

Tracker

Automatisch werkuren tracken met locatie

door Elias Meire, Haroen Viaene en Arnaud Weyts  
Bachelor in de Elektronica-ICT, afstudeerrichting ICT 2015-2016  
Mentor: Bram Gosseye

# Inhoudsopgave

[Inleiding 2](#_Toc451547277)

[1 Mogelijke oplossingen en gekozen oplossing 3](#_Toc451547278)

[1.1 Frontend 3](#_Toc451547279)

[1.1.1 Native applicatie 3](#_Toc451547280)

[1.1.2 Hybride applicatie 4](#_Toc451547281)

[1.1.3 Responsive website 4](#_Toc451547282)

[1.1.4 Gekozen frontend 5](#_Toc451547283)

[1.2 Backend 5](#_Toc451547284)

[1.2.1 Eigen oplossing 5](#_Toc451547285)

[1.2.2 MBaaS 5](#_Toc451547286)

[1.2.3 Gekozen backend 6](#_Toc451547287)

[2 Technische detailuitwerking van gekozen oplossing (met controle en verificatie) 7](#_Toc451547288)

[2.1 Webinterface 7](#_Toc451547289)

[2.2 Androidapplicatie 8](#_Toc451547290)

[2.2.1 Design en lay-out 9](#_Toc451547291)

[2.2.2 Programmalogica 11](#_Toc451547292)

[2.3 Backend 13](#_Toc451547293)

[2.4 Versiebeheer 14](#_Toc451547294)

[2 Risicoanalyse 15](#_Toc451547295)

[3.1 Beveiliging van gebruikersgegevens 15](#_Toc451547296)

[3.2 Fouten in de locatiebepaling 16](#_Toc451547297)

[2.3 Manuele tijdsinstelling 16](#_Toc451547298)

[3.4 Andere 16](#_Toc451547299)

[3 Kostenraming en levensduur 17](#_Toc451547300)

[4.1 Kostenraming 17](#_Toc451547301)

[4.2 Levensduur 18](#_Toc451547302)

[Conclusie 19](#_Toc451547303)

[Literatuurlijst 20](#_Toc451547304)

[Bijlage 1: Uiteindelijke werkplanning 23](#_Toc451547305)

# Inleiding

Momenteel is het moeilijk om als werkgever bij te houden hoeveel uur je werknemers op verplaatsing werken. Vooral bij kleine zelfstandigen worden de werkuren veelal nog handmatig ingebracht door de werknemers. Dat legt de verantwoordelijkheid voor de juistheid van deze gegevens bij de werknemers, waardoor deze foutief kunnen zijn. Voor de werknemers kan het ook storend zijn om steeds te moeten onthouden waar ze geweest zijn en hoe lang dat duurde.

Een oplossing hiervoor is een applicatie die alle werkuren en verplaatsingen van de medewerkers bijhoudt op een centrale plaats. Zo kan de werkgever deze op een overzichtelijke manier beheren en controleren. Tevens zou de werkgever hieruit gemakkelijk de loonkosten kunnen berekenen. Deze applicatie kan men onderverdelen in 3 deelaspecten: een interface voor de werkgever, een interface voor de werknemers en een backend waarin de gegevens van de werknemers worden opgeslagen.

Het doel is om als werknemer de app te kunnen gebruiken om de werkuren in te vullen. Dat kan ook geautomatiseerd worden met behulp van een locatieservice. De werkgever kan de gewerkte uren van zijn werknemers bekijken via de webinterface. Dat kan later uitgebreid worden zodat ook de werkgever toegang heeft tot een webinterface en omgekeerd ook de werkgever. Omwille van de tijdsbeperking van twaalf weken voor dit project wordt er beslist een app voor Android bruikbaar voor werknemers, en een webinterface bruikbaar voor werkgevers te maken.

Een belangrijk vraagstuk in elke applicatie die gebruikersdata verzamelt, is beveiliging. Er wordt met aandacht nagedacht over implicaties van beslissingen in verband met privacy en op elk moment wordt gecontroleerd of het mogelijk is bepaalde data te lezen zonder correct ingelogd te zijn. Indien dat het geval is, wordt de beveiliging verbeterd.

Bij het maken van de applicatie wordt er gebruik gemaakt van verschillende internetbronnen, documentatie van de gebruikte technologieën, raad van professionals enzovoort. Er is tevens besloten om de app “open source” te ontwikkelen, dit wil zeggen dat alle code zichtbaar zal zijn voor iedereen die er iets uit wil leren.

Dit rapport is op de volgende manier gestructureerd. Allereerst worden de mogelijke oplossingen in detail bestudeerd. Daaropvolgend wordt de gekozen oplossing in technisch detail uitgelegd. Voor de gekozen oplossing wordt vervolgens een gedetailleerde risicoanalyse opgesteld. Ten slotte wordt er een kostenraming gemaakt voor de realisatie en een bespreking opgesteld over de toekomst van het project.

# Mogelijke oplossingen en gekozen oplossing

## Frontend

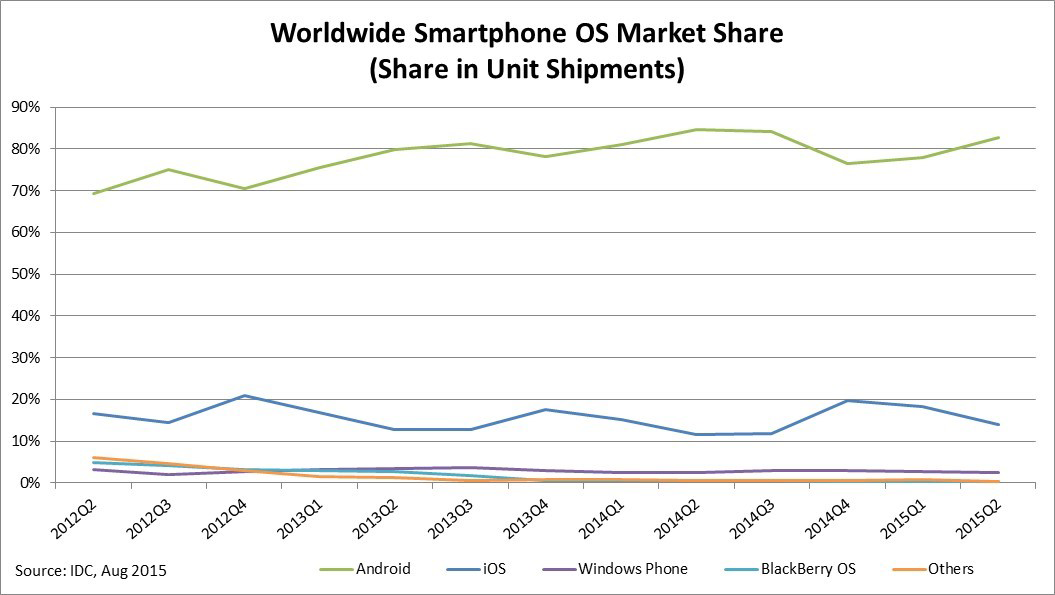
Het maken van een frontendapplicatie kan op veel verschillende methoden. De belangrijkste keuze die gemaakt moet worden is die tussen hybride en native applicatieontwikkeling.

