen.

Reflectie
De student toont aan dat hij inzicht heeft in zijn eigen functioneren en zijn leerproces. De product- en procesvaluatie beschrijven nog niet alle relevante punten, die de student mondeling wel noemt.

Advies

v	Inleveren **(bindend advies)**
	Verlengen **(vrijblijvend advies)**
	Stoppen **(vrijblijvend advies)**

Besluit student

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

v	**Afstudeerdossier wordt op afgesproken datum ingeleverd**	Inleverdatum: 21-3-2016
	Afstudeerperiode wordt verlengd	Inleverdatum:
	Student stopt met afstudeeropdracht	

Naam begeleidend examinator:	**M. Maris**
Naam tweede examinator:	**G. Mijnarends**
Datum:	**29-02-2016**

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

Bijlage I Plan van aanpak

Bijlage J Ontwerpdocument