Professionele Bachelor Toegepaste Informatica

Alternatieven voor de Google Maps API

Jasper Heeren

Promotoren:

Frank Wynants
GPS nv
Carine Derkoningen
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Alternatieven voor de Google Maps API

Jasper Heeren

Promotoren:

Frank Wynants
Carine Derkoningen

GPS nv
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

De laatste rechte lijn richting mijn diploma is ingezet. Na 3 fijne en leerrijke jaren op de PXL is het goed geweest. Maar alvorens ik de deur op PXL achter mij dicht kan gooien moet ik eerst dit eindwerk tot een goed einde brengen. Het feit dat ik dit eindwerk kan schrijven in mijn laatste jaar is al een grote prestatie. Ik zou nooit zo ver zijn gekomen zonder de hulp van bepaalde mensen, daarom wil graag enkele personen bedanken.

Eerst en vooral zou ik de firma GPS willen bedanken voor de fijne stage en werkomgeving die ze me hebben aangeboden. Wekelijks werd ons werk overlopen en werden we in de goede richting gestuurd. Ik kon altijd terecht bij mijn bedrijfspromotor Frank Wynants of één van de andere werknemers. Nick Gaens en Ingo Schelfhout in het bijzonder.

Ten tweede een hartelijke dank aan mijn hogeschoolpromotor Carine Derkoningen voor de fijne samenwerking. Wanneer ik een vraag had over het eindwerk moest ik maar een mailtje sturen en ze had een hoop informatie voor me klaar. Ook het nakijken van mijn eindwerk op taal was voor haar geen moeite.

Ik mag ook zeker niet vergeten mijn stagepartner en junior-collega Yoran Nelissen te bedanken voor de goede verstandhouding. In het begin was alles een beetje zoeken maar uiteindelijk verliep de samenwerking zeer vlot.

Vervolgens een hele grote bedankt aan mijn ouders. Zonder hen was ik mijn eindwerk nu niet aan het schrijven. Ze hebben me gedurende mijn schoolloopbaan voortdurend gemotiveerd op momenten waarop ik het allemaal weer wat liet hangen. Ze waren altijd oprecht gelukkig als ik thuiskwam met goede punten.

Ten slotte wil ik mijn vriendin bedanken. Ik ken haar nog geen 3 jaar maar ze is wel een grote motivatie om het goed te doen op school zodat ik, wanneer ik mijn studies afrond, een job heb die ik graag doe en waarmee ik er onze toekomst samen goed kan laten uitzien.

Abstract

Een mobiele applicatie die locatiedata van medewerkers verwerkt op een kaart kan veel voordelen met zich meebrengen. De huidige mobiele applicatie van de firma GPS wordt hier als beginsituatie gebruikt.

In de huidige mobiele applicatie is het nog niet mogelijk om locatiedata van werknemers te verwerken, terwijl die data wel beschikbaar is. Hierdoor loopt de app heel wat handige functionaliteit mis. Indien de gebruiker zou kunnen zien op welke plaatsen zijn collega's zich registreren zou dit overzichtelijk zijn voor de gebruiker en het bedrijf. Een kaart biedt hiervoor de beste oplossing. Er zijn drie soorten registraties: een tijd-, job- of toegangsregistratie. Het moet visueel duidelijk zijn welk soort registratie een collega heeft uitgevoerd. Wanneer alle collega's overzichtelijk op een kaart staan aangegeven, moet het ook mogelijk zijn om navigatie naar een collega te starten aan de hand van een externe applicatie voor navigatie. Ten slotte zou het zeer interessant zijn indien de applicatie zou aangeven wanneer de gebruiker bij een klant aankomt, en hem vervolgens de vraag stelt of hij een tijd-, job- of toegangsregistratie wil uitvoeren.

Een eerste doelstelling is het voeren van een onderzoek naar een efficiënte manier om alle bovengenoemde kaartgerelateerde functionaliteiten te implementeren. Hierbij wordt er vooral gekeken naar de mogelijkheden van de Google Maps API die belangrijk zijn voor de verwerking van locatiegegevens binnen de applicatie. Er wordt een vergelijkende studie uitgevoerd met een tweede framework, namelijk 'Here'. Vervolgens wordt het onderzoek ook in de praktijk gebracht. Enkele cruciale verschillen tussen de twee frameworks worden blootgelegd. Finaal kan zo bepaald worden welke van de twee de beste oplossing biedt in deze context.

De tweede doelstelling is het ontwikkelen van een Androidapplicatie die de misgelopen functionaliteiten uit de tweede paragraaf implementeert aan de hand van de Google Maps API. Om een eenvoudig te bedienen scherm met kaart te ontwikkelen wordt er gebruikgemaakt van de volgende technologieën: Google Maps API, bestaande .NET services, Java.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

Tabel 1: Gebruikte afkortingen

API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
REST	Representational State Transfer
SDK	Software Development Kit

Lijst van gebruikte technische termen

Tabel 2: Gebruikte technische termen

Activity	Een activity vertegenwoordigt een enkel scherm met een gebruikersinterface.
Fragment	Een fragment wordt meestal gebruikt als onderdeel van de gebruikersinterface van een activity.
Geofencing	Geofencing is het virtueel afbakenen van een geografisch gebied of van specifieke locaties.
Master Branch	De hoofdtak van een GIT-bewaarplaats

Inleiding

Voor het product dat GPS zijn klanten aanbiedt is het noodzakelijk dat de locaties van alle medewerkers getoond worden op een kaart. Om de stageopdracht te kaderen is het allereerst nodig om GPS kort en bondig voor te stellen. Wat zijn de missie en visie van GPS? Ten tweede wordt de stageopdracht toegelicht. Het probleem en de doelstellingen van GPS worden besproken, en er wordt al eens geschat wat de tijdsplanning van dit project zal zijn. Om dit project te kunnen realiseren moet er gebruik gemaakt worden van diverse technologieën en afspraken. Deze worden ook kort overlopen zodat er een beter beeld van de werkwijze tijdens de stageopdracht gecreëerd wordt. Vervolgens wordt de stageopdracht uitgewerkt waarna het resultaat toegelicht kan worden. Ten slotte volgt er een reflectie over de stageopdracht. Wat ging er goed? Wat verliep niet zoals gepland, of waar werden er problemen ondervonden?

GPS wil dit project realiseren door gebruik te maken van de Google Maps API omdat dit een populaire oplossing is. Maar zijn er nog andere bruikbare alternatieven voor de Google Maps API? Of is er misschien een betere optie? Aan de hand van een onderzoekspaper wordt de Google Maps API op de proef gesteld en vergeleken met enkele alternatieven. Vervolgens wordt er een uitgebreidere vergelijking gedaan met één alternatief. Verwacht wordt dat de Google Maps API de beste oplossing zal zijn voor het bedrijf maar dat de Google Maps API niet op alle vlakken de beste oplossing biedt.

Er wordt geopend met een korte abstract van het project, zodat het duidelijk is hoe en waarom dit onderzoek gevoerd werd. Vervolgens wordt er een centrale onderzoeksvraag opgesteld, deze wordt opgesplitst in deelvragen. De onderzoeksmethode toont aan hoe dit onderzoek gevoerd zal moeten worden. Wat is er nodig om dit onderzoek te laten slagen? Een volgende stap is de literatuurstudie.

Er wordt een analyse gemaakt van de plus- en minpunten van de Google Maps API, waarna deze plus- en minpunten vergeleken worden met die van enkele alternatieven. Er zijn talloze alternatieven voor het tekenen van een kaart met enkele markers. We bakenen enkele alternatieven, die het dichtst in aanleunen bij de Google Maps API af aan de hand van een kleine literatuurstudie. Hierin worden de alternatieven op een globale schaal met de Google Maps API vergeleken. Wat zijn de cruciale verschillen tussen de twee? Wat kan de één beter dan de ander? Wat zijn de zwaktes van het alternatief vergeleken met de Google Maps API? Na een korte vergelijking van de alternatieven ten opzichte van de Google Maps API, wordt één alternatief gekozen voor verder onderzoek. Het gekozen alternatief wordt dan aan de hand van een uitgebreide literatuurstudie vergeleken met de Google Maps API.

De grootste succesfactoren van het alternatief, die relevant zijn in de context van het project van GPS worden vervolgens in de praktijk gebracht met zowel de Google Maps API als met het gekozen alternatief.

Om af te sluiten wordt er op basis van de bekomen onderzoeksresultaten een concrete conclusie geformuleerd.

I. Stageverslag

1 Bedrijfsvoorstelling

GPS is een bedrijf gelokaliseerd in Genk. Het werd dertig jaar geleden werd opgericht door Roger Lambie en is vervolgens uitgegroeid tot een rasecht familiebedrijf dat nu dertig werknemers telt. Het doel van GPS is om haar klanten een totaalpakket aan te bieden waarin tijdregistratie, personeelsplanning, toegangscontrole en camerabewaking naadloos geïntegreerd zijn.

1.1 Missie

Het is de missie van GPS om digitale en strategische humanresourcesoplossingen aan te bieden voor een optimale flexibiliteit in elke organisatie.

1.2 Visie

De langetermijnstrategie van GPS voorziet in het beheersen en toepassen van de nieuwste technologie. Via een community die door GPS is opgebouwd bevestigen ze innovatieve rol die ze willen spelen. GPS staat in nauw contact met elke stakeholder en vindt het van cruciaal belang dat haar klanten centraal staan.

2 Voorstelling stageopdracht

2.1 Probleemstelling

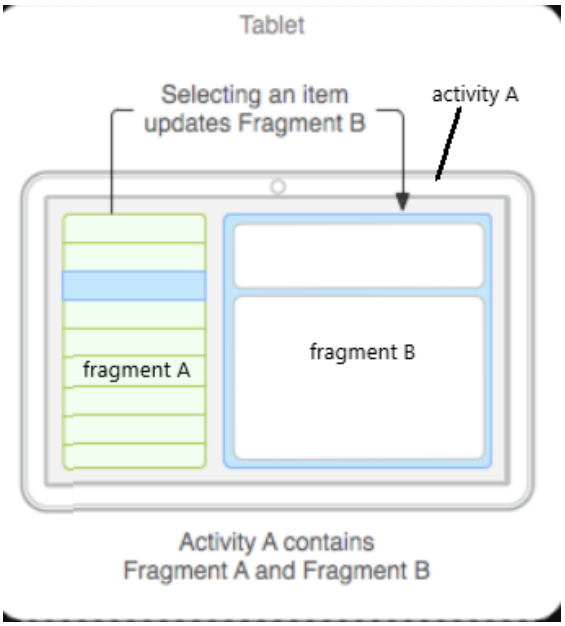
Emprova is de applicatie die GPS verkoopt aan zijn klanten. Het is een softwarepakket dat instaat voor het versoepelen en vergemakkelijken van tijdsplanning, personeelsplanning en toegangscontroles. Sommige klanten van GPS zijn grote spelers op de markt. Dit wil zeggen dat ze ook veel personeel in dienst hebben. Het kan dan voor werknemers moeilijk zijn om te weten waar collega's zich bevinden, zeker als het bedrijf gebruikmaakt van consultants voor het onderhouden van klantenrelaties. Dit kan opgelost worden door de locatie, waarop een werknemer zich het laatst registreerde, weer te geven aan de hand van een kaart. Met de huidige mobiele applicatie van Emprova is dit jammer genoeg niet mogelijk.

2.2 Doelstellingen

Het doel van deze stage is een mobiele applicatie te bouwen die het mogelijk maakt om collega's via geolocaties te traceren. Hiervoor moet er gebruikgemaakt worden van de Google Maps API. De locaties moeten worden opgehaald aan de hand van een bestaande database en worden via een .NET Web API geserveerd aan een mobiele client in JSON.

