Professionele Bachelor Toegepaste Informatica

Google Maps API versus alternatieven

Yoran Nelissen

Promotoren:

Frank Wynants
Carine Derkoningen

GPS nv
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Google Maps API versus alternatieven

Yoran Nelissen

Promotoren:

Frank Wynants
Carine Derkoningen

GPS nv
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Om te beginnen zou ik graag Carine Derkoningen willen bedanken. Tijdens de stage stond ze altijd klaar om te helpen en was ze steeds bereikbaar als ik met een belangrijke vraag zat. Ik ben dan ook zeer tevreden met de begeleiding die ik van haar kreeg gedurende de voorbije stageperiode.

Vervolgens zou ik ook graag Ingo Schelfhout, Frank Wynants en Nick Gaens willen bedanken. Zij hebben Jasper Heeren, ook student aan de PXL, en mij, enorm enthousiast ontvangen en ons meteen op ons gemak gesteld binnen het bedrijf. Ze stonden steeds paraat om te helpen en door middel van wekelijkse meetings bleven ze op de hoogte van ons werk. Zo konden ze ook bijsturen waar nodig zodat er geen tijd verloren ging en er geen onnodig werk gedaan werd. Bijgevolg vond ik het zeer interessant om zelf te ondervinden hoe alles verloopt binnen het bedrijf en om te zien hoe de collega's op mekaar zijn ingewerkt.

Daarnaast hebben ook Roger Lambie en Jeroen Vanderspikken een grote rol gespeeld in het regelen van deze stage. Daarom ben ik ook hun zeer dankbaar voor de kans die ik kreeg om de voorbije drie maanden aan hun stageopdracht, binnen GPS nv, te mogen werken.

Ook alle andere collega's binnen het bedrijf hebben bijgedragen tot deze supertoffe en onvergetelijke stage. Door hun enthousiasme en vriendelijkheid heerste er een zeer leuke sfeer op de werkvloer waardoor er elke dag weer wat te beleven was.

Tot slot zou ik ook nog graag mijn ouders en goede vrienden bedanken. Hun steun heeft me enorm geholpen tijdens de stageperiode, maar ook doorheen mijn volledige opleiding aan de PXL. Ze stonden altijd voor me klaar en wisten altijd precies wat ze moesten zeggen om me doorheen de moeilijke momenten te slepen.

Abstract

Ambitieuze bedrijven worden steeds groter en dat betekent dan ook dat hun personeelsbestand blijft groeien. Met onder andere consultants op de weg of mensen die van thuis uit werken, wordt het steeds moeilijker om op de hoogte te blijven van waar alle werknemers, die momenteel aan het werk zijn, zich bevinden.

GPS nv werkte een idee uit voor het ontwikkelen van een applicatie die dit probleem kan verhelpen. De applicatie wordt ontwikkeld met behulp van een web mapping API. Deze API biedt een set van klassen en functies aan zodat bepaalde stukken code, verantwoordelijk voor specifieke acties, niet steeds opnieuw geschreven hoeven te worden. In dit geval is dat dus code die het mogelijk maakt om op een eenvoudige manier een interactieve kaart weer te geven en hier bepaalde features aan toe te voegen.

De applicatie, die wordt geschreven in Java met behulp van de IDE Android Studio, is opgebouwd uit verschillende schermen. Zo wordt de gebruiker eerst gevraagd om zich in te loggen aan de hand van het unieke identificatienummer van het bedrijf. In het volgende scherm gebruikt hij daarvoor zijn eigen inloggegevens. Wanneer deze gegevens kloppen, wordt een scherm met een kaart getoond. Deze kaart bevat onder andere de locaties van de ingeboekte werknemers, de eigen locatie en verscheidene voorgedefinieerde geolocaties. Deze worden allemaal aangeduid met een marker.

Het doel van het onderzoek naar alternatieve API's is nagaan of er daadwerkelijk een applicatie ontwikkeld kan worden met dezelfde functionaliteit, maar zonder het gebruik van de Google Maps API. De applicatie is omwille van voorgaande reden dan ook opgebouwd met twee schermen die elk gebruikmaken van een andere API.

Een kort vooronderzoek wees al snel uit dat er voldoende alternatieve API's zijn die de Google Maps API kunnen vervangen. MapBox is hier een uitstekend voorbeeld van en is daarom ook de API die is geïmplementeerd in de applicatie. De applicatie bevat zo enerzijds een scherm dat is opgebouwd met de Google Maps API en anderzijds een scherm dat is opgebouwd met de MapBox API. Op deze manier zijn de aanwezige verschillen wat betreft het visuele aspect voor de eindgebruiker enerzijds en de implementatie voor de ontwikkelaar anderzijds, duidelijk en eenvoudig vergelijkbaar. Het stagebedrijf krijgt zo ook een klare kijk op de situatie en kan zelf bepalen welke API ze zullen implementeren in hun reeds bestaande applicatie.

Inhoudstafel

Lijst van gebruikte figuren

Lijst van gebruikte afkortingen

API	Application Programming Interface
ETA	Estimated Time of Arrival
IDE	Integrated Development Environment
JSON	Javascript Object Notation
OSM	Open Street Map
RFID	Radio-Frequency Identification
SDK	Software Development Kit

Lijst van technische termen

Geofencing	Het gebruikmaken van GPS of RFID technologie om een virtuele geografische grens te creëren, die het mogelijk maakt om software een *response* te laten *triggeren* wanneer een mobiel toestel dit gebied binnenkomt of verlaat.
Bugfix	De correctie van een fout in een computerprogramma of systeem.
caching	Het proces waarbij er data wordt opgeslagen in een cache. Dit is een tijdelijke plek om dingen in op te slaan.
Multipart upload	Uploaden van een enkel object als een set van meerdere delen.
Retry policy	Opnieuw sturen van een request.
JSON	Bestandsformaat dat gebruik maakt van tekst die mensen kunnen begrijpen om data objecten te verzenden.
Payload	De data die wordt verstuurd.

Inleiding

Een applicatie die het mogelijk maakt om collega's te kunnen traceren gebruikmakende van de Google Maps Application Programming Interface, was een van de stageopdrachten die GPS nv aanbood. Deze sprong meteen in het oog en nadat de PXL groen licht gaf voor deze stage, werd alles in sneltempo in orde gebracht. Gedurende drie maanden werd er getracht deze applicatie, samen met nog enkele extra features, te ontwikkelen. Bijkomstig was het onderzoek naar enkele interessante alternatieven voor de Google Maps API.

In het stageverslag volgt er een korte voorstelling van het bedrijf. Onder andere een situering van het bedrijf in een ruimer verband, de afdeling waar er aan de opdracht gewerkt zal worden komt aan bod en tot slot een korte beschrijving van de specialisaties van GPS nv. Dit wordt gevolgd door de probleemstelling en bijgevolg dan ook de doelstellingen van het project. Tot slot wordt er ook nog de tijdsplanning, die bij de start van de stage werd opgesteld, besproken. Op deze manier wordt er wat meer inzicht gegeven over de tijdsduur van bepaalde zaken en wordt het duidelijk dat de vooropgestelde planning niet altijd tot in de puntjes gevolgd kan worden.

Na het stageverslag volgt het onderzoekstopic. Hierin wordt een technische topic uitgewerkt, die in verband staat met de opdracht van het stagebedrijf. Zoals eerder vermeld zal het onderzoek dieper ingaan op de mogelijke alternatieven voor de Google Maps API. Het doel is dan ook om te onderzoeken als deze alternatieve API een gelijkwaardige oplossing zou kunnen bieden voor het probleem van het stagebedrijf. Alvorens er een antwoord geformuleerd kan worden op de hoofdonderzoeksvraag zullen er eerst enkele deelonderzoeken aan vooraf gaan. Daarna volgen de onderzoeksmethoden, de literatuurstudie en de uitvoering. Aan de hand van deze voorafgaande onderzoeken wordt er, na een korte reflectie op het volledige onderzoek, een duidelijke conclusie geformuleerd op de onderzoeksvraag.

Naar verwachting zijn er wel enkele interessante alternatieve API's die in aanmerking komen om te gebruiken in de plaats van de Google Maps API. Er zal eerst gekeken worden naar de voor- en nadelen van elk van deze alternatieven, waarna er een keuze zal worden gemaakt. Op deze manier ligt de focus van het onderzoek enkel op het beste alternatief en kan dit zo nauwkeurig mogelijk worden uitgewerkt. Er zullen ook enkele features ontwikkeld worden in de applicatie gebruikmakende van de gekozen alternatieve API. Op deze manier worden mogelijke verschillen tussen beide API's zeer duidelijk voorgesteld, zodat deze kunnen worden opgenomen in het onderzoek. Bijgevolg kan het stagebedrijf dan zelf een besluit nemen in verband met de API die ze zullen gebruiken om hun probleem op te lossen.

I. Stageverslag

1 Bedrijfsvoorstelling

GPS is een softwarebedrijf gesitueerd in Genk dat zich focust op tijdregistratie, personeelsplanning, toegangscontroles en camerabewaking. Het bedrijf werd meer dan 30 jaar geleden opgericht door Roger Lambie en beschikt ondertussen over meer dan 30 medewerkers die er alles aan doen om de doelstellingen te behalen.

Het bedrijf heeft als doel om de meest actuele en performante oplossingen in hard- en software aan te bieden om zo een optimale flexibiliteit binnen de arbeidsorganisatie van de klant te creëren. Ze streven dan ook naar marktleiderschap door hun uniek en hoogstaand totaalaanbod.

Doormiddel van hun GPS-community bevestigen ze hun innovatieve rol en bovendien stellen ze de klant centraal. Daarnaast voorziet hun lange termijn strategie in het beheersen en toepassen van de nieuwste technologie. Ook beseffen ze dat de koppeling en integratie van applicaties steeds belangrijker wordt en dus zorgt GPS voor een integratie van hun oplossingen met de reeds bestaande hard- en software van de klant. Op deze manier blijven vroegere investeringen renderen, blijft men gebruik maken van een vertrouwde omgeving en gaat men ondertussen toch een enorme efficiencywinst kunnen realiseren.

Figuur 1: Locatie van GPS nv

Na het onder de loep nemen van de eigen organisatie, nam GPS afscheid van het klassieke organogram met een managementteam. Vervolgens hebben ze zich herschikt in projectteams met allemaal een originele teamnaam en -logo.