### Native applicatie

Het grootste voordeel voor native ontwikkeling is performantie want in tegenstelling tot hybride applicaties zijn er geen tussenlagen zoals webviews of interpreters. De code van de applicatie kan dus direct gecompileerd en uitgevoerd worden op het apparaat. Bovendien zijn alle API’s van het gekozen platform beschikbaar en goed gedocumenteerd. Bijgevolg heeft een ontwikkelaar veel betere controle over de werking van de applicatie. Als laatste voordeel heeft elk platform zijn eigen IDE die krachtige tools bevat om efficiënt te kunnen ontwikkelen zoals debuggers en optimalisatietools. Dat is tevens de manier die Android, iOS en andere mobiele besturingssystemen aanraden om apps te ontwikkelen [1].

Een native applicatie brengt echter wel enkele nadelen met zich mee, zo zal de code van de applicatie volledig specifiek zijn aan het platform. In het geval dat er een native app voor meerdere platformen zou moeten ontwikkeld worden, zouden deze geen code gemeenschappelijk hebben [2]. Er zouden dus twee aparte applicaties moeten geschreven worden.

Het grootste nadeel van native apps is dat ze platformspecifiek zijn en dus zou er in het kader van dit project een platform moeten worden gekozen. Bijgevolg wordt er een aanzienlijk aantal potentiële gebruikers verloren. In figuur 1 zijn de wereldwijde marktaandelen van smartphones per besturingssysteem te zien. Hieruit blijkt dat Android het grootste platform is, en dus de beste kandidaat lijkt. Als bijkomend voordeel wordt er op Android geprogrammeerd in Java, een programmeertaal waar veel programmeurs mee vertrouwd zijn. Java wordt bovendien in het eerste jaar van de opleiding bachelor in de elektronica-ICT behandeld [3].



Figuur 1: Marktaandeel smartphone besturingssystemen [4]

### Hybride applicatie

Een alternatief voor een native app is een hybride app, waarin alle code geschreven wordt in één taal of framework. Hierbij wordt de app meestal volledig in een webview gemaakt. Dat wil zeggen dat alles in HTML, CSS en JavaScript geschreven is. De populairste frameworks die deze strategie gebruiken zijn Ionic, Phonegap en Cordova [5]. Het grote voordeel aan deze werkwijze is uiteraard dat er maar één codebase nodig is om apps te maken voor alle platformen. Hierdoor kan er efficiënt in team gewerkt worden en snel een applicatie voor meerdere platformen worden gerealiseerd.

Het grootste nadeel bij een hybride app is dat een hybride app steeds weinig controle zal hebben over hoe de code uiteindelijk zal uitgevoerd worden op de hardware. Dat heeft ook als gevolg dat een hybride app nooit dezelfde performantie zal behalen als een native app [6].

Een nieuwkomer in deze categorie is React Native. Deze maakt gebruik van React als framework en gebruikt JavaScript als programmeertaal voor elk aspect van de app. Het grote voordeel bij React Native is dat er slechts één codebase nodig is voor beide platformen, maar dat er wel gebruik kan gemaakt worden van native componenten van de populairste platformen. Het nadeel hiervan is dat er een grote expertise van ontwikkeling in React vereist is. Bovendien staat React Native nog in zijn kinderschoenen [7].

### Responsive website

Een website heeft als voordeel dat het volledig platformonafhankelijk is. Elk besturingssysteem met een browser kan een website weergeven. Door gebruik te maken van responsive design kan men een website gebruiksvriendelijk houden op elke schermgrootte. Een responsive website heeft als groot nadeel dat men geen API’s kan aanspreken van het platform, de website is namelijk gesandboxt in de browser. Zeker het gebruik van API’s zoals locatie terwijl de website niet specifiek open is, is zeer moeilijk [8].

### Gekozen frontend

Voor de werknemer zal er als frontend geopteerd worden voor een mobiele app. Meer specifiek wordt er gekozen voor een native Android app. Deze optie is gekozen omdat het voor werknemers belangrijk is om toegang te hebben tot de Android API’s om efficiënt om te gaan met bijvoorbeeld het bijhouden van locaties. Er wordt gekozen voor Android als platform omwille van het marktaandeel en het gebruik van Java.

Voor werkgevers wordt er gekozen voor een responsive website. Bij de werkgever moeten vooral gegevens uit de backend getoond worden. Hier is het niet belangrijk om toegang te hebben tot alle API’s. Door te kiezen voor een responsive website wordt maximale compatibiliteit behouden met zowel desktopcomputers als mobiele apparaten.

## Backend

Bij het opzetten van een backend zijn er een aantal opties beschikbaar. Enerzijds is het een mogelijkheid om zelf een backend te hosten, waarbinnen er ook nog gekozen kan worden tussen verschillende types databanken zoals MySQL, MongoDB, MariaDB ... en verschillende programmeertalen om deze in te schrijven. Anderzijds kan er ook gebruik gemaakt worden van een MBaaS [9].

### Eigen oplossing

Een eigen backend ontwikkelen brengt een aantal voordelen met zich mee. Een ontwikkelaar heeft namelijk volledige controle over de backend. Als gevolg hiervan kan de backend veel specifieker gemaakt worden en dus sneller zijn. Naarmate de applicatie op grotere schaal gebruikt wordt, zal deze optie ook goedkoper zijn dan een MBaaS. Een groot nadeel van deze oplossing is dat er veel tijd nodig is om een goede backend te ontwikkelen. De ontwikkelaar is tevens zelf verantwoordelijk voor aspecten zoals beveiliging en hosting.

### MBaaS

Bij een MBaaS wordt er gebruik gemaakt van een soort geprefabriceerde backend. Alle hosting gebeurt op een platform, zoals Firebase. Een voordeel hiervan is dat er als ontwikkelaar niet moet worden nagedacht over aspecten zoals beveiliging en hosting.

Veel verschillende types MBaaS zorgen voor gemakkelijke integratie op populaire platformen zoals Android, iOS, web … waardoor er minder inspanning nodig is om verschillende clients te verbinden met de data.

De beste optie is momenteel Firebase. Deze is vorig jaar aangekocht door Google en biedt naast het verbinden met een database ook een gestroomlijnde manier om met gebruikers te werken. Zo is het mogelijk om via standaardmethoden gebruikers te laten inloggen met OAuth [10].

Het nadeel van het gebruiken van een MBaaS is dat gratis formules meestal slechts een beperkt aantal gebruikers toelaten. In deze toepassing is dit echter niet zo relevant, aangezien er initieel geen groot gebruikersaantal verwacht wordt.

### Gekozen backend

In dit project is er gekozen om met de MBaaS Firebase te werken als backend omdat deze een goede integratie met OAuth biedt en zo logins en gebruikersbeheer vereenvoudigt. Bijgevolg kan er gefocust worden op het ontwikkelen van de applicatie zelf. Bovendien biedt Firebase een gratis pakket aan waarmee tot 100 gebruikers tegelijk toegang hebben tot de data [11].