Aanvankelijk was de enige doelstelling het rechttoe rechtaan plotten van de opgehaalde locaties op een kaart. Achteraf werden er meerdere doelstellingen aan het project toegevoegd. Zo werd er gevraagd om navigatie naar collega's te voorzien op basis van de door de gebruiker geïnstalleerde navigatietools. Vervolgens was het noodzakelijk om op naam van een medewerker te kunnen zoeken en verschillende registraties van elkaar te onderscheiden aan de hand van filtering. Hiervoor moest een menu onder de kaart voorzien worden. Een volgende doelstelling die door het stagebedrijf gegeven werd is het sturen van een notificatie wanneer een werknemer aankomt bij een klant zodat de gebruiker zich via een interface in die notificatie kan registreren. Om dit voor elkaar te krijgen raadde het bedrijf aan gebruik te maken van *geofencing*.

Qua lay-out werd er gevraagd om een eenvoudig loginscherm te ontwikkelen, bestaand uit twee inputvelden en een knop. Na het inloggen moet de gebruiker verwezen worden naar een centrale *activity* waarin verschillende *fragments* geplaatst worden. Deze *activity* moet als menu fungeren met als standaard-*fragment* de kaart. Wanneer de gebruiker op een titel in het menu klikt, moet er een ander *fragment* ingeladen worden. Verder werden er geen doelstellingen opgelegd qua lay-out.



Figuur 1: Fragment versus Activity [1]

2.3 Tijdsplanning

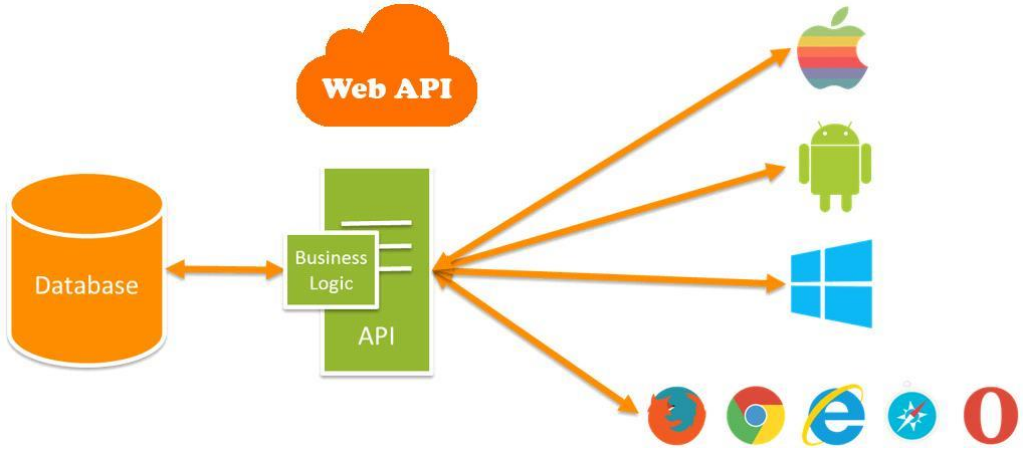
Bij de start van de stage is er een duidelijke planning opgesteld. Omdat het researchgedeelte en het ontwikkelen twee opzichzelfstaande delen van de stage zijn, werden ze opgesplitst in twee aparte planningen. Het is zaak om deze planning min of meer te volgen. Wanneer er zich problemen voordoen is het normaal dat er een beetje afgeweken wordt, dit hoeft geen probleem te vormen want de planning werd opgesteld zodat er voldoende tijd is voor elk onderdeel.

2.4 Gebruikte technologieën

2.4.1 ASP.NET Web API

ASP.NET Web API is een framework dat het mogelijk maakt om http-services te bouwen die een grote omvang clients kunnen bereiken, zoals browsers of mobiele applicaties. Het framework is ideaal voor het bouwen van REST-applicaties. [2]

Om de mobiele applicatie te kunnen laten communiceren met de services en databases van GPS maakt het bedrijf gebruik van een .NET API. Data opvragen uit zo'n API gebeurt in enkele stappen.



Figuur 2: .NET API [29]

Er wordt vanuit de mobiele applicatie een *request* gedaan naar de API aan de hand van een URL. Er zijn verschillende soorten *requests,* namelijk GET, POST, PUT, DELETE. Een GET-*request* staat in voor het ophalen van data uit de API. De POST-*request* behandelt toevoegen van data. Dit zijn de enige twee *requests* waarmee er in deze stageopdracht gewerkt worden. Nadat de API een *request* ontvangen heeft gaat het kijken welke URL er is meegegeven. Aan de hand van deze URL weet de API welke data het uit de database moet ophalen en vervolgens moet terugsturen naar de applicatie waarvan het een *request* ontving. De data wordt door de API omgezet en teruggestuurd in JSON-formaat.

2.4.2 Java

Java is een platformonafhankelijke, objectgeoriënteerde programmeertaal. Door het feit dat Java objectgeoriënteerd is, wordt het veel gebruikt door programmeurs van over heel de wereld. Het is ook de populairste programmeertaal voor het bouwen van mobiele applicaties in Android. [3] [4]

2.4.3 Android Studio

Android Studio is de officiële IDE voor het ontwikkelen van Androidapplicaties. Het is gebaseerd op Intellij, een Java IDE voor het ontwikkelen van software. Om applicatieontwikkeling voor Android te ondersteunen, maakt Android Studio gebruik van een emulator. [5]

2.4.4 Lottie

Lottie is een plug-in voor het realtime renderen van After Effects-animaties. Zo kunnen applicaties gebruik maken van animaties op dezelfde manier als ze gebruikmaken van foto's. [6]

2.4.5 Retrofit

Retrofit is een REST-client ontworpen voor Java en Android. Retrofit maakt het makkelijk om data te versturen of te ontvangen van REST-services. In Retrofit kan er gekozen worden welke converter er gebruikt wordt voor het converteren van de JSON-data. [7]

2.4.6 Jackson

De JSON Processing API die meegeleverd wordt met Java is niet echt gebruiksvriendelijk; daarom wordt er gebruik gemaakt van Jackson. Jackson zorgt voor een automatische transformatie van JSON- naar Java-object en omgekeerd. [8]

2.4.7 Google Maps API

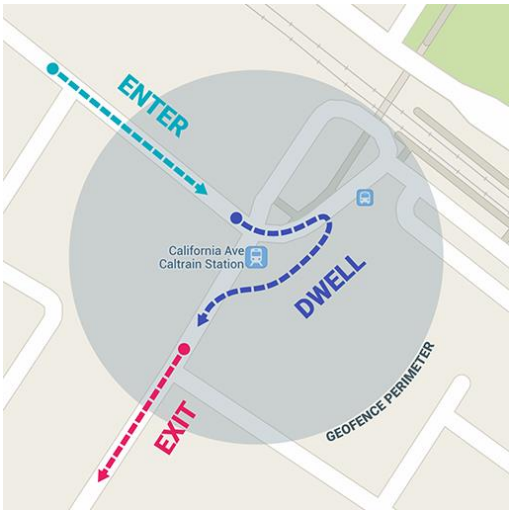
Google Maps laat de gebruiker nuttige data zien over plaatsen of locaties. De API zorgt ervoor dat Google Maps geïmplementeerd kan worden in een applicatie zonder dat er een webpagina moet ingeladen worden. De service bevat de functionaliteit om een map te maken op basis van parameters die worden meegegeven in een URL. Vervolgens wordt deze URL doorgestuurd in een http-*request*. [9]

2.4.8 Google Maps SDK Android

De Google Maps SDK voor Android zorgt ervoor dat kaarten die gebaseerd zijn op data van Google Maps, makkelijk weergeven kunnen worden op Androidapplicaties. De API zorgt automatisch voor toegang tot de Google Maps-servers, het downloaden van data, het tonen van de map en het reageren op interactie met de kaart. Met de Google Maps SDK is het ook makkelijk om markers, veelhoeken en bedekkingen te tekenen. [10]

2.4.9 Android Geofencing API

Soms willen gebruikers een bepaalde app gebruiken op een bepaalde locatie. Bijvoorbeeld de app van Colruyt, wanneer de gebruiker aankomt bij een warenhuis van Colruyt, of de app van de metro, wanneer de gebruiker aankomt bij een station. Zonder implementatie van de *Geofencing* API moet de gebruiker bij aankomst navigeren naar de app en hem manueel openen. Met gebruik van de *Geofencing* API kan er een bepaald gebied met grenzen afgebakend worden. De applicatie zendt dan een notificatie uit wanneer de gebruiker het afgebakende gebied betreedt. [11]



Figuur 3: Geofencing [11]

2.5 Afspraken

Omdat we onze stageopdracht uitwerken met twee personen is het nuttig om enkele afspraken na te leven gedurende de stage.

2.5.1 Vervoer

Op de route van PXL naar het stagebedrijf is een klant van GPS gelegen. Indien we zouden carpoolen zou één van ons de functionaliteit van de *geofencing*-API kunnen testen. Daarom hebben we besloten om omstreeks 8u15 samen te komen aan de B-blok van de PXL en vervolgens samen naar het stagebedrijf te rijden.

2.5.2 Taakverdeling

Omwille van het feit dat we met twee aan hetzelfde project werken is het noodzakelijk dat we een goede taakverdeling naleven. Dit vermijdt dubbel werk en het verlies van kostbare tijd. We bekijken de grootte van elke specifieke taak en proberen beide aan taken van dezelfde omvang te werken. Hierdoor gaat het niet lijken alsof er één persoon meer werk gedaan heeft dan de andere. Van elkaar weten aan welke taak we bezig zijn is ook nuttig zodat er bij eventuele problemen meteen hulp kan gevraagd worden aan de andere zonder dat hij eerst moet uitzoeken waarover die specifieke taak handelt.

2.5.3 Communicatie

Een goede communicatie is cruciaal in elke professionele omgeving waarin er in teams wordt gewerkt. Tijdens dit stageproject is dat niet anders. We introduceren GIT in ons stagebedrijf, daardoor werken we uitsluitend op de *master branch* zodat onze stagementor ook altijd de juiste code heeft. Wanneer ik en mijn junior-collega aan hetzelfde bestand zouden werken, zou dit problemen vormen bij het uitwisselen van onze code. Daarom maken we de afspraak dat er nooit aan dezelfde code wordt gewerkt. Maar niet enkel de communicatie tussen mezelf en mijn junior-collega is belangrijk, ook de communicatie met de stagepromotoren is van groot belang. Indien er vragen zijn over het research gedeelte kunnen we altijd bij Carine Derkoningen terecht. Bij vragen over het ontwikkelen van de applicatie kunnen we terecht bij Ingo Schelfhout, Frank Wynants of Nick Gaens.

3 Uitwerking stageopdracht

3.1 Beschrijvingen

3.1.1 Beschrijving Retrofit versus Volley

Voor de start van de stage gebruikte het stagebedrijf Volley voor het ophalen van data uit de backend. Volley is een oudere *library* en daarom was een vergelijking met een nieuwere *library* geen overbodige luxe. Een *library* die vaak terugkomt in de literatuur is Retrofit. De voor en nadelen werden voor beide *libraries* opgesomd.