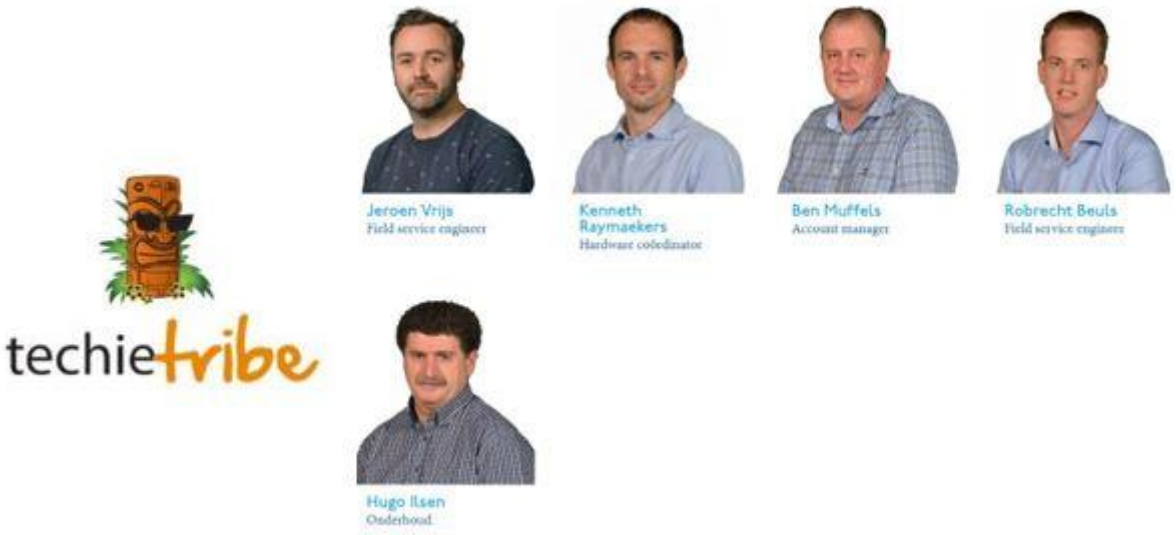
Het Road Runner team bestaat uit een combinatie van sales- en software consultancy-mensen. Zij staan continu in contact met de klant. Ze doorlopen elk project van begin tot eind en vinden het belangrijk dat de klant steeds centraal staat.

Figuur 2: Team Road Runners

Vanaf het eerste contact tot aan de oplevering garanderen de Mighty Ducks een succesvolle projectrealisatie. Het team bestaat ook uit sales- en software consultancy-mensen en ook zij vinden het belangrijk dat de klant centraal staat.

Figuur 3: Team Mighty Ducks

Het TechiTribe team is verantwoordelijk voor de verkoop, installatie en support van de hardware. Daarnaast ondersteunen ze ook de Road Runners en de Mighty Ducks met tijdregistratiehardware en werken vervolgens ook nog aan eigen projecten met betrekking tot toegangscontrole en camerabeveiliging.



Figuur 4: Team TechiTribe

Het Angry Nerds team is op hun beurt dan weer verantwoordelijk voor de ontwikkeling en de programmatie van de Emprova software en alle applicaties die hieraan gekoppeld zijn. Hun taak bestaat uit het implementeren van de door de klant gevraagde functionaliteiten.

Figuur 5: Team Angry Nerds

Het Dream team is het kleinste team, maar daarom zeker niet minder belangrijk. Zij zijn verantwoordelijk voor de algemene administratie, HR, marketing, communicatie, facturering, receptie, etc.

Figuur 6: Dream Team

2 Voorstelling stageopdracht

2.1 Probleemstelling

Momenteel telt GPS meer dan 30 medewerkers, die uiteraard niet allemaal op hetzelfde moment op kantoor zitten. Onder deze werknemers bevinden zich tal van consultants die verantwoordelijk zijn voor het onderhoud van klantrelaties en bijgevolg dus ook veel van hun tijd op de weg spenderen.

Het is op dit moment dan ook zeer moeilijk om op elke moment op de hoogte te zijn van waar elke medewerker zich bevindt. Om dit probleem te verhelpen werd er voorgesteld om een applicatie te ontwikkelen die het mogelijk maakt om collega's via geolocaties te traceren. Ter uitbreiding kunnen er nadien eventueel nog enkele features worden toegevoegd.

2.2 Doelstellingen

Het doel van het project is bijgevolg om een native app, in IOS of Android, te ontwikkelen die het mogelijk maakt om collega's te traceren gebruikmakende van de Google Maps API. Voor de opdracht wordt zowel de frontend als de backend ontwikkeld.

De applicatie, die ontwikkeld wordt in Java met behulp van de IDE Android Studio, bestaat uit meerdere schermen. Zo moet de gebruiker de mogelijkheid hebben om zich in te loggen met de gegevens van zijn/haar bedrijf en vervolgens met zijn/haar persoonlijke gegevens. Indien dit lukt, gaat er een scherm open waar er een kaart wordt weergegeven met daarop de locaties van alle collega's. Hierna hebben de gebruikers ook enkele functies waarvan ze gebruik kunnen maken. Zo is er bovenaan het scherm een zoekbalk voorzien waarmee er gezocht kan worden naar collega's. Op het moment dat er gezocht wordt, zullen de markers van alle andere collega's verdwijnen en wordt er ingezoomd op de boekingen van de gezochte collega. Daarnaast heeft de gebruiker ook nog de mogelijkheid om extra informatie te verkrijgen over de boekingen van collega's. Dit doen ze door simpelweg op de marker te drukken waarna er een infovenster opengaat. Onderaan het scherm wordt een uitschuifbaar menu voorzien. Dit voorziet de gebruiker van de mogelijkheid om te filteren op de verschillende soorten registraties. Afhankelijk van de knoppen die zijn ingedrukt worden enkel de job-, tijd- of toegangsregistraties op de kaart weergegeven. Op dezelfde manier kan er ook gefilterd worden op de verschillende geolocaties. Tot slot wordt er nog een grotere knop voorzien die als resetknop gebruikt kan worden. Als er hierop wordt gedrukt, worden alle collega's weer op de kaart weergegeven met al hun registraties. Ook worden alle geolocaties weer getoond.

De gegevens die nodig zijn om dit te realiseren zijn reeds in het bezit van het stagebedrijf. De webservice vraagt aan een bestaande database de actuele gegevens van collega's op en stuurt deze in JSON-formaat door naar de *mobile client*.

Ter uitbreiding van het project kan er nog worden gewerkt aan enkele extra features die handig zijn voor de gebruiker. Zo zou er, voor collega's die elkaar willen ontmoeten, een centrale plaats kunnen worden berekend om af te spreken, zodat de verplaatsing voor beiden ongeveer even ver is. Vervolgens zou er een feature ontwikkeld kunnen worden die de gebruiker op de hoogte brengt als het tijd is om te vertrekken naar de volgende afspraak zodat hij/zij op tijd aankomt.

Er wordt ook gewerkt met *geofencing*. Dit betekent dat de gebruiker in bepaalde, vooraf ingestelde zones een notificatie krijgt van de applicatie. Via deze notificatie krijgt de gebruiker de mogelijkheid om zich in te boeken op deze locatie. Om dit mogelijk te maken wordt er dus ook een scherm voorzien waarmee de gebruiker zich kan inklokken. Aangezien er een verschil is tussen een tijdsregistratie en een jobregistratie, worden dit dus ook twee verschillende schermen.

Tot slot wordt er voor het onderzoek ook een scherm voorzien waarin er features worden ontwikkeld met de alternatieve API.

2.3 Tijdsplanning

Zoals te zien op de afbeelding in bijlage A, is er bij de start van de stage een duidelijke planning opgesteld. Deze planning werd gebruikt als leidraad gedurende deze stageperiode en was dan ook zeer handig om alle deadlines te behalen. Zoals verwacht, was het niet gemakkelijk om deze planning consequent te volgen en werd er hier soms dus wel eens van afgeweken. In een groot project is het dan ook zeer normaal dat er taken bijkomen of dat er belangrijke *bugfixes* moeten gebeuren alvorens er verder gewerkt kan worden.

2.4 Beschrijving gebruikte technologieën

2.4.1 Java

Java is een objectgeoriënteerde programmeertaal en is ook platformonafhankelijk. Op het gebied van syntaxis is deze taal grotendeels gebaseerd op de eveneens objectgeoriënteerde programmeertaal C++. Java beschikt echter wel over een uitgebreidere klassenbibliotheek dan C++.

2.4.2 Android Studio

Android studio is de officiële *Integrated Development Environment* (IDE) voor het Android platform. Het is gebaseerd op de Jetbrains IntelliJ IDEA software en is specifiek ontwikkeld voor Android Development.

2.4.3 Client libraries

Doorheen het project wordt er gebruikt gemaakt van Retrofit. Dit is een *client library* die het mogelijk maakt om op een zeer eenvoudige wijze interfaces te creëren. Hierdoor kunnen er vervolgens dan ook zeer krachtige applicaties ontwikkeld worden.

Client libraries, ook wel helper *libraries* genoemd, zijn stukken code die applicatieontwikkelaars kunnen toevoegen aan hun projecten. Deze stukken code heeft de applicatie nodig om te kunnen communiceren met de API. De code die nodig is om een HTTP-request te creëren en de response van de API om te verwerken wordt op deze manier voorzien, zodat de ontwikkelaar dit niet zelf moet schrijven. De *libraries* omvatten ook klassen die overeen komen met de elementen of datatypes die de API verwacht en ze kunnen gebruikersauthenticatie en -autorisatie afhandelen.

2.4.4 JSON parsing libraries

Parsing libraries zijn *Java libraries* die nodig zijn bij de serialisatie en deserialisatie van Java Objecten van en naar JSON. Ze gelijken zeer sterk op mekaar, maar gebruiken andere terminologieën om hetzelfde te bereiken.

3 Uitwerking stageopdracht

3.1 Beschrijving onderzoek naar libraries

Alvorens er code geschreven kon worden, was het belangrijk om een onderzoek te starten naar het verschil tussen enkele *REST client libraries*. Na wat opzoekwerk werd al snel duidelijk dat er een keuze gemaakt moest worden tussen Volley of Retrofit. Bijgevolg werd er dan ook gezocht naar de voor- en nadelen van beide libraries zodat er daarna een besluit getrokken kon worden.

Mogelijkheden met de Volley library:

- Zeer uitgebreid en flexibel *caching mechanism*:
    - Wanneer er door Volley een *request* wordt gemaakt, wordt eerst de cache nagekeken voor een gepaste response. Indien deze wordt gevonden, wordt deze teruggegeven, zo niet wordt er een *API-request* over het netwerk gestuurd.
- Ondersteunt zowel *POST-requests* als *multipart uploads*, maar hiervoor moet nog wat extra gecodeerd worden.
- Er kan een *retry policy* worden voorzien die onder andere de *request timeouts* en aantal *retries* ondersteunt.
- Kan automatisch vier verschillende response-types opvangen aan de hand van volgende *requests*: *StringRequest, JsonObjectRequest, JasonArrayRequest, ImageRequest*
- Ingebouwde ondersteuning om afbeeldingen te laden aan de hand van een aangepaste view die speciaal hiervoor werd ontworpen

Mogelijkheden met de Retrofit library:

- Ondersteunt geen *caching*
- Volledige ondersteuning van post-requests en *multipart uploads*
- Ondersteunt geen *retrying mechanism*, maar dit kan manueel, met wat extra code, wel zelf worden bereikt
- Kan veel meer verschillende response-types automatisch verwerken
- Geen ondersteuning voor het inladen van afbeeldingen

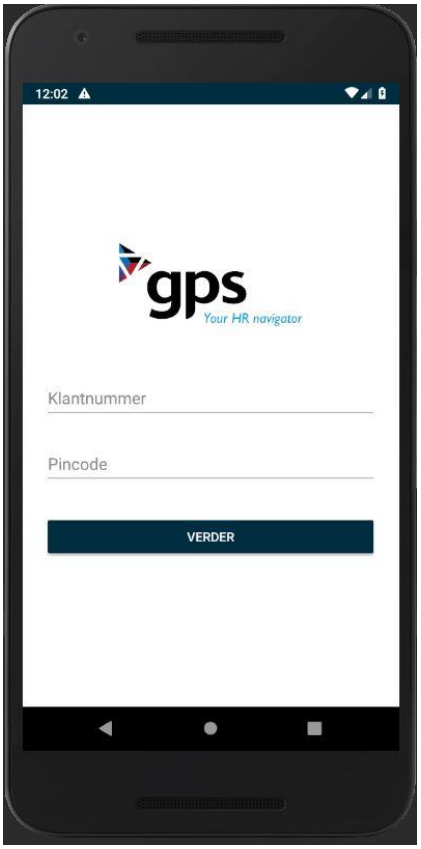
Het grootste nadeel van Volley is dat er voor de ontwikkelaars gedetailleerde en officiële documentatie ontbreekt, waardoor er soms al eens tijd verloren kan gaan door naar een oplossing te zoeken. Volley focust zich vooral om aan alle netwerkbehoeften voor Android te voldoen.

