FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

EINDVERSLAG

VAKOVERSCHRIJDEND EINDPROJECT

PARK&RIDE ROUTEPLANNER STAD GENT

TEAM PR-01

Sam DE CREUS
Gauthier MESSIAEN
Wout PROVOST
Bert ROMAN
Jenthe THIENPONDT
Seppe VANHEE

UNIVERSITEIT
GENT

Academiejaar 2016–2017
Schakelprogramma in de industriële wetenschappen: informatica
Valentin Vaerwyckweg 1, 9000 Gent

FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

EINDVERSLAG

VAKOVERSCHRIJDEND EINDPROJECT

PARK&RIDE ROUTEPLANNER STAD GENT

TEAM PR-01

Sam DE CREUS
Gauthier MESSIAEN
Wout PROVOST
Bert ROMAN
Jenthe THIENPONDT
Seppe VANHEE

UNIVERSITEIT
GENT

Academiejaar 2016–2017
Schakelprogramma in de industriële wetenschappen: informatica
Valentin Vaerwyckweg 1, 9000 Gent

Inhoudsopgave

Samenvatting

Sinds 2014 is de stad Gent bezig met het opzetten van een monitoringssysteem voor de verkeerssituatie. Dit grootschalig project heeft als doel het verkeer zo vlot en veilig mogelijk te laten verlopen voor alle weggebruikers. Hierbij wil men zo vlug mogelijk kunnen inspelen op onverwachte incidenten om opstroppingen en files te voorkomen. De stad stelt de verzamelde real-time gegevens ter beschikking via verschillende media. Twitter is de eenvoudigste manier om mensen in staat te stellen voor zichzelf de beste route te plannen. Anderzijds wordt de data ook in een technisch formaat aangeboden, waarmee derden applicaties kunnen ontwikkelen die met de actuele verkeerssituatie rekening houden.

De stad wil ook het gebruik van Park & Rides promoten. Aan de rand van de stad zijn verschillende parkings die pogen het verkeer in de binnenstad te reduceren. Door mensen daar hun auto te laten parkeren en vervolgens het openbaar vervoer te laten nemen naar het centrum, wil men het gebrek aan parkeerplaats in de stadskern oplossen. Dit systeem is echter nog niet bij iedereen ingeburgerd omdat de informatie daarrond nog niet voldoende beschikbaar is. Met een Park & Ride routeplanner wil men hiervoor in de toekomst een oplossing bieden. Door een dashboard beschikbaar te stellen waar men de vertrek- en eindplaats kan ingeven, kan de gebruiker zijn verschillende trajectmogelijkheden opvragen.

Bij de start van de casus werd een uitgebreide studie gemaakt van de opgave, enerzijds gegeven door de opdrachtgever, anderzijds met de informatie die de projectmanagers beschikbaar heeft gesteld. Hieruit vloeiden een technologische analyse en deployment diagram voort, waarmee het softwareproject werd gelanceerd. Daarna werd de focus verlegd op het verloop van de applicatie. Aan de hand van de bestaande mockups van de opdrachtgever en een zelf ontworpen mockup voor mobiele gebruikers werden use cases, verscheidene ontwerpdiagrammen en een functionele specificatie opgesteld. Uiteindelijk werd door elk lid van het team een grondige API studie uitgevoerd en kon de implementatie van start gaan.

Als laatste werd ook grondig aandacht besteed aan de user interface (UI) en user experience (UX). De applicatie kreeg een moderne aantrekkelijke look en gebruiksvriendelijk design.

Voorwoord

Dit werk is uiteraard niet enkel onze eigen verdiensten tot stand gekomen. We willen daarom in eerste plaats de Universiteit Gent en onze projectmanager Mevr. Helga Naessens bedanken voor de productieomgeving in hun datacenter beschikbaar te stellen en de feedback doorheen deze periode. We willen ook onze coaches Veerle Ongenae, Martijn Saelens en Pieter-Jan Maenhaut bedanken voor hun hulp bij onze technische vragen en problemen. Als laatste mogen we uiteraard onze opdrachtgever de heer Pieter Morlion niet vergeten. Hartelijk dank aan allen voor jullie bijdrage! Sam, Gauthier, Wout, Bert, Jenthe en Seppe.

Hoofdstuk 1

Inleiding

Park & Rides (P&R) zijn een manier om het autoverkeer in de binnenstad terug te dringen. De parkings, gelegen aan de rand van de stad, vangen het verkeer reeds buiten het centrum op en bieden bijkomend een aansluiting met het openbaar vervoer of de fiets. Het is bovendien goedkoper om een Park & Ride te combineren met een tram- of busticket van De Lijn dan te parkeren in het stadscentrum. Daarom wil de stad Gent het gebruik van Park & Rides ook promoten. Momenteel staat er echter nog geen gebruiksvriendelijk dashboard ter beschikking voor geïnteresseerden.