Voordelen van Volley:

- Zeer uitgebreid en flexibel *caching mechanism*:
    - Wanneer er door Volley een *request* wordt gemaakt, wordt er eerst in de cache gekeken of er voor die *request* al een response beschikbaar is. Indien er een response gevonden wordt, wordt deze teruggegeven. In andere gevallen wordt er een *request* over het netwerk gestuurd.
- Ondersteunt zowel POST-*requests* als *multipart uploads*.
- Er kan een *retry policy* worden voorzien die onder andere de time-outs en het aantal keer dat er opnieuw geprobeerd wordt ondersteunt.
- Kan automatisch vier verschillende responsetypes terugsturen naar gelang het type *request*, namelijk *StringResponse, JsonObjectResponse, JasonArrayResponse, ImageResponse*.
- Biedt ondersteuning voor het laden van afbeeldingen aan de hand van een aangepaste view die speciaal hiervoor werd ontworpen.

Enkele voordelen van Retrofit:

- Volledige ondersteuning van POST-*requests* en *multipart uploads.*
- Bevat veel meer response-types die automatisch verwerkt worden.

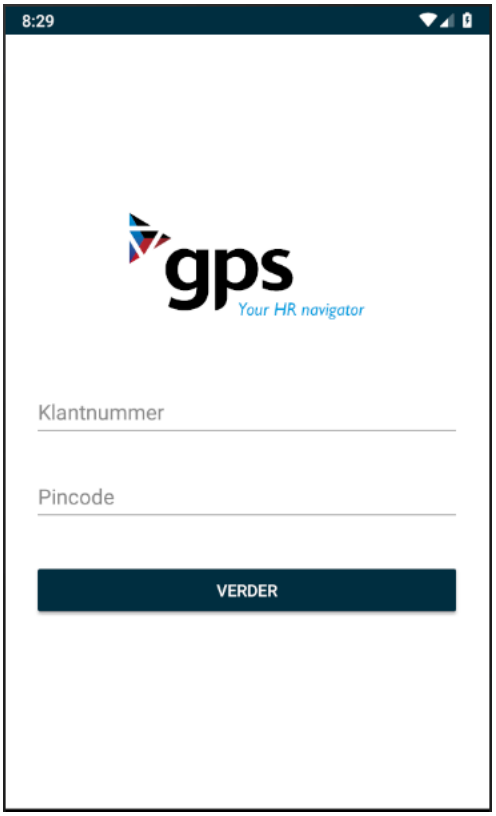
Volley is verouderd en wordt niet meer veel gebruikt in moderne applicaties. Het grote nadeel hiervan is dat er minder en minder te vinden is over de implementatie van de *library*. Eveneens ontbreekt officiële documentatie. Hierdoor wordt het voor de ontwikkelaars moeilijk om Volley in de praktijk te gebruiken.

Retrofit is een moderne, lichte *library* voor het ophalen van data via een webservice. Doordat Retrofit zo modern is, is er heel wat documentatie over te vinden en is de configuratie zeer gemakkelijk.

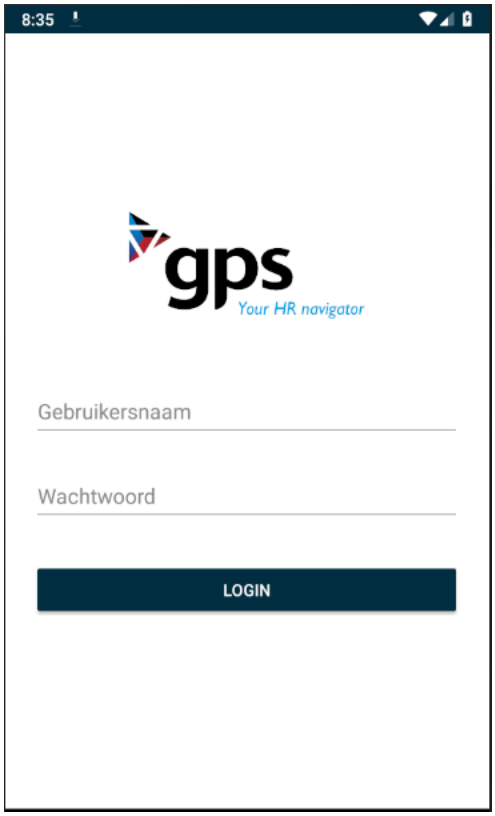
3.2 Stageverloop

3.2.1 Aanmeldingsschermen

Om de gebruiker gepersonaliseerde gegevens en services te serveren is het noodzakelijk dat de data van de gebruiker opgevraagd wordt. Hiervoor moet de gebruiker de mogelijkheid krijgen om zich aan te melden op de applicatie. De applicatie moest twee aanmeldschermen voorzien. Eén waarop de gebruiker het klantnummer van zijn bedrijf invult en één waarop de gebruiker zijn eigen gebruikersnaam en wachtwoord ingeeft.



Figuur 4: Aanmelden met klantnummer

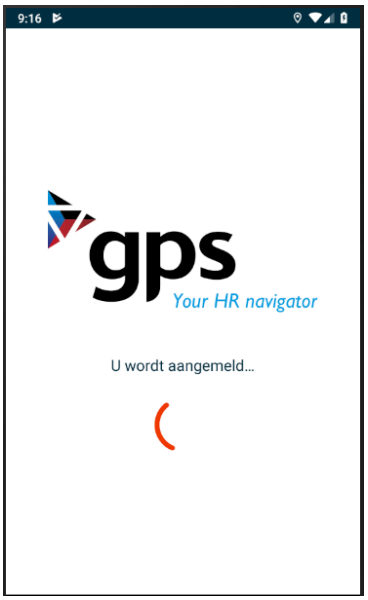


Figuur 5: Aanmelden met gebruikersgegevens

Het zou heel vervelend zijn voor de gebruiker indien hij, bij het opstarten van de app, zijn gegevens telkens opnieuw zou moeten invullen. Daarom werd door GPS gevraagd dit gebruiksvriendelijker aan te pakken. Wanneer de gebruiker voor een eerste keer zijn gegevens ingeeft, moeten de ingevulde gegevens worden bijgehouden in het geheugen van de applicatie, zodat wanneer de gebruiker de applicatie een volgende keer opent hij automatisch wordt aangemeld door het systeem.

```
/**
 * checks if all sharedprefs for auto-login are available.
 */
private void checkForAutoLogin()
{
    if (SharedPreferenceManager.hasAllLoginSharedPrefs( context: this))
    {
        apiPostCustomer();
    }
    else
    {
        Intent intent = new Intent( packageContext: this, LoginCustomerActivity.class);
        startActivity(intent);
    }
}
```

Figuur 6: Code automatisch aanmelden



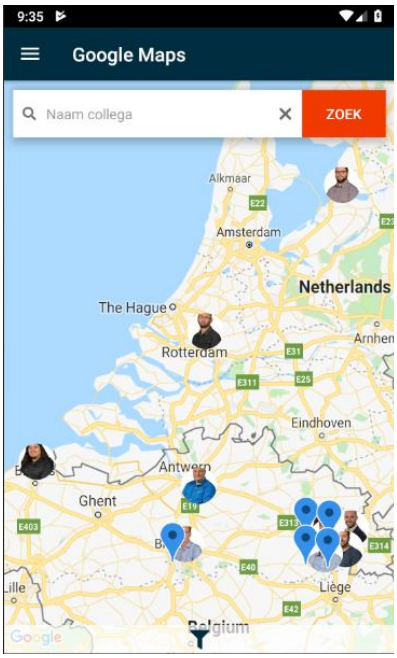
Figuur 7: Automatisch aanmelden

In figuur 6 is te zien hoe het systeem checkt of automatisch aanmelden mogelijk is. Er wordt in het geheugen van de applicatie gekeken of er gebruikersgegevens teruggevonden worden. Indien dit het geval is wordt de gebruiker automatisch aangemeld. Indien er geen gegevens teruggevonden worden wordt de gebruiker doorgestuurd naar het scherm om zijn klant- en gebruikersgegevens in te vullen (zie figuur 4 en 5).

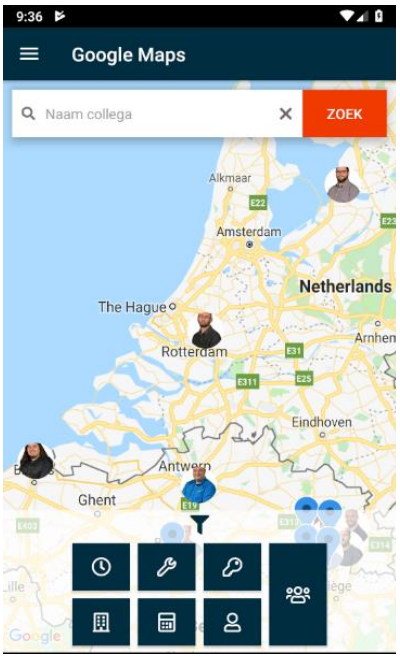
3.2.2 Kaart

Een volgende vereiste was het tonen van de locatie van alle medewerkers, alle geolocaties en de gebruiker zelf op een kaart. Hier moest goed over nagedacht worden. Wat is de beste manier om dit allemaal te tonen op een kaart? Gaat het overzichtelijk blijven? Zoals in figuur 6 te zien is, werd er gekozen om alle medewerkers met foto's aan te duiden, de geolocaties een blauwe kleur te geven en om de locatie van de gebruiker met een roze markering te laten opvallen.

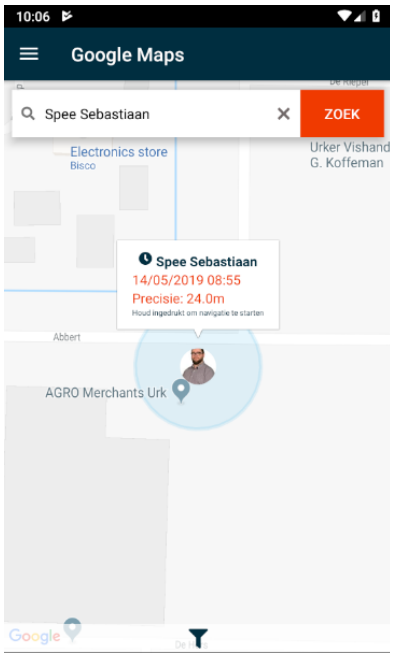
Wanneer een bedrijf een groot aantal medewerkers telt kan zo een kaart al snel onoverzichtelijk worden. Om dit te vermijden was het voor GPS noodzakelijk dat er enkele filters en een zoekfunctie in het systeem geïmplementeerd werden. Om zoveel mogelijk van de kaart te kunnen laten zien werd er gekozen voor een filterscherm dat door de gebruiker omhoog en omlaag gesleept kan worden. Met dit filterscherm kan de gebruiker dan bijvoorbeeld kiezen om enkel de jobregistraties van alle collega's te laten zien, of om alle bedrijven te verbergen zodat enkel de collega's nog op de kaart staan. Er zijn verschillende manieren om enkel de data te laten zien die de gebruiker op dat moment echt nodig heeft. Wanneer de gebruiker weet van welke collega hij de locatie en data wil opvragen, kan hij simpelweg de naam van die collega intypen in de zoekbalk en de kaart zal dan alleen die gebruiker tonen en inzoomen zodat de gebruiker een beter beeld krijgt van waar die collega zich bevindt.



Figuur 8: Kaart na opstarten



Figuur 9: Filtermenu



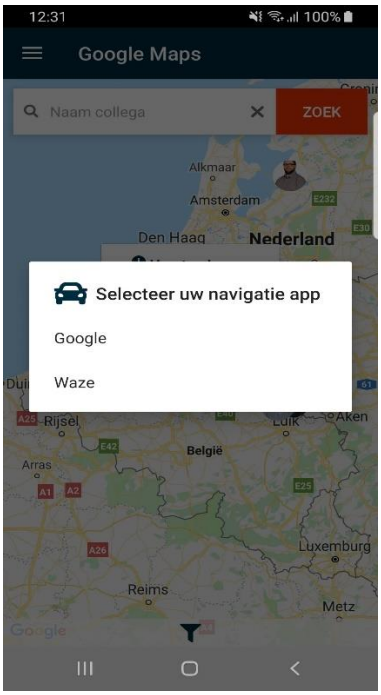
Figuur 10: Zoekfunctie + infoscherm

Om info over een collega te verschaffen wordt er gebruik gemaakt van een infoscherm. Hierop staat het type registratie van de collega, de datum en het tijdstip waarop de collega zijn registratie doorvoerde, de precisie van de locatiebepaling.