Retrofit aan de andere kant is een handige, simpele en lichte library om mee te werken en is een stuk eenvoudiger om te configureren. Het doel van deze library is dan ook om de *RESTful webservices* zo eenvoudig mogelijk te kunnen gebruiken. Een ander voordeel van Retrofit is dat het altijd up-to-date blijft, terwijl Volley nog kan vertrouwen op een library gebundeld met het besturingssysteem van de gebruiker.

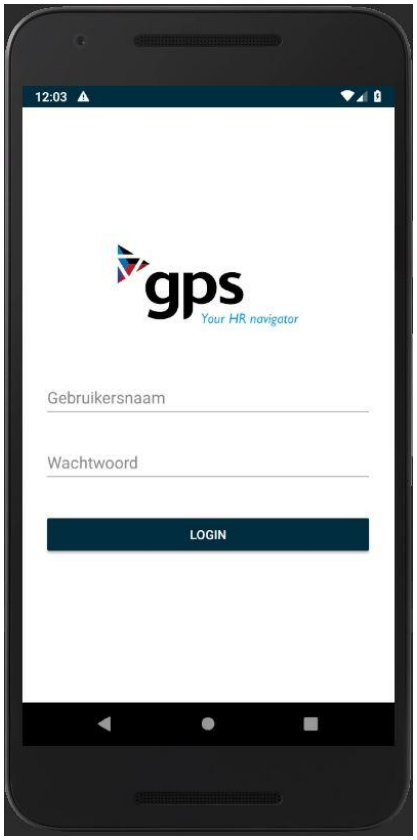
3.2 Beschrijving uitwerking schermen

Na het onderzoek naar de libraries kon de applicatie 'from scratch' ontwikkeld worden. Voor de aanvankelijke opdracht werd er verwacht dat er drie opeenvolgende schermen ontwikkeld werden. Daarna werden er extra opdrachten en features toegevoegd.

In het eerste scherm krijgt de gebruiker de mogelijkheid om zich in te loggen met het klantnummer en de pincode van het bedrijf. Indien de gebruiker dit doet met de correcte gegevens, komt er een tweede scherm tevoorschijn. Hier kan er worden ingelogd met de gebruikersnaam en het overeenkomstige wachtwoord van de gebruiker.



Figuur 7: Inloggen met klantnummer



Figuur 8: Inloggen met gebruiksnaam

Indien de applicatie wordt afgesloten en dezelfde gebruiker deze later opnieuw opent, is het niet nodig dat hij zijn gegevens opnieuw ingeeft. De gegevens van de gebruiker worden namelijk bijgehouden in de *SharedPreferences*. Dit biedt de ontwikkelaar de mogelijkheid om een kleine collectie van belangrijke informatie op te slaan. Het *SharedPreferences*-object wijst zowaar naar een file waar data kan worden bijgehouden en voorziet daarnaast ook simpele methodes om data op te kunnen halen en weg te schrijven. Figuur 9 bevat het stuk code waarmee er wordt nagegaan als alle nodige data reeds aanwezig is om de gebruiker automatisch in te loggen. Indien dit niet het geval is, krijgt de gebruiker terug het eerste inlogscherm te zien..

```
/**
 * 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 9: Controleren indien er automatisch kan worden ingelogd

Figuur 10: Automatisch inlogscherm

Het derde scherm dat werd ontwikkeld was het scherm met de Google Maps kaart. Oorspronkelijk werden hier alle originele boekingen van alle werknemers van het bedrijf op weergegeven. De communicatie voor het ophalen van de nodige data, gebruik makende van JSON, gebeurde via reeds bestaande *webservices*. Na het onderzoek, dat eerder werd uitgevoerd, werd er dan ook beslist om hiervoor gebruik te maken van Retrofit. Zoals in bijlage B duidelijk wordt gemaakt, voorzag het bedrijf hiervoor ook de nodige JSON-request en response *payloads.*

```
/**
 * Do get all selections call.
 *
 * @param url     url
 * @param headers the headers
 * @return the call
 */
@GET
Call<SelectionResponse> doGetAllSelections(@Url String url, @HeaderMap Map<String, String> headers);
```

Figuur 11: Call in de APIInterface

```
/**
 * Do a get to the selection service.
 *
 * @param context the context
 * @param dataManagerCallback callback to get the answer on
 */
/checkstyle:DescendantToken/
public static void getSelections(Context context,
                                 DataManagerCallback<SelectionResponse> dataManagerCallback)
{
    APIInterface apiInterface = APIClient.getClient().create(APIInterface.class);

    Call<SelectionResponse> getSelectionsCall = apiInterface.doGetAllSelections(SharedPreferenceManager.getSelectionsUrl(context),
            SharedPreferenceManager.getHeaders(context));

    getSelectionsCall.enqueue(new Callback<SelectionResponse>(context)
    {
        @Override
        public void onResponse(Call call, Response response)
        {
            Log.d( tag: "Selection Succes",  msg: "onResponse: " + response.body());
            dataManagerCallback.onResponse((SelectionResponse) response.body());
        }

        @Override
        public void onFailure(Call call, Throwable t)
        {
            Toast.makeText(context,  text: "Kon de clockings niet ophalen.", Toast.LENGTH_SHORT).show();
            Log.d( tag: "Selection Failure",  msg: "onFailure: " + t.getMessage());
            dataManagerCallback.onResponse(null);
        }
    });
}
```

Figuur 12: DataManager

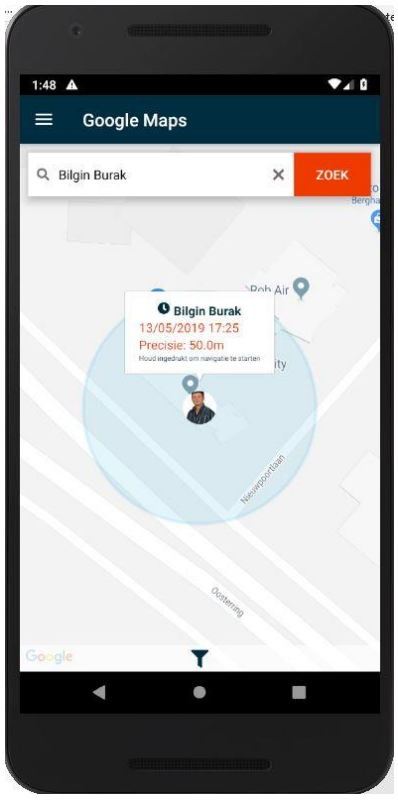
```
/**
 * Calls doGetAllSelections from APIInterface.
 */
private void apiGetSelections()
{
    loadingCircle.setVisibility(View.VISIBLE);
    loadingLayer.setVisibility(View.VISIBLE);
    searchBar.setVisibility(View.GONE);
    bottomSheet.setVisibility(View.GONE);