# Technische detailuitwerking van gekozen oplossing (met controle en verificatie)

De applicatie wordt opgedeeld in drie delen: de webinterface, de backend en de Androidapplicatie. Bij alle delen wordt er gebruik gemaakt van Git voor versiebeheer.

## 2.1 Webinterface

De webinterface is bruikbaar voor werkgevers die de prestaties van hun werknemers willen bekijken. Ook is er een landingspagina die gebruikt wordt om mensen die het project willen leren kennen een voorsmaakje te geven alvorens de app te downloaden.

Er is geopteerd om gebruik te maken van GitHub Pages [12] om de statische site op een eenvoudige manier gratis beschikbaar te maken. Er is gekozen om hierbij gebruik te maken van Travis CI [13], een service voor Continuous Integration en Continuous Deployment [14]. Zo blijft de broncode duidelijk voor ontwikkelaars, maar worden ook geoptimaliseerde resultaten voor de gebruiker verkregen.

Hierdoor is de mappenstructuur (codefragment 1) relatief eenvoudig te verkrijgen. Er wordt gebruik gemaakt van Gulp [15] als tool om de build altijd op dezelfde manier te laten gebeuren. Zo kan eender wie op een eenvoudige manier meehelpen aan het project.

Codefragment 1: Mappenstructuur van de webapplicatie

├── assets // Onrechtstreeks gebruikt

├── dashboard // Map voor het dashboard

├── dist // Geproduceerd door de buildtool

│ ├── dashboard

│ ├── firebase

│ └── src

│ ├── css

│ ├── img

│ │ ├── favicon

│ │ ├── icons

│ └── js

├── firebase // Test in verband met firebase

├── includes // Code die op verschillende keren wordt gebruikt

├── node\_modules // Node.js dependencies

└── src // broncode

├── img // afbeeldingen

│ ├── favicon

│ └── icons

├── js // Scriptbestanden

└── scss // Stijlbestanden

## 2.2 Androidapplicatie

Om een Androidapplicatie te realiseren, wordt er gebruik gemaakt van Android Studio. Dit is de IDE die door Google is ontwikkeld om een Android app te programmeren.

Aan deze applicatie worden twee personen toegewezen. De ene zal zich verdiepen in de connectie met de backend en de locatieservices. De andere werkt aan het design van de app, zoals de verschillende vensters en lay-outs. De lijn tussen deze twee aspecten is vaag, aangezien ze met elkaar communiceren.

Android Studio is vrij vanzelfsprekend als het aankomt op het aanmaken van een project, er verschijnt een wizard om hulp te bieden. Zo kan er reeds een standaardlay-out gekozen worden en worden dergelijke codefragmenten al gegenereerd.

Codefragment 2: Mappenstructuur van de Android applicatie

├─── Gradle Scripts

│ │ build**.**gradle // Instellingen voor de project build en definitie

van afhankelijkheden

│ │ **...**

│

└─── app

│

├─── manifests

│ │ AndroidManifest**.**xml // Informatie over de app voor het

Android besturingssysteem

│

├─── java // Programmacode

├─── io**.**punchtime**.**punchtime **(**release**)**

│

├─── data // Data-abstractielaag

├─── ui // Presentatielaag: activities en fragments

└─── logic // Ondersteunende programmalogica voor de ui

│

│

└─── res // Resources

│

├─── drawable // Beeldmateriaal: afbeeldingen, iconen enzovoort

├─── lay-out // Applicatiedesign (XML-bestanden)

├─── menu // Navigatiestructuur (XML-bestanden)

├─── mipmap // Het logo van de app (in verschillende groottes)

└─── values // Statische inhoud (tekst, kleurenpalet)

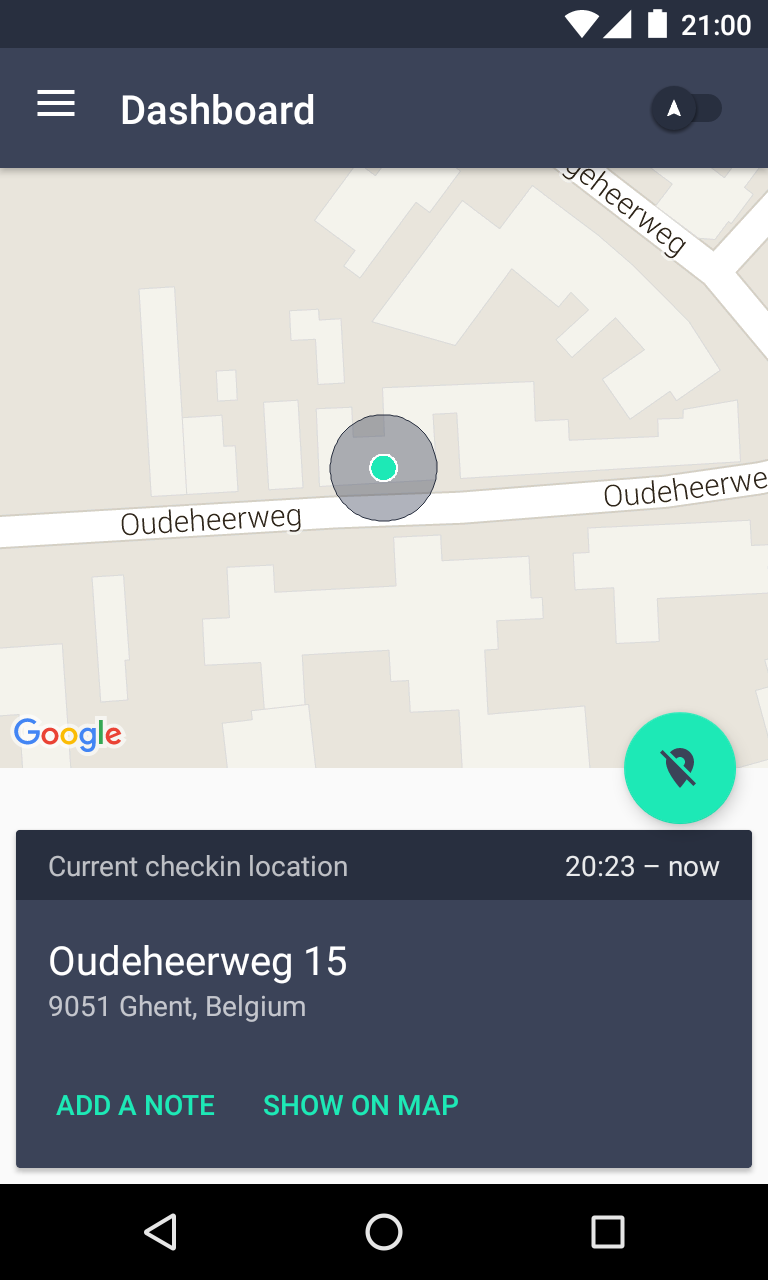
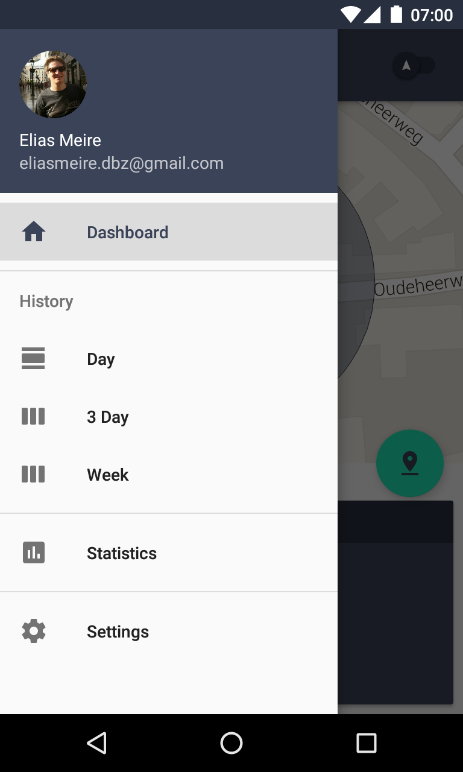
Het project heeft een vaste mappenstructuur (codefragment 2). Het eerste bestand is AndroidManifest.xml. Dat bestand bevat alle essentiële informatie over de app die het Androidbesturingssysteem nodig heeft vooraleer het de code uitvoert. Daarna komen de Javaklassen aan bod, er zijn benamingen vastgelegd voor sommige klassen. Zo moet een activiteit ActiviteitsnaamActivity genoemd worden.