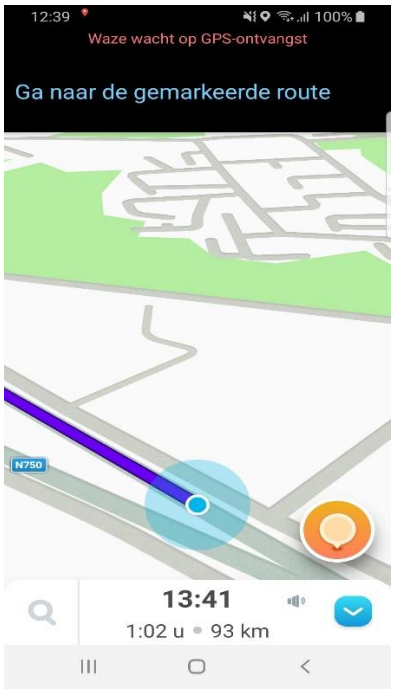
3.2.3 Navigatie naar een collega

Stel, Spee Sebastiaan, een consultant, zit bij een klant en heeft technische bijstand nodig van een IT-medewerker binnen het bedrijf. Sebastiaan vraagt aan Nick, softwareontwikkelaar binnen GPS, om langs te komen. Nick gaat via de applicatie kijken wat de locatie van Sebastiaan is, wanneer Nick de locatie van Sebastiaan gevonden heeft zou hij moeten inzoomen om het exacte adres van Sebastiaan te achterhalen om vervolgens in te geven in zijn navigatiesoftware. Dit is natuurlijk niet handig. Daarom was het noodzakelijk om met het systeem in enkele tikken navigatie te kunnen starten naar een collega.

Dit werd gedaan aan de hand van de infoschermen die de gebruiker te zien krijgt wanneer hij op de marker van een collega tikt. Zoals aangegeven op figuur 8 kan de gebruiker het infoscherm inhouden om navigatie te starten. Indien de gebruiker dit scherm inhoudt krijgt hij de mogelijkheid om tussen een lijst van geïnstalleerde navigatieapps een app naar keuze te selecteren. Met die app wordt de route dan berekend.



Figuur 11: Keuze navigatieapp

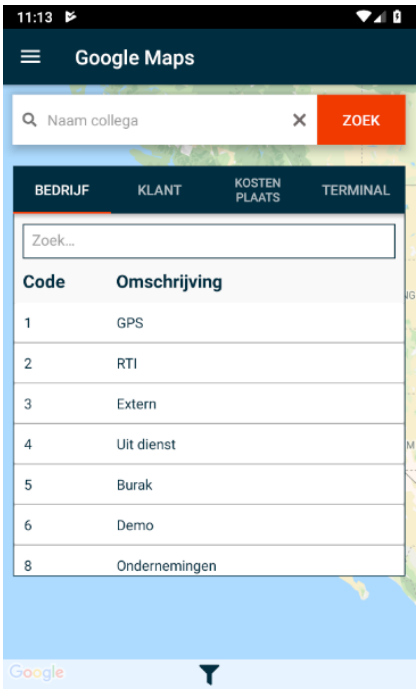


Figuur 12: navigatie na selecteren Waze

3.2.4 Toevoegen geolocatie

Een volgende feature die de applicatie moest bevatten was het koppelen van geolocaties aan locatie van de gebruiker.

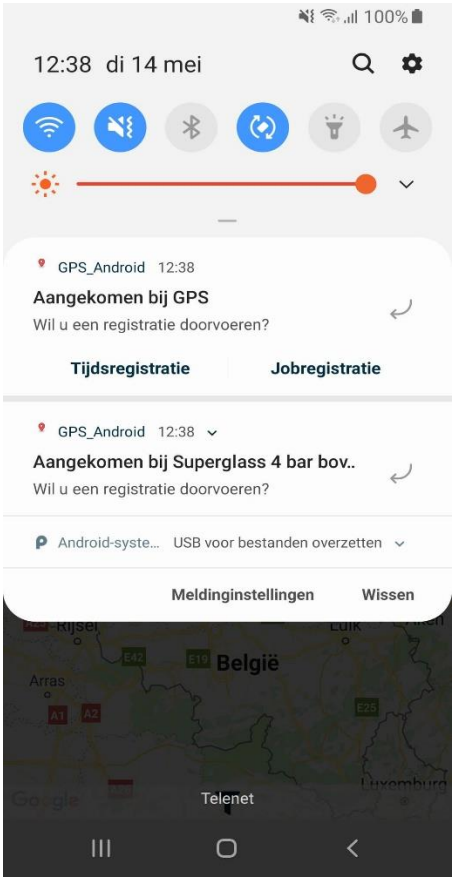
Indien een gebruiker genoeg rechten heeft kan hij geolocaties toevoegen aan het systeem. Om deze feature te voorzien wordt er op dezelfde manier tewerk gegaan als bij het navigeren naar een collega. Als de gebruiker het infoscherm van zijn eigen marker even inhoudt opent een venster waarin hij met enkele klikken een geolocatie kan koppelen aan zijn eigen locatie.



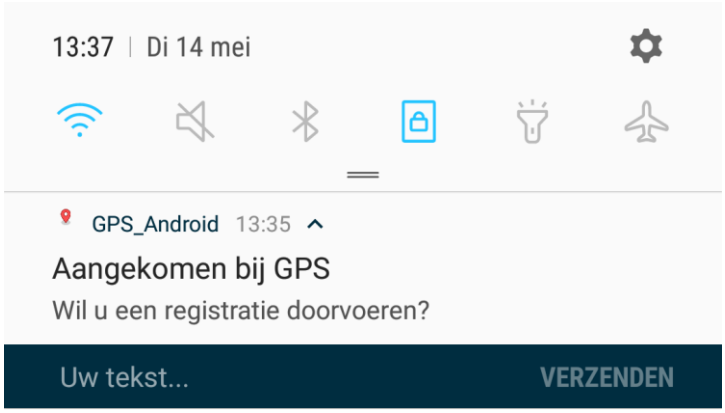
Figuur 13: Koppelen geolocatie aan eigen locatie

3.2.5 Geofencing

Op dit moment moet de gebruiker manueel naar de registratieschermen gaan om zich te registreren in de app wanneer hij aankomt bij een klant. *Geofencing* zorgt ervoor dat de gebruiker, wanneer hij een bepaald gebied (gebouw van een klant) betreedt, een melding krijgt met de vraag of hij een job of tijdregistratie wil doorvoeren. De gebruiker kan zo gemakkelijk vanuit de notificatie een tijdregistratie boeken. Dit is te zien in figuur 15. Indien hij zich wil registreren voor een job wordt het scherm voor jobregistraties geopend.



Figuur 14: Notificaties bij aankomst

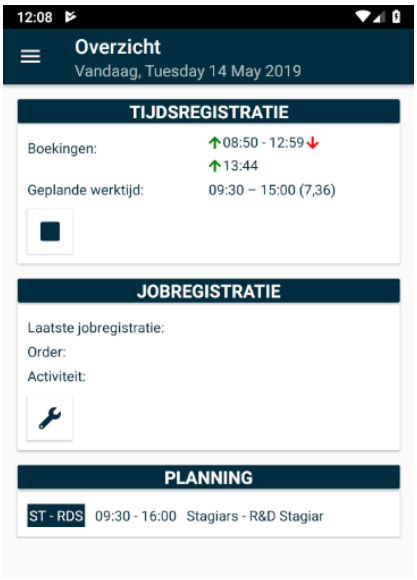


Figuur 15: Tijdregistratie via de notificatie

3.2.6 Overzicht, tijdregistratie, jobregistratie

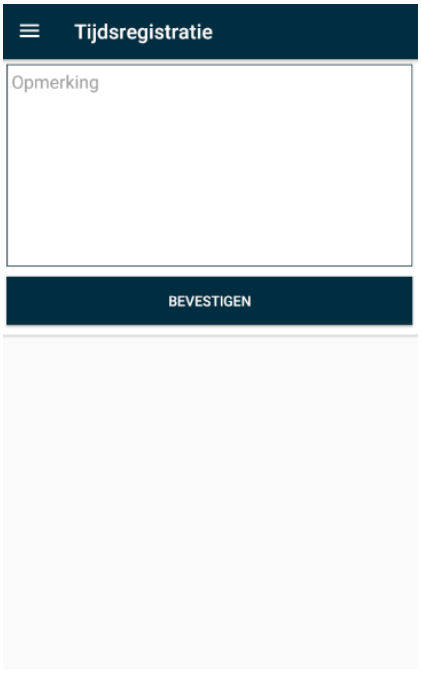
Om de gevraagde features te kunnen implementeren waren er ook enkele andere vensters nodig. Er volgt een korte toelichting over deze vensters omdat ze veel tijd in beslag hebben genomen en cruciaal waren voor de rest van het project.

Het overzicht venster bevat de dagelijkse planning van de gebruiker. Deze planning is nodig om te kunnen bepalen op welk tijdstip de gebruiker moet vertrekken naar de volgende klant, rekening houdend met het verkeer. Dit is momenteel niet mogelijk aan de hand van de webservices van GPS, maar het scherm werd reeds toegevoegd zodat het later makkelijk is om de functionaliteit te implementeren.

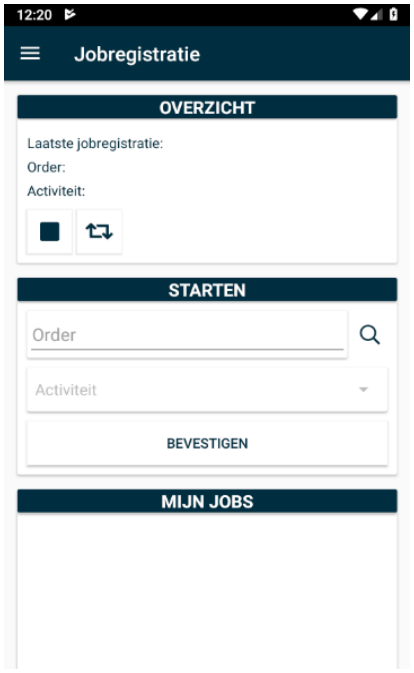


Figuur 16: Overzicht pagina

De tijd en jobregistratievensters waren beide nodig om te kunnen werken met de *geofencing*-notificaties.



Figuur 17: Tijdregistratie venster



Figuur 18: Jobregistratie venster

4 Reflectie

Gedurende een lange tijd meedraaien in een bedrijfsomgeving zorgt voor niet te onderschatten ervaring met het oog op een professionele toekomst. Je ziet voor een eerste keer hoe ervaren programmeurs te werk gaan bij het ontwikkelen van grote applicaties. Ik heb geleerd dat communicatie hierin een grote succesfactor is. Er wordt constant overlegd en slechts als iedereen op dezelfde golflengte zit wordt er geprogrammeerd. Het kan soms zijn dat ze dagen overleggen, zaken afspreken, om vervolgens één dag te programmeren.

Alle afdelingen binnen GPS zijn opgedeeld in teams. Ik werd opgenomen in het 'Angry-Nerds'-team, het team van de ontwikkelaars. Eén van de teamleden stond altijd klaar wanneer ik een vraag had. Ze gaven me echt het gevoel dat ik van even groot belang was als alle vaste waardes binnen het bedrijf.