    SelectionsDataManager.getSelections(context, response ->
    {
        if (response != null)
        {
            markOriginalClockings(response);
        }
    });
}
```

Figuur 13: Uitvoeren call

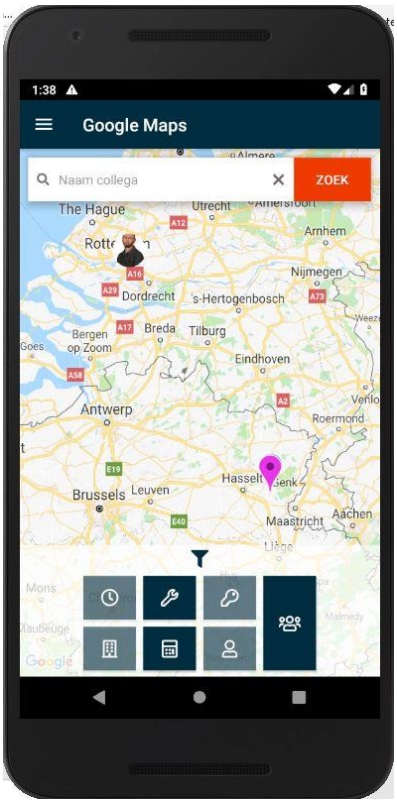
De voorgaande figuren geven weer hoe de API calls gebeuren. Er wordt een URL meegegeven samen met enkele headers.

Met voorgaande call worden alle originele boekingen opgehaald. De data die wordt teruggegeven, indien de call succesvol werd uitgevoerd, bevat o.a. de coördinaten van de locatie van waar de boeking werd geregistreerd. Met deze informatie kunnen er op de kaart markers geplaatst worden met daar rond een cirkel die de accuraatheid weergeeft.

Figuur 14: Boekingen & Geolocaties



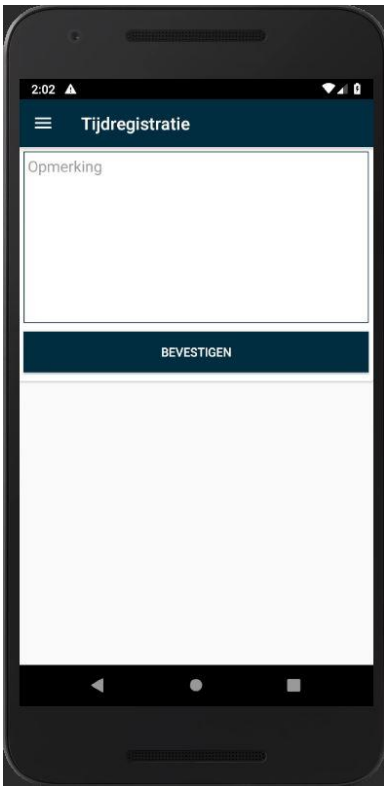
Figuur 15: Zoeken naar collega



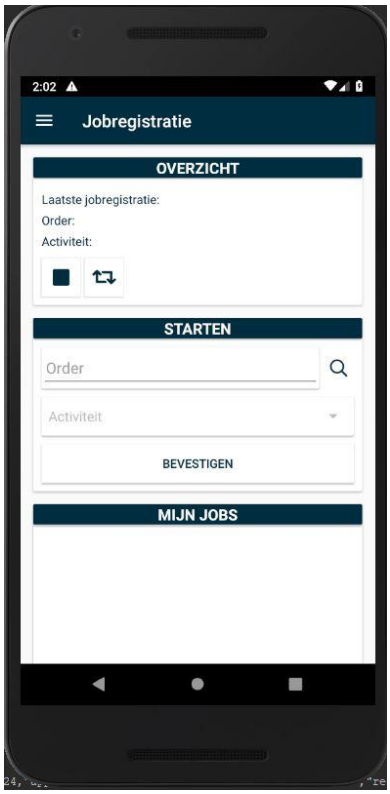
Figuur 16: Filteren van boekingen & geolocaties

Figuur 14 geeft weer hoe het derde scherm er oorspronkelijk uitzag. De gebruiker krijgt een kaart te zien met daarop de vooraf ingestelde geolocaties, gemarkeerd in het blauw, en de originele boekingen van zijn collega's, gemarkeerd met hun profielfoto. Later werden er nog wat extra features toegevoegd. Zo is op figuur 15 namelijk te zien dat er gezocht kan worden naar een collega. De boekingen van andere collega's zullen verdwijnen en er wordt ingezoomd op de boeking van de gezochte persoon. Hier is ook duidelijk de cirkel te zien die weergeeft hoe accuraat de locatie wordt bepaald. De gebruiker heeft ook de mogelijkheid om op de marker/profielfoto te drukken, waarna een infovenster open zal gaan. Hierin staat het tijdstip waarop de boeking werd geregistreerd, alsook de accuraatheid in meters. Het icoontje naast de naam van de persoon geeft dan weer aan over welk type registratie het hier gaat. Verder heeft de gebruiker ook nog de optie om het infovenster ingedrukt te houden waarna de route naar deze locatie berekend en uitgestippeld zal worden. Tot slot is op figuur 16 te zien dat er ook gefilterd kan worden. Zo koos de gebruiker er op het voorbeeld voor om enkel de jobregistraties weer te geven op de kaart en wordt er ook enkel de locatie van de terminals gemarkeerd.

Een volgende uitdaging was het implementeren van geofencing. Het zou namelijk zeer handig zijn dat de gebruikers van de applicatie de mogelijkheid hadden om een boeking te registeren op het moment ze zich kort bij een vooraf ingestelde geolocatie bevinden. Om dit te kunnen testen werden er nog twee extra schermen toegevoegd.



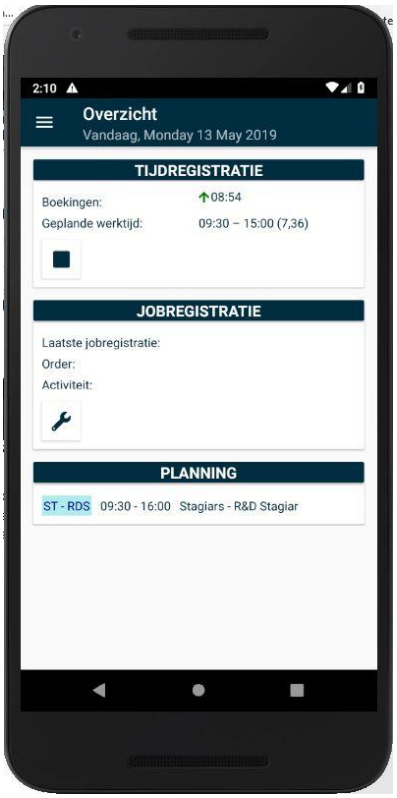
Figuur 17: Scherm tijdregistratie



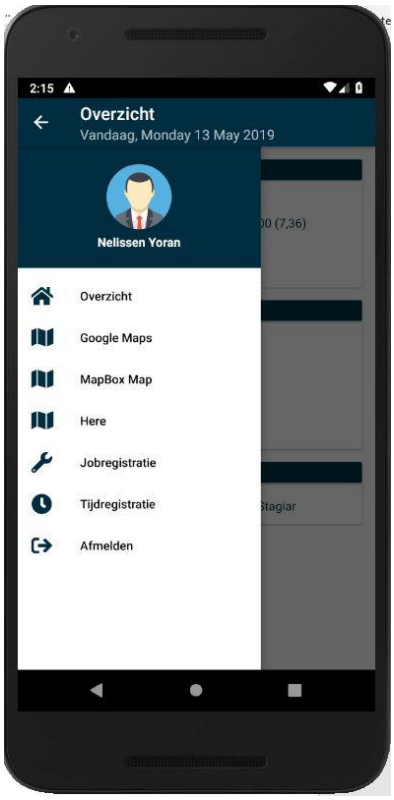
Figuur 18: Scherm jobregistratie

Figuur 19: Notificaties

Via deze schermen kan de gebruiker een tijdregistratie of een jobregistratie toevoegen. Via de notificaties die worden ontvangen, bij het in de buurt komen van een geolocatie, kan de gebruiker vanaf het startscherm van zijn toestel rechtstreeks een tijdregistratie doen. Indien hij graag een jobregistratie wil registreren kan hij via de notificatie meteen het juiste scherm openen zodat er geen tijd verloren gaat.



Figuur 20: Overzicht

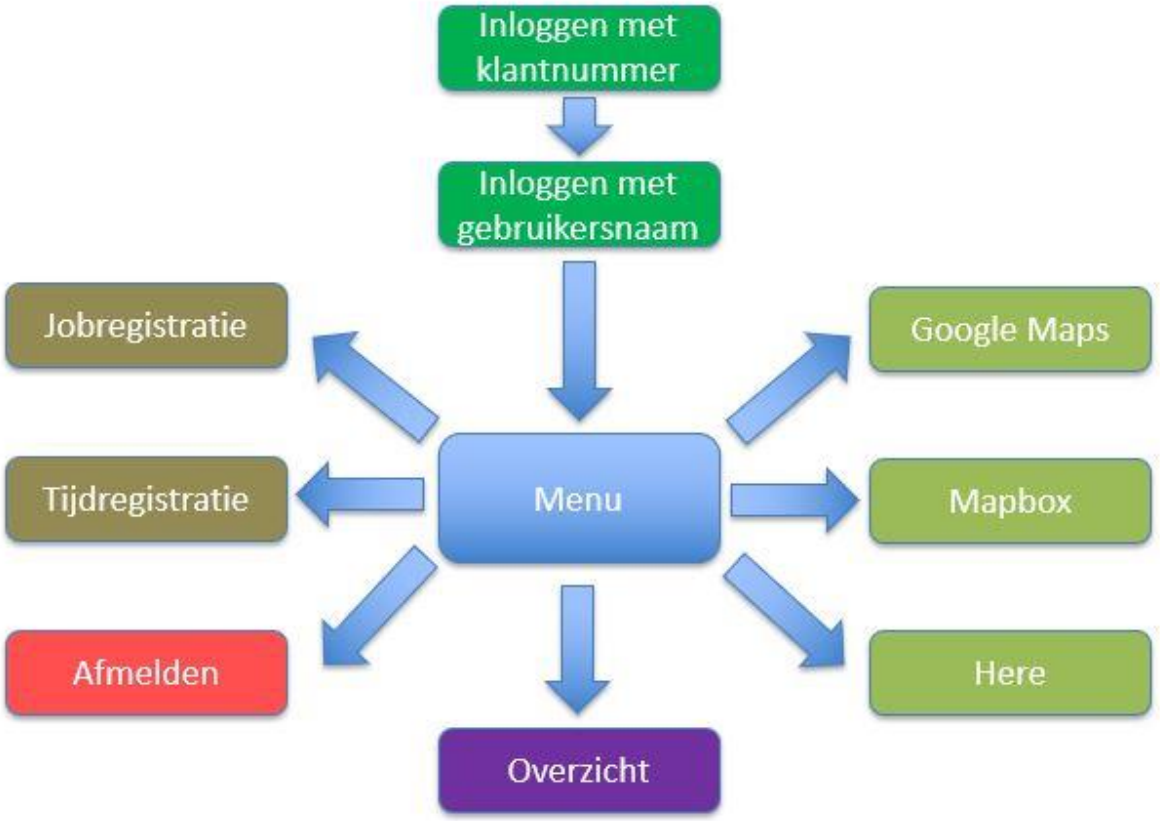


Figuur 21: Menu

Op figuur 20 is het laatste scherm te zien. Dit scherm geeft een overzicht weer van de registraties en van de dagplanning van de gebruiker. Oorspronkelijk kreeg de gebruiker meteen de kaart te zien na het inloggen, maar er werd beslist om het overzicht als startpagina weer te geven. Via het uitschuifbare menu, te zien op figuur 21, kan de gebruiker navigeren tussen alle verschillende schermen.

3.3 Resultaat

Het resultaat bestaat uit een applicatie waarmee de gebruiker de locaties van zijn/haar collega's kan opvolgen. De gebruiker kan zich inloggen met de bedrijfsgegevens en in een volgend scherm met zijn persoonlijke gegevens. In het scherm dat er opengaat wanneer het inloggen succesvol is gelukt, krijgt de gebruiker een overzicht van alle registraties te zien. Via een menu kan er worden genavigeerd tussen de verschillende schermen, namelijk het overzicht, de kaarten, de jobregistratie en de tijdregistratie. Uiteindelijk heeft de gebruiker ook nog de mogelijkheid om zich af te melden als de werkdag erop zit.



Figuur 22: Overzicht van schermen

Figuur 22 geeft duidelijk weer welke opties de gebruiker allemaal heeft.

4 Reflectie

Ik vond het persoonlijk een zeer interessant project om aan te werken en heb er veel van bijgeleerd. Het verliep zoals het in het echte bedrijfsleven er ook aan toe gaat. Zo was er eerst een kort onderzoek nodig om te beslissen met welke technologieën er gewerkt ging worden. Uiteraard verliep het ook niet altijd even vlot en waren er al eens momenten waarop het misliep. Gelukkig kon er op deze momenten gerekend worden op de collega's binnen het stagebedrijf. De sfeer op de werkvloer was overigens ook super en iedereen is er altijd bereid om elkaar een handje te helpen bij eventuele problemen.

De eerste weken was het een beetje zoeken naar hoe we het zouden aanpakken. Er werd eerst wat onderzoek gedaan en eens we beslist hadden welke libraries gebruikt gingen worden, kon het coderen beginnen. We kregen een lijst van features die we konden implementeren en hebben ons hierop gebaseerd om een tijdsplanning op te stellen. Deze planning hebben we elke week redelijk goed kunnen volgen waardoor we nooit in de problemen kwamen bij het behalen van de deadlines. Over het algemeen ondervond ik weinig problemen tijdens de stage. Ik heb geleerd dat niet alles even gemakkelijk is om te implementeren en dat er soms features zijn waar lang aan gewerkt wordt om dan tot het besluit te komen dat het toch niet op die manier zal lukken. Ik ben zeer tevreden over de verschillende mogelijkheden die de gebruiker van de applicatie heeft, maar ik vind het spijtig dat we geen clustering hebben kunnen implementeren. De mogelijkheid om hiermee te werken is er zeker, maar op het moment waarop we dit wilden implementeren hadden we al redelijk veel code geschreven en werkte de applicatie al zoals het hoorde. We zouden dan ook enorm veel code moeten herschrijven als we met clustering wilden werken en hier was spijtig genoeg geen tijd meer voor. Momenteel staan er veel markers over mekaar, hierdoor is het soms moeilijk om te kunnen zien welke collega's er allemaal zijn ingeboekt als deze kort bij elkaar zitten. Dit is zeker iets waar ik meer tijd zou insteken als er verder zou gewerkt worden aan dit project.

Naar mijn mening was het ook leuk om samen met een andere student, namelijk Jasper Heeren, deze stage te kunnen doen. Bij problemen konden we eerst hulp aan mekaar vragen en zelf een oplossing proberen te zoeken alvorens er hulp gevraagd werd aan de mensen van het bedrijf. Ik ben dan zelf ook zeer tevreden over het resultaat en over de ontwikkelde applicatie.

II. Onderzoeksopdracht

1 Onderzoeksvraag en hypothese

Kunnen alternatieven voor de Google Maps API gelijkwaardige oplossingen bieden voor de problemen van het stagebedrijf?

Om een antwoord te kunnen formuleren op deze hoofdonderzoeksvraag, zijn er eerst enkele deelvragen die onderzocht moeten worden. Bestaan er effectief alternatieven voor de Google Maps API? Als deze er zijn, welke zijn dit?

De hypothese van bovenstaande onderzoeksvraag is dat er zeker enkele alternatieven voor de Google Maps API te vinden zijn. Dit wordt verwacht aangezien het stagebedrijf dit onderzoek voorstelde en omdat ze zelf gebruik maken van zo'n alternatief, genaamd OSM. Indien er meerdere bruikbare alternatieven gevonden worden, wordt er eentje uitgekozen en volledig uitgewerkt.

Kan deze alternatieve API geïntegreerd worden in de applicatie die wordt ontwikkeld? Het doel is om uiteindelijk een applicatie te ontwikkelen waarin de gebruiker kan wisselen tussen enerzijds het Google Maps scherm en anderzijds het scherm dat wordt ontwikkeld aan de hand van de alternatieve API.

De hypothese van deze deelvraag is dat er geen problemen zijn met het implementeren van de alternatieve API in de applicatie. Er vindt wel eerst een nieuw onderzoek plaats om te leren hoe er met de andere API gewerkt kan worden. Aangezien ze beide langs mekaar in dezelfde applicatie worden opgenomen, zijn de verschillen duidelijk merkbaar. Dit heeft dan ook weer een zeer positieve invloed op het onderzoek. Er kan op deze manier namelijk een duidelijk onderscheid gemaakt worden tussen de twee en ook worden alle voor- en nadelen duidelijk zichtbaar. Hoogstwaarschijnlijk zullen er enkele features op beide kaarten zijn die niet op dezelfde manier geïmplementeerd kunnen worden, maar waarmee dus wel duidelijk wordt welke API het beste is voor die feature.

Alle features worden uitgewerkt op de Google Maps kaart, maar welke van deze features zijn ook mogelijk om uit te werken met behulp van de alternatieve API? Om het onderzoek zo accuraat mogelijk te houden, is het handig om zoveel mogelijk features met beide API's te ontwikkelen. Met veel features kan de snelheid en performantie van de applicatie eenvoudig gecontroleerd worden. Uiteraard moet de werkwijze van applicatie hiervoor dan ook zo duidelijk mogelijk zijn voor de gebruiker.

Het aantal features dat zowel met de Google Maps API als het alternatief ontwikkeld kunnen worden, is volledig afhankelijk van de mogelijkheden die de alternatieve API biedt om mee te werken. Er wordt verwacht dat de Google Maps API zeer veel mogelijkheden biedt om verschillende features op een zo eenvoudig mogelijke manier uit te werken en dat er bijgevolg altijd wat meer onderzoek nodig zal zijn om dezelfde feature met de alternatieve API te ontwikkelen.

Tot slot wordt er onderzocht of de gebruiker van de applicatie een verschil opmerkt. Zoals eerder aangehaald, verschilt de code voor beide API's op enkele plekken van mekaar. De gebruiker kan dit natuurlijk niet zien, maar kan wel andere verschillen opmerken die aan de aandacht van ontwikkelaar ontsnapt zijn.

Zo zou het perfect mogelijk kunnen zijn dat de lay-out van de kaarten van beide API's verschillend is, de pijltjes die de locaties aangeven zouden een andere lay-out kunnen hebben, de kaart inladen gaat sneller bij een van beide, enz. Mogelijks zijn dit geen verschillen die ervoor zorgen dat de applicatie minder functioneel zal zijn, maar het is altijd interessant om de feedback van gebruikers in het achterhoofd te houden.

Na dit onderzoek is er genoeg informatie om een duidelijk antwoord te kunnen formuleren op de hoofdonderzoeksvraag. Er is zeer veel informatie en documentatie te vinden over de Google Maps API, aangezien deze zeer bekend is en bijgevolg ook wereldwijd gebruikt wordt. De alternatieven zijn hierdoor veel minder bekend waardoor het ook moeilijker is om documentatie hierover te vinden. Verder wordt dan ook verwacht dat de Google Maps API veel eenvoudiger is om mee te werken en veel meer basisfunctionaliteiten aanbiedt, waardoor er enerzijds veel vlotter en minder code geschreven moet worden en anderzijds toch het verwachte resultaat wordt behaald. Dit is dan ook meteen de reden voor de verschillen die de gebruiker zal opmerken tussen beide schermen. Deze hebben echter geen invloed op de gebruikerservaring.