Daarnaast is er ook een map, genaamd resources of kort res. Daarin komen alle bestanden die gebruikt worden voor de lay-out van de applicatie in terecht. Dit houdt iconen en afbeeldingen in, maar ook alle XML-bestanden voor de lay-out van de app en de vaste waarden die gebruikt worden, zoals kleuren en strings.

Als laatste zijn er nog een aantal Gradlescripts. Deze worden uitgevoerd wanneer het project gecompileerd wordt. Het haalt de nodige afhankelijkheden op en maakt dan het APK-bestand aan. Vervolgens wordt het APK-bestand geïnstalleerd op de smartphone en kan de app uitgevoerd worden.

### 2.2.1 Design en lay-out

Voor het design van de applicatie worden de Material Designrichtlijnen [16] gevolgd die door Google werden opgesteld. Er wordt gekozen voor een standaardlay-out waarbij een menu dynamisch openschuift. Op het beginscherm zal de gebruiker een dashboard (figuur 2) te zien krijgen met een kaart, een knop om zich in te checken en een detailweergave van de huidige check-in. In de titelbalk kan hij de optie voor automatische check-ins uit- of aanzetten. Telkens de gebruiker een check-in of check-out doet, wordt er visuele feedback gegeven door middel van een snackbar. Hierdoor is het duidelijk voor de gebruiker dat de actie wel degelijk gebeurd is.

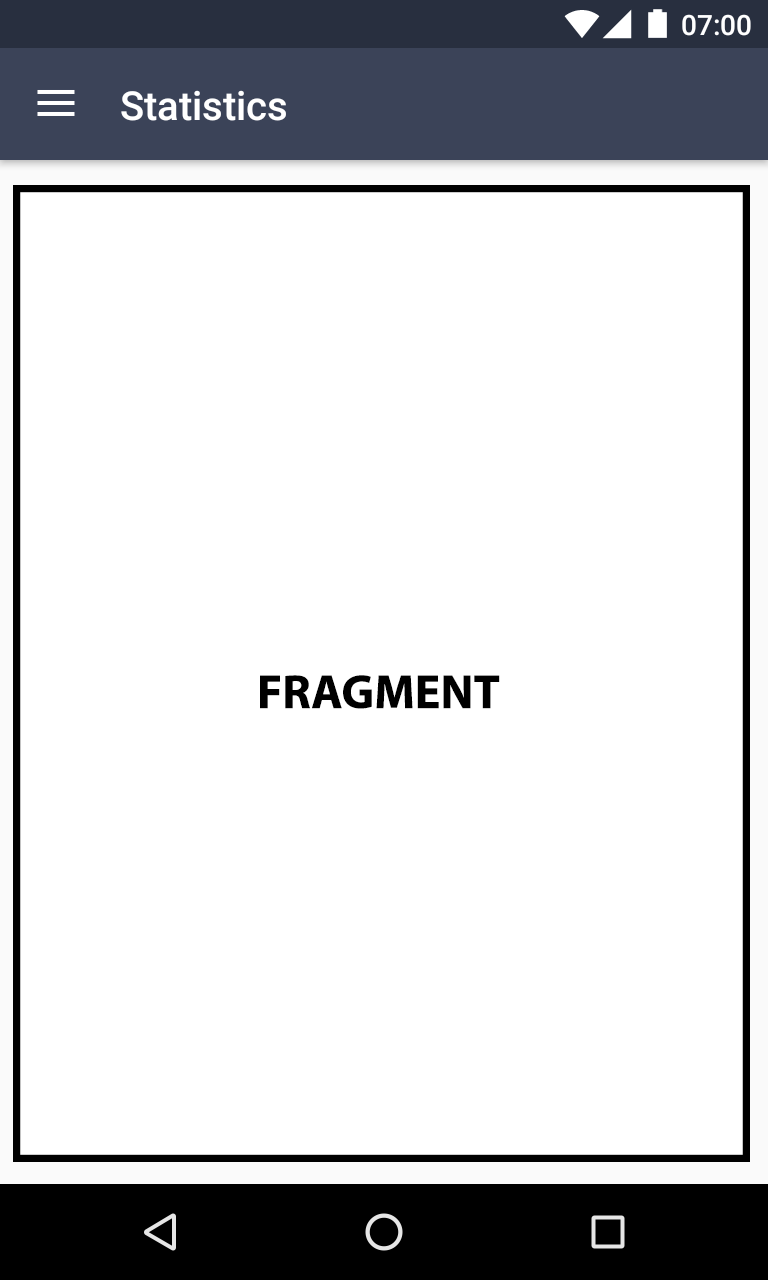
Het uitschuifmenu (figuur 3) is het belangrijkste navigatiemiddel in de applicatie. Het menu bevat links naar alle belangrijke delen van de applicatie: het dashboard, de geschiedenis onderverdeeld in verschillende tijdsintervallen, de werkstatisitieken van de gebruiker en de instellingen.

figuur 2: Startscherm van de Androidapplicatie

figuur 3: Uitschuifmenu van de Androidapplicatie

De app is onderverdeeld in activiteiten en fragmenten. Bij de activiteiten is er enerzijds sprake van een activiteit die een detailweergave omvat van de locatie op een kaart. Anderzijds is er een hoofdactiviteit voor alle andere functionaliteit. Voor elke activiteit wordt er een Javaklasse voorzien met de nodige logica.

De hoofdactiviteit bevat enkel vaste elementen van de gebruikersinterface. Dat omvat de titelbalk en het menu. De eigenlijke inhoud van het scherm, zoals bijvoorbeeld het dashboard, wordt dynamisch ingeladen aan de hand van de navigatie van de gebruiker.