Wanneer het stagebedrijf je zoveel hulp biedt wordt er ook verwacht dat je binnen een bepaalde termijn iets nuttig doet met de informatie die je gegeven wordt. Het is als stagiair soms moeilijk om in te schatten hoelang je nodig gaat hebben aan een bepaald onderdeel van de applicatie. De deadlines die werden opgelegd door het stagebedrijf zorgden daarom dan ook voor extra gezonde druk bij het programmeren.

Op het moment dat ik de stageopdracht voor het eerst las had ik er meteen een goed gevoel bij. Het is altijd fijn als je een stageopdracht hebt waarbij je echt het resultaat van je werk kan zien. Tevens zijn mobiele applicaties de toekomst, daarom is het goed dat ik als student al een keer in aanraking kom met een groot project in Android.

Gedurende de stage ben ik als student natuurlijk tegen enkele obstakels opgelopen. Ik heb lang vastgezeten bij het *Geofencing*-gedeelte. Hieruit heb ik geleerd dat je in zo'n situaties rustig moet blijven en hulp moet vragen bij andere mensen. Mijn bedrijfspromotor heeft het probleem samen met mij bekeken en stuurde me in de juiste richting waarna ik het probleem relatief eenvoudig kon oplossen.

Ik ben heel tevreden over het eindresultaat. Alle features die het bedrijf voor ogen had zijn geïmplementeerd door mezelf en Yoran Nelissen. De stage is voor mij dan ook meer dan geslaagd. GPS was heel tevreden over het geleverde werk en ik kan het bedrijf met grote overtuiging aanbevelen als stageplek.

II. Onderzoekstopic

1 Onderzoeksvraag en hypothese

1.1 Onderzoeksvraag

Om er honderd procent zeker van te zijn dat de Google Maps API de juiste oplossing is voor GPS, moet dit bewezen worden in een vergelijkende studie met één of meerdere alternatieven. Vervolgens kan er bepaald worden of het voor GPS mogelijk is om met een alternatieve API een vergelijkbaar of beter resultaat te bekomen dan wanneer er gebruik gemaakt wordt van de Google Maps API.

1.2 Deelvragen

Frameworks voor het tekenen van kaarten zijn niet schaars. De kunst is het juiste alternatief te vinden zodat er een waardige vergelijking kan plaatsvinden met de bekende Google Maps API. Er zijn talloze alternatieven voor gebruik op webapplicaties, maar dit is niet waar GPS naar op zoek is. Aangezien GPS dit systeem op een mobiele applicatie wil draaien is een eerste vraag die gesteld kan worden of de alternatieve API geïntegreerd kan worden in een Android- en IOS-applicatie. Dit vermindert het aantal bruikbare mogelijkheden aanzienlijk.

Als de alternatieve API gedraaid kan worden op Android en IOS is het van groot belang dat de API een goede documentatie bevat. Als dit niet het geval is, kan het bouwen van een applicatie gebruik makend van deze API tijdrovend en frustrerend zijn. Het is nooit aangeraden om een *library*, framework of API met een slechte documentatie te gebruiken.

Een volgende voorwaarde waaraan de alternatieve API moet voldoen is gebruiksvriendelijkheid. De kaart moet makkelijk te besturen en duidelijk zijn. Een combinatie van deze twee voorwaarden is cruciaal. Zo kan een kaart makkelijk te besturen zijn, maar als hij onduidelijk is heeft de gebruiker hier ook geen voordeel bij. Dit geldt ook omgekeerd.

Wanneer er op basis van vorige voorwaarden een API gekozen is, kan er worden overgegaan tot het vergelijken van de prestaties en functionaliteit van beide API's. Wanneer de alternatieve API vergeleken wordt met de Google Maps API worden ook de mogelijkheden van de meegeleverde SDK onder de loep genomen. Kan deze API evenveel als de Google Maps API? Zo ja, is het even makkelijk om de functionaliteit die geïmplementeerd is met de Google Maps API na te bouwen met de alternatieve API? En kan de alternatieve API zorgen voor een bonus op de al bestaande functionaliteit? Zo zijn er enkele cruciale zaken die mogelijk moeten zijn met de alternatieve API:

- Kan de API gebruikt worden voor het tekenen van flexibele markers?
- Is het mogelijk om met de API aangepaste infoboxen te tonen?
- Bevat de API de mogelijkheid om alle medewerkers op een aangepaste schaal te tonen?
- Wat is het financiële plaatje van de API? Is het haalbaar voor GPS?

Iets wat de Google Maps API op dit moment niet heeft, is ingebouwde navigatie. Indien de alternatieve API deze functionaliteit wel bevat, kan dit een groot voordeel bieden ten opzichte van de kaart van Google.

1.3 Hypothese

Zoals reeds aangehaald zal het vinden van een alternatieve API, die op Android en IOS draait, geen probleem zijn. Er zijn tal van goed gedocumenteerde mogelijkheden beschikbaar. Om het onderzoek af te bakenen wordt er dieper ingegaan op één alternatief.

Na een vergelijkend onderzoek wordt naar alle waarschijnlijkheid aangetoond dat niet alle functionaliteit even makkelijk implementeerbaar is. Het zal bijvoorbeeld misschien makkelijker zijn om een marker te tekenen op de Google Maps API dan om een marker te tekenen op een alternatieve API. Beter doen dan de Google Maps API wordt moeilijk. Google staat bekend om zijn flexibele API's en goed ondersteunde SDK's. Hierbij komt ook nog eens het feit dat Google altijd uitgebreide documentatie schrijft voor al zijn producten.

Als men vervolgens de gebruiksvriendelijkheid gaat vergelijken heeft Google altijd een streepje voor. Doordat het marktleider is op het gebied van kaarten en de kaart van Google in het grote deel van de applicaties gebruikt wordt, kan men ervan uitgaan dat de gebruiker een goede interactie met de kaart zal ervaren. Het is dan ook haast een onmogelijke opgave om beter te doen dan Google als het gaat over gebruiksvriendelijkheid.

Uit enkele eerste testen blijkt dat de Google Maps API geen ingebouwde navigatie bevat. In de veronderstelling dat er een alternatief bestaat dat wel over ingebouwde navigatie beschikt, zal Google Maps API de duimen moeten leggen. Op voorwaarde dat de alternatieve API natuurlijk ook aan de overige eisen voldoet.

2 Onderzoeksmethoden

Om de onderzoeksvragen te beantwoorden, moet eerst en vooral een uitgebreide literatuurstudie gevoerd worden.

Om te beginnen moet er gezocht worden naar een bruikbaar alternatief voor de Google Maps API. Er zijn enkele grote spelers op de markt die Google het vuur aan de schenen proberen te leggen. Het is zaak om aan de hand van een korte blik op de literatuur het meest geschikte alternatief te selecteren. Welke API komt het dichtst in de buurt van Google? Wie claimt er iets meer te kunnen dan Google?

Vervolgens moeten de twee kaarten op functioneel vlak vergeleken worden. Wat kan de een meer dan de ander? Hiervoor is een grondiger onderzoek aan de hand van literatuur nodig. Er wordt een vergelijkingsmatrix opgesteld waarin de voor- en nadelen van beide API's uit de doeken worden gedaan. Dit zal een waslijst aan informatie opleveren. Er wordt dan gekeken wat handig is voor het bedrijf binnen dit project en in de toekomst. Het kan zijn dat het alternatief een betere oplossing biedt voor één aspect, maar dat daarna blijkt dat het op langere termijn toch niet de juiste oplossing biedt, omdat de mogelijkheden van het alternatief gelimiteerd blijken te zijn. Om dit te vermijden leggen we de verschillen aan de hand van de verstrekte informaliteit voor aan GPS en zullen zij beslissen welke oplossing voor hen het beste lijkt.

Hierna is het de bedoeling dat deze verschillen in de praktijk worden gebracht. Er wordt een applicatie gebouwd die twee schermen bevat. Eén scherm bevat de kaart van Google en een ander scherm bevat de alternatieve kaart. De applicatie wordt eerst op de kaart van Google uitgebouwd, waarna de cruciale verschillen tussen de twee kaarten worden aangetoond door deze uit te werken op de alternatieve kaart. Hierdoor zal voor zowel GPS als voor de gebruiker duidelijk zijn wat de beste optie is.

Ten slotte is het de bedoeling dat enkele testgebruikers de vergeleken aspecten van de map testen en feedback verlenen aan het bedrijf qua performance en gebruiksvriendelijkheid.

3 Literatuurstudie

Op basis van relevante artikels en gelijkaardige experimenten kunnen volgende conclusies getrokken worden over de onderzoeksvragen en de hypothese.

3.1 Alternatieven voor de Google Maps API

Het kan overweldigend en moeilijk zijn om een *mapping*-API te kiezen als het bedrijf nog nooit een map heeft ontwikkeld, en nog niet echt weet welke richting het project op zal gaan. Het is daarom belangrijk om te kijken welke opties er beschikbaar zijn. Hierna kunnen deze opties met elkaar vergeleken worden en kan er op basis van deze vergelijking beslist worden welke *mapping*-API er gebruikt zal worden.

Google Maps is voor de meeste developers het eerste platform waaraan gedacht wordt bij het bouwen van een interactieve kaart. Dit komt door de alomtegenwoordigheid en populariteit van Google. Maar de Google Maps API heeft zijn populariteit niet enkel aan zijn naam te danken.

De Google Maps API bevat heel veel handige meegeleverde services zoals de Google Places API, de Google Geolocation API, de Google Time Zone API en de Google Geocoding API. Deze services kunnen heel makkelijk met elkaar samenwerken. Dit zorgt voor een grote bonus ten opzichte van andere *mapping*-API's, die meestal *third-party* services moeten gebruiken om dezelfde functionaliteit te kunnen voorzien. Sterker nog, volgens Codementor [12] is Google de beste en misschien zelfs wel de enige optie als verkeersinformatie buiten het westelijk gedeelte van de wereld noodzakelijk is. Verkeersinformatie buiten het westen is heel moeilijk te vinden, Google Maps API geeft hierin volgens Codementor [12] de accuraatste oplossing.

Ook de documentatie van Google is makkelijk om mee te werken volgens Codementor [12]. Al kan het soms moeilijk zijn om iets specifieks over *mapping* terug te vinden omdat de documentatie van Google verschillende platformen moet ondersteunen en daarom heel groot geworden is. Dit mag echter geen probleem vormen, want er zijn een groot aantal artikels over de Google Maps API aanwezig op websites als StackOverflow, blogs, etc.

Een derde en laatste aspect dat kan doorwegen in de keuze van een *mapping*-API is de kwaliteit van de code en de interface. De volgende bevindingen zijn gebaseerd op ervaringen van Victor Gerard Temprano [12]. Wanneer er functionaliteit toegevoegd wordt aan een kaart gaat erin heel veel gevallen een marker getekend moeten worden. Hierdoor kan het zijn dat de kaart snel vol komt te staan met markers. Het is dan een grote bonus als de markers op een efficiënte manier verwerkt worden. De Google Maps API kan volgens de bevindingen van Victor Gerard Temprano makkelijk meer dan tienduizend markers op een efficiënte manier tekenen. Tegelijkertijd is de kaart van Google ook heel *responsive*. Nieuwe gebieden en componenten op de map worden binnen enkele milliseconden ingeladen.