2 Onderzoeksmethoden

Om antwoorden te verkrijgen op de verschillende deelvragen, en uiteindelijk op de hoofdonderzoeksvraag, worden er een aantal experimenten opgezet en uitgevoerd die meer duidelijkheid brengen in het onderzoek.

Allereerst is het belangrijk om te weten of er alternatieven bestaan. Dit wordt onderzocht aan de hand van een literatuurstudie. Als er geen alternatieven bestaan, is dit ook meteen het einde van het onderzoek. Anders wordt er een lijst opgesteld van alle mogelijke API's die de Google Maps API zouden kunnen vervangen. Daarna wordt er een keuze gemaakt en volgt er een uitgebreider onderzoek naar de gekozen API. Wanneer de keuze is gemaakt en het onderzoek heeft uitgewezen dat de API een degelijk alternatief zou kunnen zijn, wordt deze geïmplementeerd in de applicatie.

In een volgende stap wordt er gekeken welke features met zowel de Google Maps API als de alternatieve API ontwikkeld kunnen worden. De applicatie bevat een scherm met een kaart van Google Maps. In dit scherm worden alle features, in verband met de Google Maps API, ontwikkeld. Daarna wordt er bij elke feature onderzoek gevoerd naar de manier waarop deze geïmplementeerd zou kunnen worden met de alternatieve API. Als er voldoende documentatie wordt gevonden, wordt dezelfde feature ontwikkeld met de alternatieve API. Features waarvan niet zeker is dat ze afwerkt kunnen worden, worden overgeslagen en worden bijgevolg niet mee geïmplementeerd in het scherm van de alternatieve API.

In een volgend experiment wordt er gevraagd aan de gebruikers, mensen met de ontwikkelde applicatie op hun smartphone, om deze applicatie te testen. Hierbij kan de gebruiker feedback geven over de gebruiksvriendelijkheid van de applicatie enerzijds en over de verschillen die hij/zij opmerkt tussen de schermen ontwikkeld met Google Maps en het alternatief anderzijds.

Een nog uitgebreider onderzoek gebeurt, zoals eerder al vermeld, aan de hand van een literatuurstudie. Er kunnen bij dit deel van het onderzoek vergelijkingsmatrices worden opgesteld die de verschillen en voor- en nadelen duidelijk maken. De feedback van testpersonen en de informatie gevonden in andere onderzoeken rond dit onderzoekstopic speelt dan ook een grote rol om tot een conclusie voor de hoofdonderzoeksvraag te komen.

3 Onderzoek API's

3.1 Alternatieven

Voor veel ontwikkelaars is de Google Maps API een favoriet om te gebruiken als het gaat over geolocatieservices. Met hun uitgebreide database aan geografische features en *streetviews*, zijn ze daarom ook al jarenlang de eerste keuze van ontwikkelaars. Ondanks de grote tevredenheid van alle bedrijven, besloten ze bij Google om hun prijs op te drijven waardoor gebruikers nu al veertienmaal de oorspronkelijke prijs betalen [1]. Voor grote bedrijven is dit geen probleem, maar voor kleinere ontwikkelaars is deze prijsstijging misschien wel een reden om eens op zoek te gaan naar alternatieven voor de Google Maps API.

3.1.1 Here

Een eerste gevonden alternatief is Here. Volgens Counterpoint Research is Here "de onbetwiste leider in het locatie-ecosysteem". Ze bieden dezelfde features aan als alle andere alternatieven en kunnen daarnaast ook nog eens een uitgebreide kaartvisualisatiefunctionaliteit aanbieden. Wanneer Here ontstond kregen gebruikers 15000 transacties/maand of met andere woorden, gebruikers konden elke maand slechts 15000 requests sturen naar de Here Servers. Daarnaast was er ook nog geen toegang tot meer geavanceerde features en de mobiele SDK [2]. Uiteindelijk kon Here, na wat updates en enkele prijswijzigingen, de concurrentie aangaan met de Google Maps API. Jasper Heeren, ook stagiair bij GPS nv, onderzoekt dit alternatief en kan de verschillen tussen de Google Maps API en Here uitgebreider opsommen en voorstellen.

3.1.2 OpenLayers

Een volgende mogelijkheid is OpenLayers. Ontwikkelaars kunnen deze *opensource mapping library* gratis gebruiken om dynamische kaarten weer te geven in een applicatie [3]. Onderzoek wijst uit dat deze API-tegels uit verschillende andere bronnen, zoals OSM, haalt en daar een kaart mee opbouwt. Enkele andere voordelen van OpenLayers is dat er op zeer eenvoudige wijze markers geplaatst kunnen worden en dat er kan gewerkt worden met *vector layers*. Dit zijn lagen waarop vectorobjecten, zoals lijnen, vormen en andere figuren, worden opgeslagen en waarop deze niet zijn gekoppeld aan vaste pixels. Deze verschillende *layers* kunnen onzichtbaar of transparant gemaakt worden zodat de ontwikkelaar een originele animatie aan zijn kaart kan toevoegen.

3.1.3 Mapfit

Een volgend alternatief is Mapfit. Deze mapping API heeft, in tegenstelling tot de andere alternatieven, een zeer specifiek doel. Er wordt namelijk een zeer accurate kaart aangeboden om mee te werken. Zo kunnen er zelfs deuren, laadperrons en garages herkend worden. Met Mapfit kan er elke maand gebruikgemaakt worden van 50000 gratis requests of heeft men voor $49/maand recht op 250000 requests. Ook biedt deze API dezelfde features aan als de vorige twee, namelijk maps opgebouwd uit tegels, vectorkaarten, markers, zoeken naar locaties en routebegeleiding.

3.1.4 TomTom

De vierde alternatieve API is TomTom en deze heeft een zeer goede reputatie als het aankomt op navigatie. Er worden 2500 gratis requests per dag aangeboden, wat neerkomt op zo'n 75000 gratis requests/maand. Anderzijds bieden ze ook meer functionaliteiten aan. Zo is er naast kaartweergaven en locaties waar men naar kan zoeken ook de mogelijkheid om op de hoogte te blijven van de verkeersdichtheid. Ook kan er tijdens de routebegeleiding worden geopteerd om te zoeken naar de beste route, wat dus niet altijd de kortste route is. Naast deze features zijn het dan ook weer de standaardfeatures die worden aangeboden, net zoals bij de vorige alternatieven.

3.1.5 Mapbox

De vijfde en laatste interessante mogelijkheid is de Mapbox API. Deze wordt door zowel Facebook als Snapchat gebruikt bij het tekenen van data van zowel open als bedrijfseigen bronnen. Naast de features die de vorige alternatieven aanbieden, staat Mapbox vooral bekend om zijn aangepaste kaarten. Het feit dat er slechts zeer weinige concurrenten deze feature kunnen aanbieden, maakt Mapbox dan ook zeer populair. Daarnaast bieden ze de gebruikers 50000 gratis requests/maand aan en wordt er daarna per 1000 requests $0.50 aangerekend.

Mapbox' platform blijft maar groeien, met als gevolg dat er ook steeds meer features worden geïmplementeerd. Doordat men zeer degelijke geografische informatie kan aanbieden doormiddel van OSM, waar ontwerpers zich meer kunnen focussen op de userinterface, kunnen de ontwikkelaars zich op hun beurt dan weer meer bezig houden met programmeren. Waar ontwikkelaars en ontwerpers van Google beperkt zijn tot de ontwikkeling en de bouw in het Google Framework, wordt er bij Mapbox de mogelijkheid geboden om zoveel mogelijk features toe te voegen naargelang de behoeften van de gebruikers. Aangezien Mapbox *opensource* is met een community die verbeteringen ondersteunt zijn ze, net zoals Google, zeer afhankelijk van de feedback van hun gebruikers. Een ander groot voordeel voor Mapbox is dat ze "een symbiotische relatie hebben met OSM" [4]. Hierdoor kunnen er binnen de 48 uren aanpassingen worden gemaakt aan de onderliggende OSM data.

3.1.6 Overzicht alternatieven

Kort samengevat kan er dus worden besloten dat er zeker enkele degelijke alternatieven zijn voor de Google Maps API. Als er wordt gekeken naar de prijs, is OpenLayers de overduidelijke winnaar aangezien het compleet gratis is. Daarna volgt Here met 250000 requests/maand, TomTom met 75000 requests/ maand en Mapfit en Mapbox met beide 50000 requests/maand. Vervolgens heeft elke API een functionaliteit waarin hij uitblinkt en kan er afhankelijk van de focus van het project een keuze worden gemaakt. Indien de navigatie belangrijk is, wordt er best geopteerd voor de TomTom API. Wanneer de accuraatheid belangrijk is, wordt er best gekozen voor de Mapfit API. OpenLayers is dan weer de eerste keus als het draait om betaalbaarheid. Voor de beste visualisatie van de kaarten primeert Here en voor aangepaste kaarten valt de keuze altijd op Mapbox. Dit wordt duidelijk weergegeven in bijlage C.

Voor het huidige project maken het aantal requests/maand niet uit waardoor er hier dus geen rekening mee gehouden moet worden tijdens de keuze van een geschikt alternatief. Later in het project moeten gebruikers de mogelijkheid hebben om te kunnen navigeren, maar aangezien dit niet de focus is, is het dus ook niet belangrijk genoeg om voor de TomTom API te kiezen. OpenLayers is gratis, maar aangezien elke alternatieve API voor dit project voldoende requests aanbiedt moet hier geen rekening mee gehouden worden. Jasper Heeren, ook stagiair bij GPS nv, koos voor de Here API omdat deze het beste is in het visualiseren van de kaarten. Bijgevolg blijven er dan nog twee alternatieven over, namelijk de MapBox API en de Mapfit API. Een zeer goede accuraatheid is steeds een pluspunt, maar bij het plaatsen van de markers van de collega's is de accuraatheid van iedere marker bekend. Het voordeel van Mapfit is in deze situatie dus niet echt nuttig en daarom is Mapbox dan ook de enige logische keuze om te gebruiken als alternatieve API in dit project.

Onderzoek wijst uit dat het perfect mogelijk is om deze API te integreren in het project. In de huidige versie kan er gewerkt worden met onder andere verschillende kaartstijlen, data clusters, camera manipulatie, opvragen van de kaart en nog veel meer. Alvorens de Maps SDK gebruikt kan worden, moet de SDK als *dependency* worden toegevoegd aan het project. Waarbij een *dependency* de afhankelijkheid is van de functionaliteit die wordt aangeboden door een externe component. Hoe deze configuratie precies zal verlopen, wordt verder uitgelegd in de uitvoeringsfase.

4 Uitvoering

4.1 Opzetten Mapbox

Na het implementeren van de belangrijkste features gebruikmakende van de Google Maps API, wordt er getracht deze features te dupliceren in een ander scherm. Het enige verschil was dat de gedupliceerde features worden geïmplementeerd gebruikmakende van de alternatieve API, namelijk de Mapbox API.

Alvorens de features van het eerste scherm nagemaakt kunnen worden, is het natuurlijk wel belangrijk dat er een kaart getoond kan worden met de Mapbox API. Dit bleek echter helemaal niet zo moeilijk te zijn.