Om deze functionaliteit te bekomen wordt er gebruik gemaakt van fragmenten (figuur 4). Vanuit de activiteit worden de nodige fragmenten dynamisch ingeladen, wat resulteert in een goede performantie terwijl de gebruiker door de app navigeert. Bij elk van deze fragmenten hoort een apart XML-bestand en een Javaklasse. Het XML-bestand bevat de lay-out van de verschillende elementen en de klasse bevat de logica die erop toegepast wordt.

figuur 4: Dynamisch fragment in een activiteit

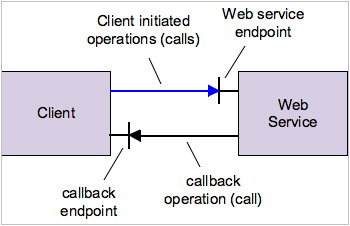
### 2.2.2 Programmalogica

#### 2.2.2.1 Hoofdactiviteit

##### 2.2.2.1.1 Connectie met backend

De connectie met de Firebasebackend is geprogrammeerd in Java. De data die wordt opgehaald uit de backend wordt eerst omgezet naar Javaobjecten. Dat zorgt ervoor dat er in de verdere programmacode gemakkelijk kan gewerkt worden met deze data.

Data wordt asynchroon uit de backend opgehaald via callbacks [17] (figuur 5). Ook opslaan van data in de backend gebeurt asynchroon, waardoor de performantie van de applicatie niet gehinderd wordt.



figuur 5: Principe van asynchrone communicatie en callbacks [18]

##### 2.2.2.1.2 Authenticatie

Firebase voorziet code voor authenticatie waardoor deze niet in de app moet geschreven worden. Authenticatie van gebruikers is een van de gevaarlijkste aspecten van een applicatie, maar door Firebase te gebruiken worden hier allerlei beveiligingsrisico’s uitgesloten. Tevens heeft Firebase integratie met verschillende loginproviders zoals Google, Facebook en Twitter. Dat laat gebruikers toe in te loggen met een account van die providers met als gevolg dat gebruikers geen lang formulier moeten invullen om de app te kunnen gebruiken.

De status van de authenticatie wordt door Firebase doorgegeven door middel van een callback. In deze callback wordt de gebruikersinterface geüpdatet om de ingelogde gebruiker weer te geven. Tevens worden er in deze callback bepaalde features uitgeschakeld zodat niet ingelogde gebruikers hier geen toegang toe hebben. Fragmenten kunnen ook de huidige authenticatiestatus opvragen aan de hoofdactiviteit. Dat is bijvoorbeeld nuttig in ons instellingenfragment want een persoon die niet ingelogd is mag bijvoorbeeld geen lid worden van een bedrijf.

#### 2.2.2.2 Dashboard

##### 2.2.2.2.1 Locatieservice

Als locatieservice wordt er gekozen voor Google Maps. Via de API kan er gemakkelijk een locatie opgevraagd worden en kunnen de coördinaten opgeslagen worden via Firebase.

Op het dashboard zal er zich een kaart bevinden met de huidige locatie. Wanneer de gebruiker de kaart aanraakt, wordt het dashboard vervangen door een grotere kaart en kan men de locatie eventueel manueel verfijnen.

Om het batterijverbruik van de locatieservice te beperken wordt er gebruik gemaakt van de FusedLocationProvider API van Android. De FusedLocationProvider API [19] zal in de achtergrond de locatie bepalen aan de hand van alle beschikbare sensors. Indien het niet nodig is zal de gps-sensor hierbij niet gebruikt worden. Dit is belangrijk omdat de gps-sensor meer batterij verbruikt dan de andere sensors.

Om de locatie van de gebruiker te tonen op de kaart wordt er een interval ingesteld van één seconde [20]. Dat tijdsinterval is een compromis tussen batterijverbruik en locatienauwkeurigheid. Met de FusedLocationProvider API kan de locatie soms ook frequenter worden bepaald dan het ingestelde interval. Dat gebeurt enkel als er een andere applicatie in de achtergrond ook de locatie opvraagt. Bijgevolg vereist dit geen extra batterijverbruik.

##### 2.2.2.2.2 Reverse geocoder

De locatieservice geeft telkens de locaties in coördinaten weer. Dit is echter geen leesbaar formaat voor de gebruiker. Om coördinaten om te zetten naar een leesbaar adres moet er gebruik gemaakt worden van een API. Er werd gekozen voor de Reverse Geocoding API [21] van Google Maps. Omdat aanvragen naar deze API soms enige tijd in beslag nemen, is het belangrijk dat deze asynchroon gebeuren. Om dat te realiseren wordt er gebruik gemaakt van de AsyncTask interface [22] van Android.

#### 2.2.2.3 Geschiedenis

Om de geschiedenis van de check-ins en check-outs weer te geven, wordt er gebruik gemaakt van een kleine bibliotheek die data weergeeft op een kalender. Deze bibliotheek is volledig open-source en beschikbaar op GitHub [23].

Er zijn drie verschillende versies van de geschiedenis beschikbaar: events per dag, per drie dagen of per week. Via de bibliotheek is het relatief gemakkelijk om in te stellen hoeveel dagen er moeten weergegeven worden.

Alle check-ins worden aan een lijst van Javaobjecten toegevoegd. Nadien wordt de lijst overlopen en wordt de nodige data van elk object opgehaald. Deze data wordt opnieuw aan een nieuwe lijst toegevoegd. Dit is dan de uiteindelijke lijst met events die zal weergegeven worden op de kalender.

Wanneer de gebruiker een specifiek event aanraakt, wordt de data voor dit event weergegeven. Via de “Toon op kaart” knop wordt de gebruiker doorgestuurd naar de map met de locatie van de check-in. Hierin wordt simpelweg doorverwezen naar de map en worden de coördinaten meegegeven van de check-in. Er wordt gecontroleerd of er reeds een notitie ingevoerd is voor een event en vervolgens wordt de tekst van de knop aangepast. Als er reeds notitie aanwezig is, kan de notitie aangepast worden. Zo niet kan er een nieuwe notitie toegevoegd worden. De data van de check-in wordt gezocht in de backend en de notitie wordt aangepast.

#### 2.2.2.4 Statistieken

Om de statistieken te berekenen worden alle check-ins opgehaald uit de laatste week. Uit deze check-ins wordt berekend hoeveel tijd er al werd gewerkt in de huidige week en in de huidige dag. Deze gegevens worden vervolgens weergegeven op een grafiek.

De grafiek is gerealiseerd met behulp van de DecoView library [24] die gratis beschikbaar is op GitHub. Met deze library kan de grafiek ook worden geanimeerd.

#### 2.2.2.5 Lokalisatie

Er is gezorgd voor lokalisatie in de applicatie. Afhankelijk van de van de systeeminstellingen zal de taal in de applicatie aangepast worden. Voor deze applicatie is er een Engelse, Franse en Nederlandse vertaling beschikbaar.