Er zijn ook enkele nadelen aan de Google Maps API. De API van google is geen opensourcesoftware. Dit betekent dat de API minder toegankelijk is voor ontwikkelaars. Een tweede nadeel is het feit dat een ontwikkelaar zich moet houden aan de servicevoorwaarden van Google wanneer hij gebruik wil maken van de Maps API. Dit houdt in dat de ontwikkelaar rekening moet houden met het privacybeleid en de juridische regels van Google. Een laatste nadeel dat de Google Maps API met zich meebrengt is de standaard layout. Google verplicht zijn gebruikers om de typische stijl van Google te gebruiken. Enkele kleuren kunnen gewijzigd worden, maar het grote logo van Google zal altijd onderaan de kaart te zien zijn.

3.1.1 Mapbox

Mapbox is een platform voor het ontwikkelen van mobiele en webapplicaties aan de hand van locatiedata. Het voorziet locatieservices zoals kaarten en navigatie. Het platform behandelt het geolocatiegedeelte voor de ontwikkelaar zo dat die zich kan concentreren op het bouwen, uitwerken en ontwerpen van de applicatie. [13]

Ook de API van Mapbox heeft enkele handige meegeleverde services. Een eerste is de Maps API, die instaat voor het tekenen van rasters en figuren. Een tweede belangrijke service die meegeleverd wordt met Mapbox is de Styles API. Deze API zorgt ervoor dat de stijl, het lettertype en de afbeeldingen van de map kunnen worden aangepast. [14] Hoewel deze services leuk zijn voor het uitzicht van de kaart, hebben ze niet dezelfde omvang als bijvoorbeeld de Google Places API, waarmee miljoenen plaatsen van over de hele wereld kunnen worden opgevraagd.

Vergeleken met de documentatie van de Google Maps API is de documentatie van MapBox veel eenvoudiger. Omdat Mapbox zich enkel toelegt op het ontwikkelen van kaarten, blijft de documentatie overzichtelijk en is alles makkelijk en met duidelijke instructies terug te vinden.

De code van Mapbox is vergelijkbaar met de code van de Google Maps API. Enkele artikelen klagen over de performance van Mapbox wanneer er meer dan duizend markers worden getekend. [15] [16] In tegenstelling tot Mapbox heeft de Google Maps API geen problemen wanneer het dit aantal markers moet tekenen en bevat het vele manieren om het laden van markers nog sneller te maken. Een voorbeeld hiervan is het op verschillende manieren clusteren van de markers.

Een voordeel dat Mapbox heeft vergeleken met de Google Maps API is Mapbox Studio. Mapbox Studio is een applicatie die ervoor zorgt dat de gebruiker zijn eigen kaarten kan creëren, met de kleuren en datasets die hij belangrijk vindt. [17]

Een groot nadeel van Mapbox is dat het gebouwd is met OpenStreetMap; dit is een opensourceproject en Mapbox heeft hierdoor minder kaartbedekking dan de Google Maps API. Omdat bedrijven zich vaak in verschillende landen vestigen, moet de kaart een grote kaartbedekking hebben. Er wordt dan ook niet verder ingegaan op Mapbox.

3.1.2 Leaflet

Leaflet is een opensource javascript-*library* voor het bouwen van interactieve mobiele kaarten. [18]

Een eerste groot verschil met de Google Maps API is dat leaflet geen extra services levert om zijn map te ondersteunen. Het is de map, en alleen de map. Dus geen navigatie, geen plaatsherkenning, etc. Dit zijn tools die men achteraf zelf zal moeten installeren. Dit maakt het voor de ontwikkelaar moeilijker om een map te ontwikkelen die verschillende functionaliteiten implementeert. [12]

Volgens Codementor heeft Leaflet een duidelijke en typerende *javascript-plugin-style* documentatie en kan er ook heel veel over gevonden worden op het internet. Victor Gerard Temprano, schrijver van het artikel op Codementor en al gedurende een lange tijd ontwikkelaar, vindt de documentatie van Leaflet hierdoor beter als die van Google. [12]

Het derde belangrijke aspect van een SDK die gebruikt wordt voor het ontwikkelen van kaarten is de kwaliteit van de code en de interface van een kaart. Leaflet kan net zoals Google Maps gemakkelijk markers, pop-ups tekenen. Er zijn wel enkele verschillen tussen de twee. Bevindingen van Victor Gerard Temprano zeggen dat de Google Maps API minder problemen heeft met het tekenen van een groot aantal markers. Maar dit kan natuurlijk liggen aan de manier waarop markers getekend worden en wanneer ze getekend worden. Een tweede verschil is het feit dat Leaflet opensource is. Dit brengt enkele voor- en nadelen met zich mee. Omdat Leaflet opensource is wordt de code dagelijks nagekeken door een groot aantal ontwikkelaars. Dit kan Leaflet enkel ten goede komen en de code is bovendien niet gelimiteerd aan de servicevoorwaarden van Google. Een opensourceproject brengt jammer genoeg ook wat problemen met zich mee. Doordat Leaflet opensource is moet het ook gebruik maken van een opensource kaart. Het maakt gebruik van dezelfde kaart als Mapbox namelijk OpenStreetMap. OpenStreetMap heeft aanzienlijk minder kaartbedekking dan Google Maps. [12]

Bij gebruik van de Google Maps API is de gebruikersinterface voor elke ontwikkelaar hetzelfde. Op elke kaart staat het logo van Google onderaan. Dit is misschien niet naar de zin van elke ontwikkelaar. Leaflet daarentegen geeft de ontwikkelaar de mogelijkheid om het logo van Leaflet weg te halen. [12]

Doordat Leaflet geen extra services meelevert voor het opvragen van verkeersdata, plaatsherkenning, … en dit wel nodig is in het project van GPS, wordt er niet dieper ingegaan op deze *library*.

3.1.3 Open Layers

Open Layers is een opensource JavaScript *library* die het gemakkelijk maakt om dynamische kaarten te maken. [19]

Eveneens als bij Leaflet, de andere opensource JavaScript *library* zagen, levert Open Layers geen extra services bovenop de kaart. Voor extra functionaliteit op de kaart moeten er weer *third-party* plug-ins gedownload worden. Dit is en blijft een heel groot nadeel vergeleken met de Google Maps API. [19]

De documentatie van Open Layers is overzichtelijk en makkelijk om te gebruiken. Het bevat ook een handleiding voor starters, enkele tutorials en een workshop. Deze extra's heeft de Google Maps API natuurlijk ook, maar de documentatie van Open Layers is een stuk rustiger voor het oog.

Volgens SwitchToOsm is de leercurve van Open Layers matig. [20] Hieruit kan worden afgeleid dat de Open Layers niet het beste alternatief is voor de Google Maps API als er enkel gekeken wordt naar de code.

Open Layers heeft een simpele look zonder al te veel opvallende zaken. De stijl kan echter wel eenvoudig aangepast worden. Het vergt slechts enkele lijnen code. [21]

Doordat Open Layers ook weer geen extra services meelevert en er vrij weinig informatie over te vinden is, wordt het zeer moeilijk om het te implementeren in het stageproject. Daarom wordt er ook op deze tool niet dieper ingegaan.

3.1.4 Here

Here is een API die het de gebruiker makkelijk maakt om een interactieve map aan te passen naar eigen smaak en functionaliteit. Om te kaderen hoe Here uitgegroeid is tot het bedrijf dat het vandaag de dag is, wordt er gestart met een klein overzicht. Nokia Here startte enkele jaren geleden als een applicatie die enkel meegeleverd werd op Windows smartphones. Sindsdien heeft het twee naamsveranderingen ondergaan, eerst veranderde de naam naar Here Maps en vervolgens naar Here WeGo. Ondertussen is het bedrijf verkocht aan enkele Duitse autofabrikanten, namelijk, Audi, BMW en Mercedes. Sinds enkele jaren kunnen ontwikkelaars de Here API gebruiken voor het bouwen van interactieve kaarten. Here wordt geprezen voor zijn flexibele prijzen, goede documentatie en stijlvolle design. [22]

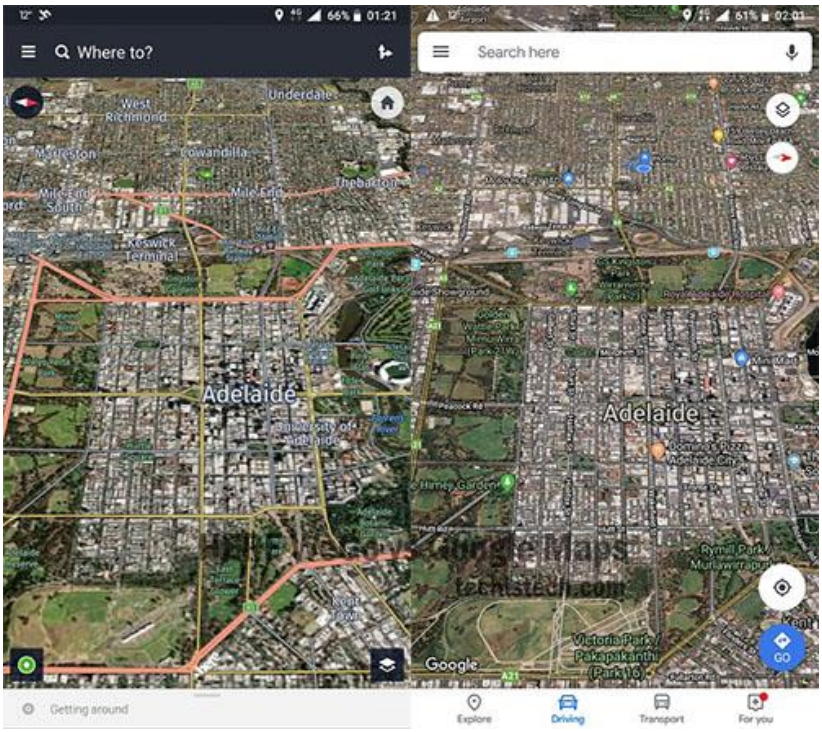
Omdat Here een waardig, goedgedocumenteerd alternatief is voor de Google Maps API en omdat het gebruikt wordt als navigatiesoftware in auto's van Audi, BMW, Mercedes en als *driver guidance* in auto's van Jaguar [22], wordt Here uitgebreider vergeleken met de Google Maps API. Er worden ook enkele verschillen in de praktijk gebracht.

Om de verschillen tussen Here en Google Maps op een duidelijke manier in kaart te brengen worden verschillende aspecten van beide kaarten met elkaar vergeleken.