```
dependencies {
  implementation 'com.mapbox.mapboxsdk:mapbox-android-sdk:7.3.2'
}
```

Figuur 23: Dependancy

```
<string name="mapbox_access_token">MAPBOX_ACCESS_TOKEN</string>
```

Figuur 24: Access Token

```
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)
{
    Mapbox.getInstance(Objects.requireNonNull(getContext()),
            accessToken: "pk.eyJ1IjoieW9yYW5uZWxpc3NlbiIsImEiOiJjanN5ZTJleGQwYmwwNGFxdDl0cWdyc2NtIn0._k9MremisSACSPr5bs1btg");

    return inflater.inflate(R.layout.fragment_map_box, container,  attachToRoot: false);
}
```

Figuur 25: onCreateView Mapbox

Allereerst werd de juiste *mapbox-android-sdk* toegevoegd bij de *dependancies*. Wanneer dit was gebeurd en er opnieuw gesynchroniseerd was, moest er een unieke access token gegenereerd worden. Deze token is belangrijk voor het aanmaken van de view de gebruiker te zien zal krijgen.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

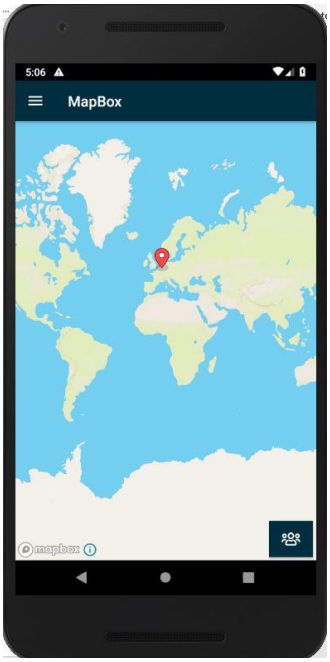
Figuur 26: Permissies

Vervolgens is het belangrijk dat de juiste permissies zijn ingesteld en zodat de gebruiker weet dat de applicatie zijn toestemming nodig heeft om de locatie op te kunnen vragen.

```
private MapView mapView;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

  Mapbox.getInstance(this,
  pk.eyJ1IjoieW9yYW5uZWxpc3NlbiIsImEiOiJjanN5ZTJleGQwYmwwNGFxdD10cWdyc2NtIn0._k9MremisSACSPr5bs1btg);

  setContentView(R.layout.activity_main);

  mapView = (MapView) findViewById(R.id.mapView);
  mapView.onCreate(savedInstanceState);
  mapView.getMapAsync(new OnMapReadyCallback() {
  @Override
  public void onMapReady(@NonNull MapboxMap mapboxMap) {

    mapboxMap.setStyle(Style.MAPBOX_STREETS, new Style.OnStyleLoaded() {
    @Override
    public void onStyleLoaded(@NonNull Style style) {

        // Map is set up and the style has loaded. Now you can add data or make other map adjustments



    }
  });
  }
});
```

Figuur 27: Toevoegen kaart

Op figuur 27 is de code te zien die ervoor zorgt dat de gebruiker effectief een kaart op zijn scherm zal zien. Met enkele regels code die achteraf nog werden toegevoegd, werd er op de kaart ook een marker geplaatst op de huidige positie van de gebruiker, zoals te zien is op figuur 28.



Figuur 28: Mapbox kaart

4.2 Implementeren features

De eerste feature die zowel in Google Maps als in Mapbox is geïmplementeerd, zijn de cirkels die aanduiden hoe accuraat de markers de locatie van de collega's weergeven.

```
/**
 * Setup the accuracy circles for the markers.
 *
 * @param originalClocking original clocking
 * @return circle options
 */
private CircleOptions setupCircles(OriginalClocking originalClocking)
{
    CircleOptions circleOptions = new CircleOptions();
    circleOptions.center(new LatLng(originalClocking.getGeolocation().getCoordinates()
            .getLatitude(), originalClocking.getGeolocation().getCoordinates().getLongitude()));
    circleOptions.radius(originalClocking.getGeolocation().getCoordinates().getAccuracy());
    circleOptions.fillColor(ResourcesCompat.getColor(getResources(), R.color.accentToFillCircle,  theme: null));
    circleOptions.strokeColor(ResourcesCompat.getColor(getResources(), R.color.accentToFillCircle,  theme: null))

    return circleOptions;
}
```

Figuur 29: Instellen cirkelopties bij Google Maps

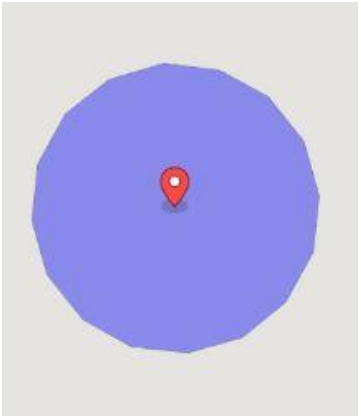
```
/**
 * Generate the polygonOptions.
 *
 * @param centerCoords centerCoords
 * @param radiusInKm radiusInKm
 * @return polygonOptions
 */
private PolygonOptions generatePerimeter(LatLng centerCoords, double radiusInKm)
{
    List<LatLng> positions = new ArrayList<>();
    double distanceX = radiusInKm / (111.319 * Math.cos(centerCoords.getLatitude() * Math.PI / 180));
    double distanceY = radiusInKm / 110.574;

    double slice = (2 * Math.PI) / nrOfSides;

    double theta;
    double x;
    double y;
    LatLng position;
    for (int i = 0; i < nrOfSides; i++)
    {
        theta = i * slice;
        x = distanceX * Math.cos(theta);
        y = distanceY * Math.sin(theta);