## Backend

Voor de backend is er gebruik gemaakt van Firebase, een NoSQL MBaaS. De databank is georganiseerd als een boomstructuur. Er is een tak met data over de bedrijven en een tak met data over de de gebruikers, deze tak bevat ook de lijst met check-ins. In codefragment 3 is een voorbeeld te zien van een check-in van een gebruiker in Firebase.

"-KI9Nn4u3NrtVv6MJ3c1" **:** **{**

"addressCityCountry" **:** "9051 Ghent, Belgium"**,**

"addressStreet" **:** "Oudeheerweg 15"**,**

"check-in" **:** 1463684378462**,**

"check-out" **:** 0**,**

"confirmed" **:** **true,**

"employee" **:** "e44f77b1-b82e-4584-89d6-50858b2862d8"**,**

"employer" **:** "-KBdSPf90dvJCeH3J8m7"**,**

"latitude" **:** 51.0203347**,**

"longitude" **:** 3.6733156**,**

"note" **:** "some note"

**}**

codefragment 3: Check-in in Firebase

Verder is er een tak voor alle uitnodigingen. Elk van de uitnodigingen bevat het e-mailadres, het bedrijf die de uitnodiging verstuurd heeft en nog een aantal velden. Wanneer een werknemer een uitnodiging opneemt komt er eerst een veld bij met zijn of haar id. In codefragment 4 is een voorbeeld te zien van een uitnodiging die reeds opgenomen is door een gebruiker.

"-KFxr1QE0y6zhcTaSf\_y" **:** **{**

"claimed" **:** **true,**

"company" **:** **{**

"id" **:** "-KBdSPf90dvJCeH3J8m7"**,**

"name" **:** "my-company"

**},**

"email" **:** "hello@haroen.me"**,**

"sent" **:** **true,**

"user" **:** "google:116529723379255029542"

**}**

codefragment 4: Uitnodiging in Firebase

## Versiebeheer

Versiebeheer is vereist om de aanpassingen van verschillende personen te harmoniseren. Git is hiertoe de beste optie, de Gitrepository wordt gehost op GitHub, een populaire website voor allerhande opensourceprojecten. Op GitHub kan men gemakkelijk het volledige project bekijken en alle aanpassingen beheren. Bovendien kunnen ook buitenstaanders of zelfs gebruikers hier problemen melden of bijdragen aan de code.

Er zijn verschillende methodieken die het werken met versiecontrole gestroomlijnder maken. In dit project wordt er gebruik gemaakt van Gitflow (figuur 4). Gitflow [25] is een geheel van een aantal richtlijnen voor het gebruik van Git in teamverband. In Gitflow wordt er gebruik gemaakt van een master en een develop branch. In de master branch hoort enkel code die klaar is voor een release. Tijdens de ontwikkeling wordt er steeds gebruik gemaakt van de develop branch. Het werk wordt opgedeeld in kleine features die dan steeds een eigen branch vormen gebaseerd op de develop branch. Telkens als er een feature klaar is, wordt de code terug in de develop branch gebracht en wordt de feature branch verwijderd. Er kan ter aanvulling ook met bugfix- en hotfixbranches gewerkt worden.

figuur 6: Gitflow branchingmodel

# Risicoanalyse

## 3.1 Beveiliging van gebruikersgegevens

Bij het maken van een applicatie is de grootste uitdaging het beveiligen van de gebruikersdata. De data van de applicatie bevat gevoelige informatie zoals wanneer werknemers werken en waar ze zich bevinden. Dankzij het gebruik van Firebase [26] is het mogelijk deze data op een efficiënte manier te beschermen. Bovendien kan er gebruik gemaakt worden van externe accounts om in te loggen, zoals Facebook of Google. Op deze manier worden enkel aanvragen met juiste logingegevens aanvaard door de server. Indien de logingegevens fout zijn, wordt er een foutmelding teruggestuurd. Configuratie voor deze veiligheidsbestanden ziet eruit als de code in codefragment 4.

Codefragment 4: Firebase beveiligingscode

**{**

"rules"**:** **{**

"users"**:** **{**

"$uid"**:** **{**

".read"**:** "auth != null && auth.uid === $uid"**,**

".write"**:** "auth != null && auth.uid === $uid"

**}**

**},**

"pulses"**:** **{**

"$pulse"**:** **{**

".read"**:** "auth != null && auth.uid === data.child('name').val()"**,**

".write"**:** "auth != null && auth.uid === data.child('name').val()"**,**

".validate"**:** "newData.hasChildren(['name', 'time', 'employer', 'type']) && data.parent().child('companies').child('employee').child(newData.child('employer').val()).exists()"

**}**

**},**

"companies"**:** **{**

"$company"**:** **{**

".read"**:** "auth != null"**,**

".write"**:** "auth != null"**,**

".validate"**:** "newData.child('employees').key() == auth.uid || newData.child('employers').key() == auth.uid "

**}**

**}**

**}**

**}**

## 3.2 Fouten in de locatiebepaling

Bij het automatisch bijhouden van de locatie moet er rekening mee gehouden worden dat deze fout kan zijn. Vooral bij het leggen van de link tussen coördinaten en adres kan het vaak fout gaan. Een oplossing ligt echter voor de hand, de gebruiker zal de mogelijkheid moeten hebben om handmatig in te checken in een locatie.

Deze oplossing introduceert echter wel een nieuw probleem, want de gebruiker zou namelijk een verkeerd adres kunnen ingeven. Om dit te omzeilen werd er toch geopteerd om de automatisch bepaalde adressen te gebruiken. De fouten die de Reverse Geocoding API van Google Maps maakt zijn klein en het ingeven van foute locaties wordt vermeden.

## Manuele tijdsinstelling

Bij het inchecken en uitchecken wordt het huidige tijdstip opgehaald. Er moet rekening gehouden worden met de mogelijkheid dat de gebruiker het huidige tijdstip kan aanpassen op het toestel. Zo kan er bijvoorbeeld een onlinecontrole uitgevoerd worden, of kan de tijd meteen online opgehaald worden.

## 3.4 Andere

Andere moeilijkheden zijn eerder technisch van aard. Zo moet er bijvoorbeeld voor gezorgd worden dat bij het bijhouden van de locatie niet te veel batterij gebruikt wordt. Dat probleem werd al besproken in het vorige hoofdstuk.

# Kostenraming en levensduur

## 4.1 Kostenraming

|  |  |  |
| --- | --- | --- |
| **Beschrijving van de kost** | **Begunstigde** | **Kostprijs** |
| Domeinnaam eerste jaar  Domeinnaam vernieuwing | Namecheap | € 30,81  € 29,37 per jaar |
| Hostingkosten op GitHub Pages | GitHub | gratis |
| SSL-certificaat | CloudFlare | gratis |
| Hostingkosten backend | Firebase | gratis |
| Hostingkosten email-backend | Uberspace | € 1,00 per maand |
| Google Play eenmalige registratiekosten | Google | € 22,29 |
| **Totaal (eerste jaar)**  **Totaal (volgende jaren)** |  | **€ 65,10**  **€ 41,37 per jaar** |

De domeinnaam “punchtime.io” werd aangekocht op Namecheap. De prijs hiervan bedroeg 30,81 euro, samen met 1 jaar gratis WhoIsGuard, deze beschermt de gegevens van de eigenaar. Het vernieuwen van de domeinnaam kost 29,37 euro per jaar. Een extra jaar WhoIsGuard kost 2,57 euro.