Een eerste en misschien wel het meest doorslaggevende aspect wanneer er tussen één van de twee API's gekozen moet worden is de prijs. Het is nog geen jaar geleden dat Here zijn gratis gebruikers slechts vijftienduizend *requests* aanbood en hen geen toegang gaf tot de geavanceerde features. Sterker nog, men kreeg de premium mobiele-SDK's pas vanaf vierhonderdnegenenveertig euro per maand. Daar is midden 2018 verandering in gekomen. Here lanceerde het '*Freemium*'-pakket. Dit pakket zorgde voor een grote uitbreiding van de mogelijkheden voor de gratis gebruikers. Zo kunnen er nu tweehonderdvijftigduizend gratis-*requests* gestuurd worden naar de servers van Here en krijgen alle gebruikers gratis toegang tot de premium mobiele-SDK's. Wanneer de gratis-*requests* opgebruikt zijn, moet er een vast bedrag van één euro per duizend *requests* betaald worden. Indien een bedrijf ver over het maximaal aantal toegelaten gratis-*requests* zit, kan er beter overgeschakeld worden naar het 'PRO'-pakket met één miljoen *requests*. [23]

In tegenstelling tot Here, dat de prijs van zijn producten versoepelde, schroefde Google de prijzen op. Het vraagt niet meteen een grote som geld, maar alle API's van Google werken vanaf juni 2018 met een '*pay-as-you-go'*-pakket. Dit pakket zorgt volgens ontwikkelaars voor een overweldigende prijsstijging van duizendvierhonderd procent. Google geeft zijn gebruikers maandelijks tweehonderd dollar aan *requests* gratis. Dit lijkt op het eerste zicht veel, maar voor tweehonderd dollar krijgt de gebruiker, door het '*pay-as-you-go'*-pakket slechts achtentwintigduizend gratis *requests.* Het contrast met Here kan niet groter zijn. [24]

Wat al even belangrijk is als de prijs is de gebruikersinterface. Beide kaarten zijn qua interface compleet verschillend. Here gebruikt een donkerroze kleur om grote wegen aan te duiden. Als de gebruiker de kaart in satellietmodus bekijkt, is dit veel duidelijker dan het grijs dat Google gebruikt. Het grijs van Google ziet er dan weer gedetailleerder uit in de navigatiemodus. Als we het hetzelfde zoomniveau gebruiken voor beide kaarten is Google de overzichtelijkere kaart. De straten zijn goed herkenbaar en de straatnamen goed leesbaar. Hoewel de straten op de kaart van Here ook goed zichtbaar zijn, laat Here enkele namen weg van kleinere straatjes. [25]



Figuur 19: Interface Here (links) versus Google (rechts) [25]

Om te testen welke app als beste de route berekent naar de eindbestemming van de gebruiker lieten de mensen van TechIsTech beide apps de route berekenen naar eenzelfde punt. Beide kaarten houden rekening met het verkeer en geven alternatieve routes bij een eventueel ongeval. Google kwam als winnaar uit de boot. Het berekende een route die een kilometer korter en vier minuten sneller was als de route van Here. [25]

Beide kaarten hebben functioneel veel gelijkenissen, maar er is één feature die Here volgens Hannah Bouckley [26] een streepje voor geeft, namelijk het feit dat er met Here kaarten, bus-, tram- en treininformatie van 150 landen gedownload kan worden. Dit heeft als grote voordeel dat je deze kaarten offline kan gebruiken. Het nadeel hiervan is dat de kaarten veel geheugen in beslag nemen. De kaart van het Verenigd Koninkrijk is ongeveer 600MB groot. Hoewel de kaart van Google ook de functionaliteit bevat om kaarten offline te downloaden, is dit jammer genoeg enkel mogelijk voor beperktere gebieden. Meer nog, volgens Rahul Gupta van GuidingTech beschikken de gedownloade kaarten van Google over een vervaldatum [27]. De kaarten zullen dus verdwijnen binnen x-aantal tijd. Dit is bij de Here-kaarten niet het geval, een kaart blijft tot de gebruiker beslist hem te verwijderen, op het toestel staan. Rahul Gupta bekijkt, in tegenstelling tot Hannah Bouckley, het feit dat er met Google Maps enkel kleinere gebieden kunnen worden opgeslagen als een groot voordeel [27]. Rahul geeft als argument dat de gebruikers meestal toch niets hebben aan de volledige kaart van een land en dat het goed is dat de gebruikers een kleiner gebied kunnen afbakenen en vervolgens kunnen downloaden. Dit bespaart heel veel geheugen.

Om af te sluiten werd er gekeken naar extra services die Here met zich meebrengt. Volgens NearPlace is Here een uitstekend alternatief voor de Google Maps API omdat het enkele interessante features met zich meebrengt. Zo kunnen ontwikkelaars met behulp van de Here API eenvoudig realtime weerinformatie opvragen om zo de gebruiker voor te stellen om misschien wat eerder te vertrekken als het hard regent. Het is ook heel eenvoudig om via de Here-API de tolkosten en de staat van de wegen op te vragen zodat de gebruiker een duidelijk zicht krijgt op de kost en de kwaliteit die zijn rit met zich mee zal brengen. [28]

4 Uitvoering

4.1 Markers

Het is duidelijk dat zowel de Here API als de Google Maps API geschikt zijn voor het plotten van de locaties van de medewerkers op hun kaart.

Het tekenen van een marker op de kaart is met beide API's mogelijk en niet al te moeilijk. De applicatie moet echter ook vlot werken bij bedrijven met meer dan duizend werknemers. De vraag is dus; zijn beide kaarten even performant wanneer er meer dan duizend markers getoond moeten worden? Dit wordt getest aan de hand van het tekenen van een groot aantal markers op de twee kaarten, zo kunnen we zien of er een opmerkelijk verschil is tussen de Google Maps API en Here.

Op beide kaarten worden tweeduizend markers getekend. Dezelfde geografische locaties worden gebruikt voor zowel de kaart van Here als de kaart van Google.

Volgende functies werden uitgevoerd en getimed bij het opstarten van de applicatie:

```
public void draw2000Markers()
{
    for (int i = 0; i < 2000; i++)
    {
        double lat;
        double longt;

        lat = 50 + (i / 100F);
        longt = 5 + (i / 100F);

        GeoCoordinate geoCoordinate = new GeoCoordinate(lat, longt);
        MapMarker marker = new MapMarker(geoCoordinate);
        map.addMapObject(marker);
    }
}
```

Figuur 20: Functie Here

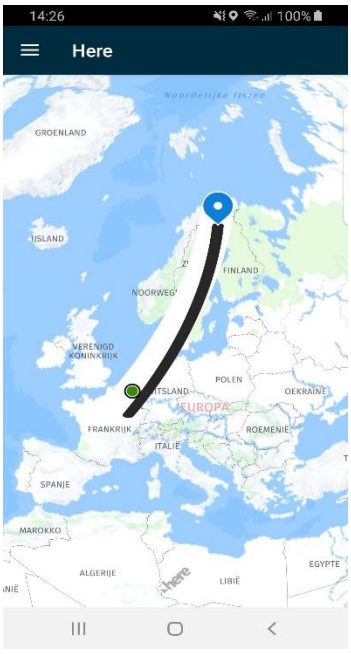
```
public void draw2000Markers() {
    for (int i = 0; i < 2000; i++)
    {
        double lat;
        double longt;

        lat = 50 + (i / 100F);
        longt = 5 + (i / 100F);

        LatLng latLng = new LatLng(lat, longt);
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        googleMap.addMarker(markerOptions);
    }
}
```

Figuur 21: Functie Google Maps

Dit was het resultaat op de kaart:



Figuur 22: Markers Here

Figuur 23: Markers Google Maps

Alle markers worden op beide kaarten vijf keer ingeladen. Op die manier ontstaat er een algemeen beeld van de snelheid van beide API's.

Tabel 3: Tekentijd markers Here versus Google Maps

HERE (in seconden)	**Google Maps** (in seconden)
4,882	0,695
3,549	1,197
3,814	1,002
4,441	1,405
5,232	0,870

Uit de resultaten blijkt dat Here, bij het tekenen van tweeduizend markers, beduidend trager is dan de kaart van Google. Bij Here duurde het soms zelfs 5 seconden om alle markers te laden, het zou heel vervelend zijn voor werknemers indien ze elke keer vijf seconden moesten wachten bij het openen van de applicatie. Google doet veel beter met een gemiddelde laadtijd van ongeveer één second. Op het vlak van opstartsnelheid heeft Google dus zeker een streepje voor.

4.2 Infoschermen

Wanneer de gebruiker op een marker van een collega tikt, is het de bedoeling dat er informatie over die bepaalde collega getoond wordt in een infoscherm. Met de Google Maps API is het heel makkelijk om zo'n infoscherm te tekenen. Het enige wat de ontwikkelaar moet doen is een lay-out ontwerpen en die koppelen aan een *InfoWindowAdapter.* De InfoWindowAdapter is een klasse die ervoor zorgt dat de zelfgemaakte lay-out geladen wordt bij het openen van een infoscherm.

Op de kaart van Google levert dit volgend resultaat:

Figuur 24: Infoscherm Google Maps

Het is een stuk moeilijker om dezelfde functionaliteit te implementeren op de kaart van Here. Het ontwerpen van een infoscherm is relatief eenvoudig en gebeurt op dezelfde manier als bij de Google Maps API. De ontwikkelaar moet ook een eigen lay-out aanmaken. Maar wanneer de ontwikkelaar een infoscherm wil tonen op de kaart is hij verplicht van dat te doen aan de hand van een *view* die hij zelf moet toevoegen aan de kaart. Sterker nog, de ontwikkelaar moet zelf bepalen op welke positie deze *view* getekend wordt. Het infoscherm wordt met andere woorden niet automatisch afgestemd op de positie van de marker. Hieruit kan geconcludeerd worden dat er geen functie bestaat die het mogelijk maakt om infoschermen op een dynamische manier te tekenen op de kaart.

Figuur 25: Infoscherm Here

4.3 Medewerkers tonen op aangepaste schaal

Nadat alle markers getekend zijn op de kaart moet de kaart zoomen op de getekende markers. Dit zorgt voor een overzichtelijk beeld van alle werknemers die aan de slag zijn. Op beide kaarten is dit relatief gemakkelijk. Er zijn ook geen verschillen qua performance waarneembaar.

```
for (MapMarker marker : markers)
{
    GeoCoordinate coordinate = marker.getCoordinate();
    double latitude = coordinate.getLatitude();
    double longitude = coordinate.getLongitude();
    minLat = Math.min(minLat, latitude);
    minLon = Math.min(minLon, longitude);
    maxLat = Math.max(maxLat, latitude);
    maxLon = Math.max(maxLon, longitude);
}

GeoBoundingBox box = new GeoBoundingBox(new GeoCoordinate(maxLat, minLon),
        new GeoCoordinate(minLat, maxLon));

ViewRect viewRect = new ViewRect(padding, padding,
        i2: map.getWidth() - padding * 2,
        i3: map.getHeight() - padding * 2);
map.zoomTo(box, viewRect, Map.Animation.BOW, Map.MOVE_PRESERVE_ORIENTATION);
```

Figuur 26: Zoom naar markers Here

```
CameraUpdate cu;
if (!list.isEmpty())
{
    LatLngBounds.Builder builder = new LatLngBounds.Builder();
    Timber.d(String.valueOf(list));
    for (LatLng position : list)
    {
        builder.include(position);
    }
    LatLngBounds bounds = builder.build();

    cu = CameraUpdateFactory.newLatLngBounds(bounds, padding);

    if (list.size() == 1)
    {
        cu = CameraUpdateFactory.newLatLngZoom(
                new LatLng(list.get(0).latitude, list.get(0).longitude),  v: 18
        );
    }

    googleMap.animateCamera(cu);
```

Figuur 27: Zoom naar markers Google

4.4 *Turn-by-turn*-navigatie via Here

Bij Google krijgen ontwikkelaars slecht één *request* per dag om een route te berekenen. Laat staan dat er *turn-by-turn*-navigatie voorzien is in kaart van Google. Het hele grote voordeel dat Here heeft ten opzichte van Google is dat navigatie gestart kan worden vanaf de kaart waarop alle medewerkers aangeduid staan.

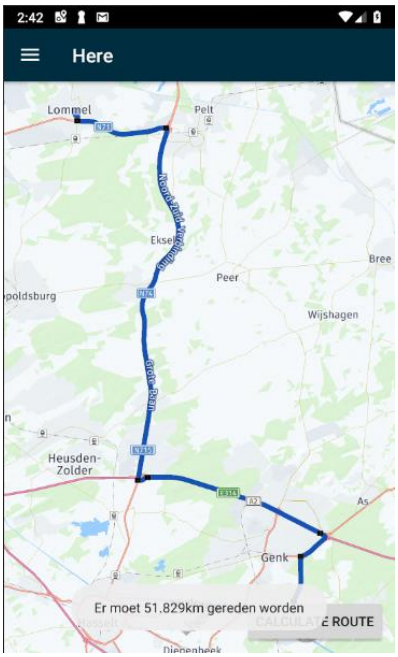
Om dit aan te tonen wordt er een voorbeeld gegeven waarin eerst de route van Lommel naar het stagebedrijf te Genk berekend wordt, en vervolgens navigatie naar de eindbestemming gestart wordt.