        position = new LatLng( latitude: centerCoords.getLatitude() + y,  longitude: centerCoords.getLongitude() + x);
        positions.add(position);
    }
    return new PolygonOptions().addAll(positions).fillColor(Color.BLUE).alpha(0.4f);
}
```

Figuur 30: Genereren van polygonen bij Mapbox

Zoals te zien op bovenstaande figuren is er een duidelijk verschil in de code die nodig is om deze features te implementeren. Bij Google Maps is het zeer eenvoudig, er kunnen namelijk cirkels worden toegevoegd aan de kaart. Het enige wat er hier nog moet gebeuren is het instellen van de opties met behulp van de data die we binnenkrijgen. De coördinaten van de collega worden ingesteld als het middelpunt en de accuraatheid geeft de radius van de cirkel aan. Tot slot wordt de achtergrondkleur aangepast zodat de cirkels duidelijk te zien zijn op de kaart.

Bij Mapbox is dit helemaal niet zo simpel. De mogelijkheid om cirkels toe te voegen aan de kaart is er namelijk niet. Er worden dus polygonen gegenereerd en aan de kaart toegevoegd. Aan de hand van het middelpunt en de radius die worden meegegeven in de code, worden er van 500 punten de coördinaten berekend. Deze worden bijgehouden in een lijst die op zijn beurt wordt meegegeven bij het genereren van een polygoon. In de onderstaande figuren wordt het verschil dan ook zeer duidelijk. Aangezien er bij Mapbox zelf een cirkel zal moeten worden gegenereerd, zal deze er dan ook niet perfect rond uit zien.

Figuur 31: Google Maps cirkel



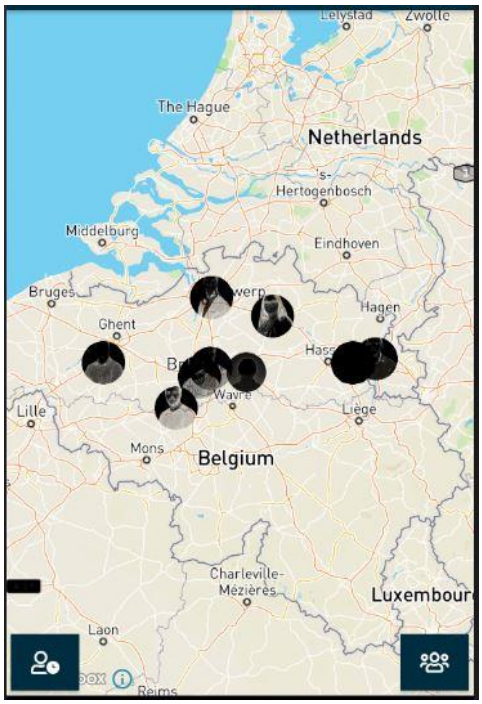
Figuur 32: Mapbox polygoon

Het filteren op oudere registraties is een volgende feature die werd toegevoegd. Bij het opstarten zal deze filter reeds worden toegepast en ziet de gebruiker enkel de laatste registraties van alle collega's. Als de gebruiker toch graag alle registraties te zien krijgt, inclusief de oudere registraties, kan hij/zij op de button in de linkerbenedenhoek drukken.

Figuur 33: Google maps met filter op oudere registraties

Figuur 34: Google Maps zonder filter op oudere registraties

Figuur 35: Mapbox met filter op oudere registraties



Figuur 36: Mapbox zonder filter op oudere registraties

Ook hier zijn de verschillen weer duidelijk zichtbaar. Bij het indrukken van de filter worden de profielfoto's plots zwart en wit. Dit is geen probleem dat kan worden opgelost in de code. Na uitvoerig testen is gebleken dat dit soms perfect werkt en soms problemen geeft. De reden waarom dit gebeurt blijft echter onduidelijk.

```
/**
 * Show the registrations.
 *
 * @param filterList filterList
 * @param filter filter
 * @param show show
 */
private void showRegistrations(List<ClockingMarkerDetails> filterList, String filter, boolean show)
{
    if (filter != null)
    {
        for (ClockingMarkerDetails detail : filterList)
        {
            if (detail.getOriginalClocking().getProcessTypeCD().equals(filter))
            {
                detail.getMarker().setVisible(show);
                detail.getCircle().setVisible(show);
            }
        }
    }
    else
    {
        for (ClockingMarkerDetails detail : filterList)
        {
            detail.getMarker().setVisible(show);
            detail.getCircle().setVisible(show);
        }
    }
}
```

Figuur 37: Google Maps tonen van markers

```
/**
 * Show markers.
 *
 * @param detailsList detailsList
 */
private void showMarkers(List<ClockingMarkerDetailsMb> detailsList)
{
    mapboxMap.clear();
    for (ClockingMarkerDetailsMb detailsMb : detailsList)
    {
        mapboxMap.addMarker(detailsMb.getMarkerOptions());
        mapboxMap.addPolygon(detailsMb.getPolygonOptions());
    }
}

/**
```

Figuur 38: Mapbox tonen van markers

Het verschil is niet alleen zichtbaar voor de gebruiker, de achterliggende code is ook anders. Zo is het Bij Google Maps perfect mogelijk om markers onzichtbaar te maken. Op gebied van performantie is dit een zeer groot voordeel. De markers worden bij het inladen van de kaart allemaal getekend en worden daarna zichtbaar of onzichtbaar gemaakt afhankelijk van de filters die van toepassing zijn. Bij Mapbox is dit niet zo, hier kunnen de markers niet onzichtbaar worden gemaakt. Ze moeten dus elke keer opnieuw allemaal getekend worden wanneer de gebruiker de filterknop indrukt.

Zoals op enkele voorgaande figuren te zien is, zijn er sommige stukken code doorstreept. Dit betekent dat de gebruikte features *deprecated* zijn. Dit wilt zeggen dat de features niet meer op deze manier gebruikt worden of dat er een betere, sterkere feature in de plaats is gekomen. Ze zullen niet meteen verdwijnen, maar op deze manier wordt aangegeven dat de ontwikkelaar best op zoek gaat naar een andere oplossing. Dit zou mogelijk het probleem kunnen zijn bij de profielfoto's die van kleur veranderen. Het grote nadeel van Mapbox is echter dat er zeer weinig degelijke documentatie is en dat er voor veel *deprecated* features nog geen oplossing of alternatief is.

```
@Override
public boolean onMarkerClick(@NonNull Marker marker)
{
    CameraUpdate cu = CameraUpdateFactory.newLatLng(marker.getPosition());

    mapboxMap.animateCamera(cu);