De website wordt gehost op GitHub Pages. GitHub stelt deze service ter beschikking voor het hosten van statische webpagina’s. Er kan dus geen backend gehost worden op GitHub. Er staat geen limiet op het webverkeer via GitHub pages. Daardoor komen er geen kosten bij ook al zou de site een groot aantal bezoekers krijgen.

Het SSL-certificaat wordt verkregen doordat het verkeer via een gratis formule van CloudFlare gerouteerd wordt, dit verzekert dat de verbinding tussen de website en de gebruiker veilig is.

De backend werkt voorlopig op de gratis pakket van Firebase. Hiervoor staan limieten vast en naarmate de applicatie groeit, zal het nodig zijn om over te schakelen naar een betaald pakket. Deze stijgt snel in prijs en zal zonder twijfel de grootste kost zijn bij een uitbreiding.

Voor het verzenden van uitnodigingen via email naar de gebruikers wordt er gebruik gemaakt van een extra kleine backend. Hosting voor deze backend wordt aangekocht met een PWYW-formule (Pay What You Want) bij Uberspace.

Bij het publiceren van een applicatie op de Google Play Store moet een eenmalige prijs van 22,37 euro betaald worden. Deze kost is voor de registratie van een ontwikkelaar en zal dus niet per applicatie moeten worden betaald.

Met al deze kosten opgeteld wordt er een eenmalige opstartkost van 65,10 euro en een onderhoudskost van 41,37 euro per jaar bekomen. Er wordt hierbij rekening gehouden met een klein aantal gebruikers. Indien dit aantal stijgt zou het noodzakelijk zijn om over te gaan op een betalende formule van Firebase. Dat zou de jaarlijkse onderhoudskosten aanzienlijk verhogen.

## Levensduur

Momenteel is er zekerheid van het blijven bestaan van de applicatie tot minstens mei 2017. Indien er gemerkt wordt dat er interesse komt, zou dit eenvoudig kunnen verlengd worden. Als er grote interesse komt in de applicatie, zouden de hostingkosten aanzienlijk stijgen en zou er een betalende formule of een donatieformule voor de applicatie moeten ingevoerd worden. Zo niet zou het onderhoud van de applicatie te veel geld en tijd in beslag nemen en wordt de mogelijkheid geboden aan de gebruikers om hun data af te halen en over te zetten naar een ander platform.

Indien er een betalende versie zou worden ingevoerd zou er een maandelijkse som worden gevraagd aan de werkgever. Voor de werknemers zal de applicatie uiteraard altijd gratis en vrij van advertenties blijven.

# Conclusie

Er wordt een werkend product afgeleverd dat een interface voor werkgevers biedt om de werkuren van hun werknemers na te kijken en te bevestigen. Werknemers kunnen hun locatie met een mededeling delen van op hun Android smartphone en kunnen hun huidige status nakijken.

Er zijn uiteraard enkele mogelijkheden tot uitbreiding mogelijk. De meest dringende is om meer mogelijkheden aan de pagina voor beheerders toe te voegen. Zo kan er vooral in de richting van onmiddellijke communicatie met werknemers gekeken worden, alsook het genereren van rapporten over de werknemers.

In de applicatie kan er een automatische modus uitgewerkt worden die automatisch de werknemers in- en uitcheckt aan de hand van hun locatie. Dit zou het gemakkelijker maken voor werknemers aangezien zij niet meer handmatig zouden moeten inchecken.

Technologisch gezien is het verder ook mogelijk om te groeien naar andere platformen. Met Android is al een groot deel van de doelgroep bereikt, maar iOS heeft ook een groot marktaandeel. Het zou het mogelijk zijn om ook voor iOS een applicatie te ontwikkelen.

Ook aan de marketingkant is er groei mogelijk. Er kan gezocht worden naar proefbedrijven die hun wensen kunnen geven rond reclame zodat de applicatie automatisch vlotter gevonden kan worden. Ook publicaties in de pers etc. kunnen hier een bijdrage toe leveren, zodat er in totaal een nog uitgebreider geheel gevormd wordt. Hierbij kan er ook hulp van derden ingeschakeld worden.

Tijdens de week waarin dit project ingediend werd, werd er een nieuwe versie van Firebase aangekondigd op Google I/O 2016. De nieuwe versie omvat een groot aantal nieuwe functies. In de toekomst kan er overgeschakeld worden naar de nieuwe versie voor onder andere betere analytics, berichten enzovoort.