De route kan op een eenvoudige manier opgevraagd worden. Aan de hand van *RouteOptions* kunnen er allerlei opties ingesteld worden voor het berekenen van de route. Welk vervoersmiddel wordt er gebruikt? Wil de gebruiker over snelwegen rijden? Moet de kortste of snelste route berekend worden? Wil de gebruiker alternatieve routes te zien krijgen? Onderstaande foto geeft een verduidelijking.

```
RouteOptions routeOptions = new RouteOptions();
/* Other transport modes are also available e.g Pedestrian */
routeOptions.setTransportMode(RouteOptions.TransportMode.CAR);
/* Disable highway in this route. */
routeOptions.setHighwaysAllowed(true);
/* Calculate the shortest route available. */
routeOptions.setRouteType(RouteOptions.Type.FASTEST);
/* Calculate 1 route. */
routeOptions.setRouteCount(1);
/* Finally set the route option */
routePlan.setRouteOptions(routeOptions);
```

Figuur 28: Opties voor het berekenen van een route

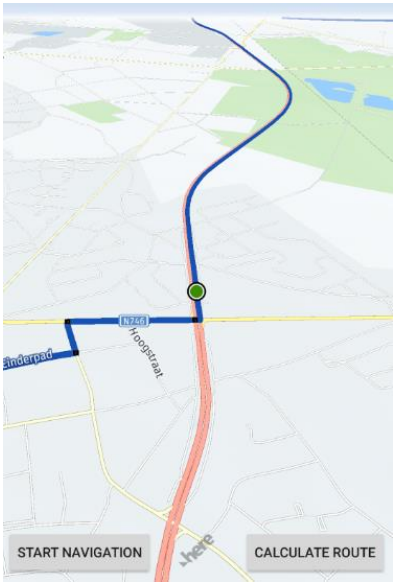
Vervolgens wordt er een *request* gestuurd naar de Here-API, er wordt een route teruggegeven die dan op de kaart getekend kan worden.



Figuur 29: Routeberekening in Here

Wanneer de route is opgevraagd kan de ontwikkelaar aan de hand van enkele lijnen code navigatie voorzien op dezelfde kaart waarop alle werknemers aangeduid staan. Voor de gebruikers van de applicatie is dit heel handig, ze hoeven geen andere navigatietools te voorzien er wordt dus ook geen onnodige tijd verloren bij het opstarten van deze tools.

Als de route beschikbaar is kan de ontwikkelaar bijvoorbeeld een knop voorzien waarmee de navigatie kan worden gestart. Vervolgens krijgt de gebruiker een ingezoomd beeld van de route, waarop zijn locatie staat aangegeven te zien, zodat hij zonder problemen op zijn bestemming kan geraken (zie figuur 30). De ontwikkelaar kan deze functionaliteit uitbreiden. Zo kan stemassistentie worden ingeschakeld en kunnen er icoontjes worden getoond bij een richtingsverandering.



Figuur 30: Ingebouwde navigatie Here

5 Conclusie

Na het bekijken van de literatuur, het onderzoeken van verschillende alternatieven voor de Google Maps API en een uitgebreide vergelijking met de Here-API, kunnen volgende zaken geconcludeerd worden.

Hoewel Here ook gebruikt kan worden voor het systeem dat GPS wil ontwikkelen en zelfs ingebouwde navigatie ter beschikking heeft, heeft het enkele beperkingen. Na het uitvoeren van een vergelijkend onderzoek dat in de praktijk werd gebracht, werd duidelijk dat de opstarttijd van de kaart van Here beduidend hoger ligt dan die van Google. Het duurde gemiddeld zelfs drie seconden langer om alle markers op de kaart te kunnen zien.

Het bleek vervolgens heel moeilijk te zijn om infoschermen toe te voegen aan markers. Hoewel de ontwikkelaar, net zoals bij de kaart van Google, zelf infoschermen kan ontwerpen, zijn deze allesbehalve flexibel. Het ontworpen infoscherm zal voor elke marker als aparte view getekend en gepositioneerd moeten worden op de kaart. Wanneer een bedrijf meer dan honderd medewerkers heeft wordt het voor de ontwikkelaar moeilijk om al deze infoschermen en bijbehorende markers te managen.

Omdat Google zijn klanten geen gratis routeberekening of navigatie biedt, en Here dit wel doet, is het zeker een optie om deze twee API's te combineren. Wanneer er enkel gebruik wordt gemaakt van de Google Maps API is de gebruiker verplicht om een externe navigatieapplicatie te installeren indien hij gebruik wil maken van routebegeleiding. Om de gebruikerservaring te verbeteren kan de routebegeleiding voorzien worden op de kaart van Here.

Ten slotte is het belangrijk om te weten welke financiële inspanning beide API's met zich meebrengen. Google schroefde vorig jaar zijn prijzen op. Developers betalen nu een bedrag naar gelang het aantal requests dat ze doen per maand. De eerste tweehonderd dollar aan requests zijn gratis. Dit komt neer op ongeveer achtentwintigduizend requests. Here versoepelde zopas zijn prijzen. Het biedt zijn klanten een '*Freemium*'-pakket. De eerste tweehonderdvijftigduizend requests zijn gratis, nadien betaald de klant één dollar per duizend requests.

De conclusie die getrokken kan worden uit dit onderzoek is dat het zeker mogelijk is om het systeem te bouwen met beide API's, maar dat de kaart van Google gebruiksvriendelijker, eenvoudiger en sneller is voor zowel de gebruiker als de ontwikkelaar. Beide kaarten kunnen gecombineerd worden voor een optimale gebruikservaring.

6 Bibliografie

[1] „Fragments,” [Online]. Available: https://developer.android.com/guide/components/fragments.html. [Geopend 29 05 2019].

[2] „ASP.NET MVC - Web API,” tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_web_api.htm. [Geopend 29 04 2019].

[3] M. Rouse, „DEFINITION JAVA,” 04 2019. [Online]. Available: https://www.theserverside.com/definition/Java. [Geopend 29 04 2019].

[4] „Java (programmeertaal),” 20 01 2019. [Online]. Available: https://nl.wikipedia.org/wiki/Java_(programmeertaal). [Geopend 29 04 2019].

[5] M. Rouse, „DEFINITION ANDROID STUDIO,” 10 2018. [Online]. Available: https://searchmobilecomputing.techtarget.com/definition/Android-Studio. [Geopend 29 04 2019].

[6] „Lottie,” [Online]. Available: https://airbnb.io/lottie/#/. [Geopend 29 04 2019].

[7] S. S. D. W. Lars Vogel, „Using Retrofit 2.x as REST client - Tutorial,” 05 06 2018. [Online]. Available: https://www.vogella.com/tutorials/Retrofit/article.html. [Geopend 29 04 2019].

[8] Pankaj, „Jackson JSON Java Parser API Example Tutorial,” [Online]. Available: https://www.journaldev.com/2324/jackson-json-java-parser-api-example-tutorial. [Geopend 29 04 2019].

[9] „Maps Static API,” [Online]. Available: https://developers.google.com/maps/documentation/maps-static/intro. [Geopend 29 04 2019].

[10] „Maps SDK for Android,” 19 12 2018. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/intro. [Geopend 29 04 2019].

[11] „Create and monitor geofences,” Android, [Online]. Available: https://developer.android.com/training/location/geofencing. [Geopend 29 05 2019].

[12] v. g. temprano, „Google Maps API or Leaflet: What's Best for your Project?,” 18 01 2017. [Online]. Available: https://www.codementor.io/victorgerardtemprano/google-maps-api-or-leaflet--what-s-best-for-your-project-faaev60vm. [Geopend 05 04 2019].

[13] „Mapbox REVIEW,” [Online]. Available: https://reviews.financesonline.com/p/mapbox/#what-is. [Geopend 29 04 2019].

[14] „Maps service,” Mapbox, [Online]. Available: https://docs.mapbox.com/api/maps/. [Geopend 29 04 2019].

[15] Zeliax, „Mapbox performance suffers after adding 1000+ markers using geojson,” 03 2018. [Online]. Available: https://stackoverflow.com/questions/44802793/mapbox-performance-suffers-after-adding-1000-markers-using-geojson. [Geopend 29 04 2019].

[16] benrudolph, „Marker performance issue,” 25 11 2017. [Online]. Available: https://github.com/alex3165/react-mapbox-gl/issues/457. [Geopend 29 04 2019].

[17] C. Kleimeier, „Switching to Mapbox: the Less Expensive Google Maps Alternative,” 22 08 2018. [Online]. Available: https://gomasuga.com/blog/mapbox-google-maps-alternative. [Geopend 29 04 2019].

[18] „Leaflet,” [Online]. Available: https://leafletjs.com/. [Geopend 10 05 2019].

[19] „A high-performance, feature-packed library for all your mapping needs.,” [Online]. Available: https://openlayers.org/. [Geopend 10 05 2019].

[20] „Getting started with OpenLayers,” [Online]. Available: https://switch2osm.org/using-tiles/getting-started-with-openlayers/. [Geopend 10 05 2019].

[21] ca0v, „Applying styles in OpenLayers 3?,” 12 04 2017. [Online]. Available: https://gis.stackexchange.com/questions/235114/applying-styles-in-openlayers-3. [Geopend 10 05 2019].

[22] „Here (company),” 04 05 2019. [Online]. Available: https://en.wikipedia.org/wiki/Here_(company). [Geopend 10 05 2019].

[23] P. Sawers, „Here launches new freemium plan for developers following Google Maps API changes,” 03 08 2018. [Online]. Available: https://venturebeat.com/2018/08/03/here-launches-new-freemium-plan-for-developers-in-response-to-google-maps-api-changes/. [Geopend 22 05 2019].

[24] I. Singh, 03 05 2018. [Online]. Available: https://geoawesomeness.com/developers-up-in-arms-over-google-maps-api-insane-price-hike/. [Geopend 23 05 2019].

[25] „HERE WeGo vs Google Maps – Comparing Two Mapping Apps,” [Online]. Available: https://www.techistech.com/here-wego-vs-google-maps/. [Geopend 23 05 2019].

[26] H. Bouckley, „Here Maps review,” 19 01 2019. [Online]. Available: https://home.bt.com/tech-gadgets/phones-tablets/here-the-free-mapping-app-that-could-replace-your-sat-nav-11363969427308. [Geopend 14 05 2019].

[27] R. Gupta, „Google Maps vs HERE WeGo: The Best Offline Navigation App,” 24 03 2018. [Online]. Available: https://www.guidingtech.com/google-maps-vs-here-wego/. [Geopend 22 05 2019].

[28] „Google Maps API alternatives,” 26 07 2018. [Online]. Available: https://nearplace.com/blog/google-maps-api-alternatives/#google_maps_vs_here. [Geopend 22 05 2019].

[29] m. kumar, „Dapper and Repository Pattern in Web API,” 2016. [Online]. Available: http://www.mukeshkumar.net/articles/web-api. [Geopend 29 05 2019].

Bijlagen

Bijlage A: Tijdsplanning

Deze bijlage bevat een document met de tijdsplanning van het eindwerk. De eerste kolom bevat alle taken, de eerste rij bevat het aantal weken. Aan de hand van de vakjes wordt aangegeven hoelang er aan een bepaalde taak gewerkt is.