    return false;
}
```

Figuur 39: Mapbox camera verschuiven naar marker

Een laatste klein verschil tussen beiden wordt duidelijk bij het drukken op de markers. Bij Google Maps verschuift de camera automatisch zodat de marker in het midden van het scherm wordt weergegeven. Bij Mapbox is dit niet en is er een extra stuk code nodig dat hiervoor zal zorgen.

5 Literatuurstudie

5.1 Jungleworks

Volgens Jungleworks heeft Google uitstekend werk geleverd inzake cartografie. In 2009 kondigde ze de eerste navigatie aan voor Android, waarna ze konden beginnen concurreren met andere bedrijven. Jungleworks vergelijkt de API's van Google en Mapbox en gaat ze op vier gebieden met mekaar vergelijken.

Een van de grote verschillen is zonder twijfel het verschil in prijs. Bij Google Maps krijgt de gebruiker 2500 gratis requests/maand en moeten ze $0.50 betalen per 1000 extra requests, met een maximum van 100000 requests/dag. Indien dit nog niet voldoende is moet Google Maps gecontacteerd worden en dan wordt er gekeken als de gebruiker eventueel in aanmerking komt voor een licentie. Mapbox maakt het makkelijker en geeft elke gebruiker 50000 requests/maand gratis.

Het volgende voordeel dat Mapbox heeft op zijn concurrent is het aanpassen van de kaarten naargelang de behoeften van de gebruiker. De API maakt gebruik van een *map editor* genaamd TileMill en is zeer eenvoudig om mee te werken voor de gebruikers. Voor $50/maand krijgt men toegang tot een bibliotheek met daarin vooraf gedefinieerde kaarten en de mogelijkheid om drie eigen kaartstijlen toe te voegen.

De accuraatheid van de kaart is dan weer een enorm grote troef van de Google Maps API. Ze hebben dan ook toegang tot enorme datastromen die geen enkel ander bedrijf zich zou kunnen veroorloven. Om tijdens het navigeren de gebruikers een ETA te kunnen geven, zijn er talloze variabelen die gecontroleerd moeten worden. Informatie over het terrein, gemiddelde snelheid, data i.v.m. files, weer, etc., zijn allemaal aspecten die belangrijk zijn bij een schatting van het tijdstip van aankomen. Uiteraard is de Google Maps API dan ook veel accurater dan Mapbox. OSM heeft anderzijds ook geen robuuste database voor ontwikkelingslanden, hierdoor is informatie op bepaalde zoomniveaus niet zo accuraat.

Tot slot heeft Google ook nog enkele bijhorende diensten die ze samen met de Google Maps API kunnen aanbieden, zoals Places, Business, Streetview, satellietbeelden, etc. In tegenstelling tot het aanbieden van bovenstaande diensten, focust Mapbox zich standaard op een open platform en aanpasbare kaarten.

Naast enkele verschillen zijn er natuurlijk ook heel wat features die door beide API's worden aangeboden. Een voorbeeld hiervan is ETA, die dan zoals hierboven vermeld wel accurater zal werken bij de Google Maps API. Real Time Tracking, waarbij men op elk moment de exacte locatie van een persoon of object kan bepalen, en Turn by Turn navigation, waarbij de gebruiker constant updates krijgt van de richting die hij uit moet, zijn ook features die door meerdere API's worden ondersteund. [5]

5.2 Masuga

Ook Catherine Kleimeier maakte de overstap van de Google Maps API naar Mapbox. Via haar blog op Masuga gaf ze hier meer uitleg over en haalde ze enkele redenen aan waarom ze de overstap maakte.

Ook zij is van mening dat de prijswijzigingen bij Google wel eens voor wat onrust bij verschillende sites, die gebruik maken van Google Maps, kunnen zorgen. Kleinere sites, die niet zoveel kaartweergaven genereren, zullen weinig tot geen problemen ondervinden. Maar er zijn echter ook veel sites die wel intensief gebruik maken van de services van Google, zij zouden dus wel eens enorme kostenstijging kunnen verwachten. Hierdoor moeten sommige bedrijven zeer moeilijke beslissingen nemen. Veranderen ze de manier waarop ze gebruikmaken van de kaarten of gaan ze opzoek naar een alternatieve API?

Hiervoor was Mapbox de ideale oplossing. Ze bieden kaarten aan die de gebruiker kan aanpassen naar zijn behoeften en zijn anderzijds dan ook nog eens zeer gedetailleerd en interactief. Ook Mapbox studio ziet Catherine als een groot voordeel ten opzichte van de Google Maps API. Dit is een applicatie waarmee kaarten volledig kunnen worden aangepast en waarbij men herbruikbare gegevenssets kan maken.

Verder zijn de vergelijkbare gegevensbronnen en functionaliteiten een enorm pluspunt en kan er zonder al te veel problemen worden overgeschakeld naar Mapbox. [6]

5.3 Novapex

Novapex maakt ook een vergelijking tussen de meest gebruikte Google Maps API en de nieuwe, maar veelbelovende Mapbox API. Ze vergelijken beide aan de hand van hun voordelen.

Zo biedt Google Maps, een enorm bekend web mapping-service van Google, ontwikkelaars de mogelijkheid om verschillende types kaarten en kaartafbeeldingen in hun applicaties te integreren. Naast de Google Maps API bieden ze dan ook nog eens enkele andere API's aan die gebaseerd zijn op de data van Google Maps, namelijk Google Maps Image API, Google Places API, Google Earth API en de Google API for businesses.

Een zeer groot voordeel van Google Maps is zijn integratie in Google Search, zo helpt de informatie van Google Maps in het verbeteren van de ranglijst van websites. Ze bieden dan ook steeds meer features en diensten aan met betrekking tot Google Maps.

Maar net zoals bij Jungleworks en Catherine, merken ze hier dat de prijswijziging van Google ervoor zorgt dat er steeds meer wordt gezocht naar een alternatief.

Mapbox werd opgericht in 2010 en hier werd vooral de nadruk gelegd op open data en opensource technologieën. Men is bij Mapbox dan ook zeer gefocust op het weergeven van een zo mooi mogelijke kaart. Ze kwamen met het idee om een zo groot mogelijke verzameling van afbeeldingen van een bepaald gebied te verzamelen. Vervolgens gaan ze deze allemaal vergelijken en de minst troebele afbeelding gebruiken om hun kaart mee op te bouwen.

Verder bieden ze enorm krachtige API aan die dan ook nog eens een eenvoudige map editing tool aanbiedt.

Sinds de prijswijziging bij Google ziet Mapbox een grote stijging in het gebruik van hun diensten. En zijn er enkele zeer bekende bedrijven die ook de overstap gemaakt hebben. [7]

5.4 Conclusie literatuurstudie

Het is enorm opvallend wat een simpele prijswijziging bij een bedrijf allemaal teweeg kan brengen. Zowel Jungleworks, Catherine als Novapex spreken over de prijsstijging van Google Maps en dat er daardoor veel meer beroep wordt gedaan op het gebruik van alternatieven.

Google Maps is en blijft een enorm populair en kan zonder twijfel nog zeer lange tijd concurreren met al zijn alternatieven. Ze kunnen beroep doen op een datastroom die andere bedrijven zich niet kunnen veroorloven en zullen hierdoor dus altijd een stapje voor zijn.

Mapbox doet het, ondanks het voordeel dat Google Maps op hun heeft, bijlange niet slecht en zou zelfs op termijn wel eens de grootste concurrent kunnen worden. Ze bieden uitstekende diensten en functionaliteiten aan die dan ook nog eens betaalbaar zijn.

6 Conclusie

Na onderzoek kan er worden besloten dat er zeker enkele interessante alternatieven zijn om Google Maps te vervangen. Ze zijn en blijven waarschijnlijk nog een hele tijd het meest populair, maar door de prijswijzigingen die ze doorvoerde zullen de alternatieve API's ook meer gebruikt worden.

Mapbox is hier een zeer mooi voorbeeld van. De diensten die zij aanbieden zijn minstens even goed als die van Google Maps en zijn op hun beurt ook nog een goedkoper. Ze kunnen niet gebruikmaken van dezelfde datastroom als Google, maar dit proberen ze te compenseren met de kwaliteit van hun kaarten. Zo proberen ze er steeds voor te zorgen dat ze hun gebruikers de best mogelijke afbeeldingen kunnen laten zien. Ook zijn hun kaarten zeer eenvoudig om aan te passen, de ontwikkelaar kan dit doen naargelang de behoeften van de klant.

Daarnaast gelijken de functionaliteiten en diensten van Mapbox zeer sterk op die van Google Maps waardoor de overgang niet zo moeilijk is om te maken. Veel meer grote en bekende bedrijven kiezen tegenwoordig ook voor de overstap, het ziet er dus naar uit dat Google een waardige concurrent gevonden heeft.

7 Bibliografie

[1] T. Bush, „5 Powerful Alternatives to Google Maps API,” 1 November 2018. [Online]. Available: https://nordicapis.com/5-powerful-alternatives-to-google-maps-api/?fbclid=IwAR3KloMaRr9bJC4aGHETeXvleOWB41APZqvGALGcm2G-A-rvrKgaAW2yzVo. [Geopend 5 April 2019].

[2] P. Sawers, „Here launches new freemium plan for developers following Google Maps API changes,” VentureBeat, 3 Augustus 2018. [Online]. Available: https://venturebeat.com/2018/08/03/here-launches-new-freemium-plan-for-developers-in-response-to-google-maps-api-changes/. [Geopend 5 April 2019].

[3] F. Online, „OpenLayers,” Finances Online, [Online]. Available: https://reviews.financesonline.com/p/openlayers/. [Geopend 19 April 2019].

[4] S. Slater, „Mapbox and Google: Mapping Out Pros and Cons,” Concept3D, 21 November 2017. [Online]. Available: https://blog.concept3d.com/mapbox-and-google-mapping-out-pros-and-cons. [Geopend 19 April 2019].

[5] Jungleworks, „google-vs-mapbox,” Jungleworks, [Online]. Available: https://jungleworks.com/google-vs-mapbox/. [Geopend 14 Mei 2019].

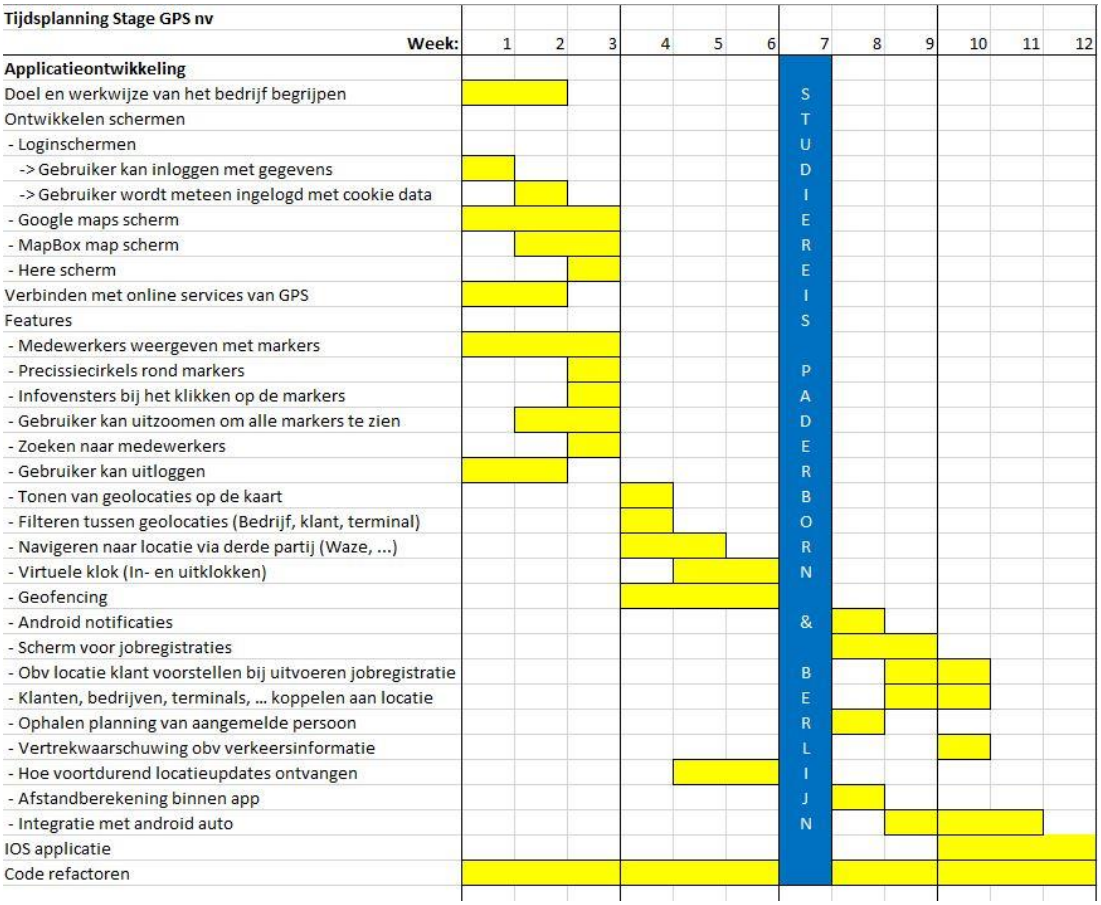
[6] C. Kleimeier, „mapbox-google-maps-alternative,” Masuga, 22 Augustus 2018. [Online]. Available: https://gomasuga.com/blog/mapbox-google-maps-alternative. [Geopend 14 Mei 2019].

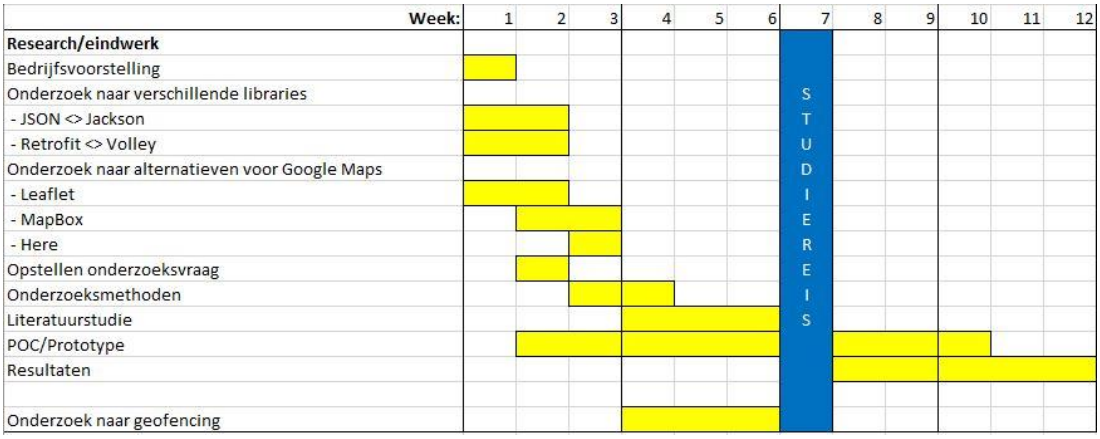
[7] Admin, „google-maps-vs-mapbox,” Novapex, 28 Decemeber 2018. [Online]. Available: http://www.novapextech.com/v2/2018/12/28/google-maps-vs-mapbox/. [Geopend 14 Mei 2019].

Bijlagen

Bijlage A: Tijdsplanning

Deze bijlage bevat de tijdsplanning die werd opgesteld in Excel. Features en taken werden in de eerste kolom toegevoegd en in de eerste rij wordt het aantal weken weergegeven. Voor elke taak werden er een aantal vakjes ingekleurd die weergaven hoeveel tijd er werd voorzien om hieraan te werken. Er is een tijdsplanning van zowel het ontwikkelgedeelte, als het researchgedeelte.





Bijlage B: JSON request en response payload

Deze bijlage bevat een voorbeeld van de payloads die het bedrijf voorzag om de nodige API-calls te kunnen doen.

HTTP **POST** naar *https://emprovaportal.gps-time.be/emprova-portal-service/api/customers/serviceurl*

Request body:

{"CustomerNR": "999998", "Pincode": "9955"}

Response body:

{"serviceUrl": "https://..."}

Opmerking: drie custom request headers instellen voor **alle** requests:

- "GPS-Platform": "android"
- "GPS-Application": "emprova-android"
- "GPS-DeviceType": "phone" of "tablet"

Opmerking: eventueel een '/' toevoegen aan de *service URL* uit de response, altijd 'gps-services/api/' toevoegen.

Bijlage C: Overzicht alternatieven

Deze tabel geeft een mooi overzicht van de gratis requests van elk alternatief, alsook hun beste eigenschap.

Overzicht	**Here**	**OpenLayers**	**Mapfit**	**TomTom**	**Mapbox**
Requests	250000/maand	gratis	50000/maand	75000/maand	50000/maand
Beste eigenschap	Visualisatie	Betaalbaarheid	Accuraatheid	Navigatie	Aangepaste kaarten