# Literatuurlijst

|  |  |
| --- | --- |
| [1] | Wijmo, „Mobile App Development: How to Decide on Hybrid vs. Native,” 10 december 2015. [Online]. Available: http://wijmo.com/mobile-app-development-how-to-decide-on-hybrid-vs-native/. [Geopend 9 maart 2016]. |
| [2] | A. Ziflaj, „Native vs Hybrid App Development,” Sitepoint, 15 augustus 2014. [Online]. Available: http://www.sitepoint.com/native-vs-hybrid-app-development/. [Geopend 8 maart 2016]. |
| [3] | „Programmeren 2,” Odisee, [Online]. Available: http://onderwijsaanbod.odisee.be/2015/syllabi/n/JPW274N.htm#activetab=doelstellingen\_idp1159136. [Geopend 11 maart 2016]. |
| [4] | Ramon Llamas, „Smartphone OS Market Share, 2015 Q2,” IDC, augustus 2015. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Geopend 1 maart 2016]. |
| [5] | „What are the pros and cons of using Apache Cordova (phone-gap) for building your iOS app?,” Quora, 24 juli 2012. [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-of-using-Apache-Cordova-phone-gap-for-building-your-iOS-app. [Geopend 5 maart 2016]. |
| [6] | R. Kraus, „From Native To Hybrid App Development And Back,” TechCrunch, 19 november 2015. [Online]. Available: http://techcrunch.com/2015/11/19/lessons-in-switching-from-native-to-hybrid-app-development-and-back/. [Geopend 3 maart 2016]. |
| [7] | S. Almog, „Java is superior to React Native in practically every way,” Codename One, 2 november 2015. [Online]. Available: http://www.codenameone.com/blog/java-is-superior-to-react-native-in-practically-every-way.html. [Geopend 6 maart 2016]. |
| [8] | „It is possible to watch the location in the background on Mobile (iOS / Android)?,” Stackoverflow, 15 november 2015. [Online]. Available: http://stackoverflow.com/questions/33926217/it-is-possible-to-watch-the-location-in-the-background-on-mobile-ios-android. [Geopend 10 maart 2016]. |
| [9] | „Should I use a MBaaS platform (like Parse) or create my own backend?,” Quora, [Online]. Available: https://www.quora.com/Should-I-use-a-MBaaS-platform-like-Parse-or-create-my-own-backend. [Geopend 10 maart 2016]. |
| [10] | Firebase, „Firebase Authentication,” Firebase, [Online]. Available: https://www.firebase.com/features.html#features-authentication. [Geopend 11 maart 2016]. |
| [11] | Firebase, „Pricing & Plans,” Firebase, [Online]. Available: https://www.firebase.com/pricing.html. [Geopend 11 maart 2016]. |
| [12] | „Github Pages,” [Online]. Available: https://pages.github.com. [Geopend 5 april 2016]. |
| [13] | „Travis CI,” [Online]. Available: https://docs.travis-ci.com/user/getting-started/. [Geopend 6 april 2016]. |
| [14] | B. Fitzgerald, „Continuous Software Engineering and Beyond: Trends and Challenges (PDF). 1st International Workshop on Rapid Continuous Software Engineering,” 3 juni 2014. [Online]. Available: http://staff.lero.ie/stol/files/2014/03/rcose2014\_fitzgerald.pdf. [Geopend 1 april 2016]. |
| [15] | Medium, „Gulp vs. Grunt. Why one why the other,” [Online]. Available: https://medium.com/@preslavrachev/gulp-vs-grunt-why-one-why-the-other-f5d3b398edc4#.6zx40w59g. [Geopend 15 maart 2016]. |
| [16] | Google, „Material Design,” Google, [Online]. Available: https://www.google.com/design/spec/material-design/introduction.html. [Geopend 16 maart 2016]. |
| [17] | H. Pennington, „Callbacks, synchronous and asynchronous,” [Online]. Available: http://blog.ometer.com/2011/07/24/callbacks-synchronous-and-asynchronous/. [Geopend 20 maart 2016]. |
| [18] | Oracle, „Asynchronous communication,” [Online]. Available: http://www.oracle.com/ocom/groups/public/@otn/documents/digitalasset/108452.gif. [Geopend 18 april 2016]. |
| [19] | Android Developer, „FusedLocationProviderApi,” [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi. [Geopend 2 mei 2016]. |
| [20] | Android Developer, „Receiving Location Updates,” [Online]. Available: https://developer.android.com/training/location/receive-location-updates.html. [Geopend 15 april 2016]. |
| [21] | Google, „Geocoding API,” [Online]. Available: https://developers.google.com/maps/documentation/geocoding/intro. [Geopend 20 april 2016]. |
| [22] | Android Developer, „AsyncTask,” [Online]. Available: https://developer.android.com/reference/android/os/AsyncTask.html. [Geopend 25 april 2016]. |
| [23] | A. Kanak, „Android Week View,” [Online]. Available: https://github.com/alamkanak/Android-Week-View. [Geopend april 20 2016]. |
| [24] | B. Marr, „DecoView,” [Online]. Available: https://github.com/bmarrdev/android-DecoView-charting. [Geopend 24 april 2016]. |
| [25] | V. Driessen, „A successful Git branching model,” [Online]. Available: http://nvie.com/posts/a-successful-git-branching-model/. [Geopend 2 april 2016]. |
| [26] | „Securing Your Data,” Firebase, 2015. [Online]. Available: https://www.firebase.com/docs/security/guide/securing-data.html. [Geopend 20 maart 2016]. |
| [27] | Android Developer, „Toasts in Android,” [Online]. Available: http://developer.android.com/intl/in/guide/topics/ui/notifiers/toasts.html. [Geopend 28 maart 2016]. |

# Bijlage 1: Uiteindelijke werkplanning

De werkplanning is opgemaakt per lesweek. Er is geen onderscheid gemaakt tussen personen omdat de taken meestal door meerdere personen worden uitgevoerd. Voor meer gedetailleerde informatie over welke persoon welke taak heeft uitgevoerd kan de Gitrepository worden geraadpleegd.

|  |  |  |
| --- | --- | --- |
| **Week** | **Technisch** | **Rapportering** |
| 1 | Opzoeken van technologieën die we kunnen gebruiken voor zowel front- en backend. | Inleiding schrijven. |
| 2 | Opgezochte technologieën tegen elkaar afwegen om zo te beslissen welke er zullen gebruikt worden. | Inleiding afwerken. |
| 3 | Nadenken over het concept van onze applicatie en interfaces schetsen voor de frontend. Extra kennis verwerven over de ontwikkeling van Androidapplicaties en alle beschikbare API’s. Beginnen met de ontwikkeling van de Androidapplicatie. | Het mogelijke oplossingen hoofdstuk schrijven en de inleiding verbeteren aan de hand van de feedback van de docent. |
| 4 | Afwerken van het design van onze applicatie. Designen en realiseren van een backend voor onze applicatie met behulp van Firebase. | Het mogelijke oplossingen hoofdstuk afwerken. |
| 5 | De interface voor de frontends realiseren aan de hand van het eerder gemaakte design. Hierbij maken we gebruik van de gekozen technologieën zoals gulp en Android design support library. | Risicoanalyse schrijven en het mogelijke oplossingen hoofdstuk verbeteren aan de hand van de feedback van de docent. |
| 6 | Firebase verbinden met beide frontends en testen of ons backenddesign voldoet aan de praktische noden van de frontends. | Risicoanalyse afwerken. |
| 7 | Authenticatie implementeren in de Androidapplicatie. Hierbij worden er met behulp van Firebase ook loginproviders zoals Google, Facebook en Twitter voorzien. | Het hoofdstuk over de detailuitwerking schrijven. |
| 8 | Dashboard van de Androidapplicatie uitwerken. Check-ins en check-outs mogelijk maken en deze tevens persisteren in de Firebase. | Verder schrijven aan het hoofdstuk over de detailuitwerking. |
| 9 | Mogelijkheid om de check-ingeschiedenis te bekijken in de Androidapplicatie. Om dit gebruiksvriendelijk te maken worden er drie verschillende kalenderweegaves gemaakt. Een die de check-ins per dag toont, een per drie dagen en een per week. | Verder schrijven aan het hoofdstuk over de detailuitwerking. |
| 10 | Gebruikers verbinden met hun bedrijf met behulp van een uitnodigingssysteem per mail. Om deze mails te versturen wordt er gebruik gemaakt van een Node.js backend en Mailchimp. In de Androidapplicatie wordt app linking geïmplementeerd zodat gebruiker vanuit hun uitnodigingsmail direct naar app worden gebracht via een link. | Kostenraming opmaken en een analyse maken van de levensduur van ons project. |
| 11 | Statistieken implementeren in de Androidapplicatie zodat gebruikers kunnen zien hoeveel ze al gewerkt hebben op de huidige dag en in de huidige week. | Conclusie schrijven. |
| 12 | Bugs en crashes oplossen in de Androidapplicatie. Tests uitvoeren voor randsituaties in de webinterface. | Het rapport afwerken en grondig nalezen. Het geheel toetsen aan de gegeven richtlijnen over het rapport en wijzigingen aanbrengen waar nodig. |