Dit project zal het dashboard proberen tot stand te brengen. De implementatie bestaat uit een web- en serverapplicatie die vooral de data van de stad gebruikt met de beschikbaar gestelde Application Programming Interfaces (API's). De belangrijkste aspecten, naast een aantrekkelijk design, zijn gebruiksvriendelijkheid en intuïviteit. De applicatie moet snel en eenvoudig te begrijpen zijn zoals bij bestaande routeplanners. Daarnaast moet de gebruiker ook genoeg geïnformeerd worden over de reisweg, zodat hij gemakkelijk extra opties en alternatieve routes naar zijn wens kan selecteren.

Via Universiteit Gent kwamen de studenten in contact met Dhr. Morlion, die de opdracht verder kwam voorstellen. Uit zijn vereisten en uit eigen analyse, werden enkele requirements afgeleid, die in hoofdstuk 2; opgelijst staan. Dat hoofdstuk bespreekt ook de nodige use cases met domeinregels, een use case diagram en een feature lijst.

Verder behandelt dit document de verscheidene analysediagrammen in hoofdstuk 3; de systeemarchitectuur. Die diagrammen werden, net zoals de documentatie van de code en de use cases, in het Engels geschreven. Daardoor wordt de applicatie gemakkelijk herbruikbaar voor andere steden en leesbaar voor anders-

taligen.

In hoofdstuk 4: het testplan, staat meer informatie over de toegepaste testen en in hoofdstuk 5; de installatiehandleiding, wordt stap voor stap uitgelegd hoe men de applicatie zelf kan opzetten, welke programma's hiervoor nodig zijn en waar deze te vinden zijn.

Tijdens de ontwikkeling is de applicatie bereikbaar op een server[1] van de Universiteit Gent en is de repository beschikbaar op GitHub[2].

[1] http://95.85.34.130:8080
[2] https://github.ugent.be/iii-vop2017/pr-01

Hoofdstuk 2

Gebruikersaspecten

2.1 High-level requirements

1. **Algemene vereisten**
   (a) Waar zijn de P&R's gelokaliseerd
2. **Vereisten per P&R**
   (a) Aantal vrije plaatsen
   (b) Kostprijs
   (c) Betaalmogelijkheden
   (d) Passerende bus- en tramlijnen (met visualisatie op de kaart)
   (e) Frequentie van bussen en trams (om de hoeveel minuten is er een tram/bus, wanneer is de laatste tram/bus naar de stad, wanneer is de laatste tram/bus terug naar de P&R)
3. **Vereisten per individuele gebruiker**
   (a) Bereken de meest geschikte P&R locatie
   (b) Toon de geschatte verplaatsingsduur (inclusief openbaar vervoer) van A naar B
   (c) Toon de verschillende tram- en busverbindingen
   (d) Bereken de totale kostprijs

   (e) Geef alternatieven qua kostprijs en reistijd (parkeren op straat of ondergronds ...)

   (f) Betaalmogelijkheden: toon waar te betalen, of er een betaalautomaat is, geef een link naar de app ...

4. **Vereisten bij de berekeningen**

   (a) Breng onverwachte incidenten in rekening en toon ook wat de extra reistijd zal zijn (Analoog met de NMBS app die u toont hoeveel minuten vertraging een trein heeft)

   (b) Het nachtnet verschilt van het dagnet

5. **Vereisten bij de gebruikersinterface**

   (a) Ingeven van vertrekplaats

   (b) Ingeven van bestemming

   (c) Tonen van verschillende opties

2.2 Use cases

2.2.1 Calculate routes

Use Case 1	**Calculate routes**
Primary actor:	Any user
Stakeholders:	/
Precondition:	/

Postcondition:

- Short details of the 3 quickest routes are shown (DR Information)
- Quickest route is visualised on the map
- Detailed info of the quickest route is displayed (DR Detailed information)

Basic flow:

1. User fills in the search form (DR Search form)
2. System validates the data
3. System calculates quickest routes
4. System shows information of quickest routes (DR Information)
5. System shows quickest route on the map
6. System shows detailed info of quickest route (DR Detailed information)

Alternative flow:

1A. User wants to pass a specific P&R
  1A1. System validates the information
  1A2. System calculates quickest routes from this P&R
  1A3. Go to step 4

2A. System detects incorrect information
  2A1. System alerts the user
  2A2. Go to step 1

2.2.2 Select route

Use Case 2	**Select route**
Primary actor:	Any user
Stakeholders:	/
Precondition:	- System has calculated 3 quickest routes - System shows 3 quickest routes in a list
Postcondition:	- Selected route is shown on map - Detailed info of selected route is shown (DR Detailed information)
Basic flow:	1. User selects a route from the list 2. System shows the route on the map 3. System shows detailed info of the selected route (DR Detailed information)
Alternative flow:	/

2.2.3 Postpone or expedite trip

Use Case 4	**Postpone or expedite trip**
Primary actor:	Any user
Stakeholders:	/

Precondition:

- System has calculated 3 quickest routes
- System shows 3 quickest routes in a list

Postcondition:

- Info of 3 new routes are shown (DR Information)
- First route is shown on map
- Detailed info of first route is shown (DR Detailed information)

Basic flow:

1. User changes departure or arrival time by 30 minutes
2. System calculates 3 new quickest routes
3. System shows information of 3 new routes (DR information)
4. System shows first route on the map
5. System shows detailed info of the first route (DR Detailed information)

Alternative flow: /

2.3 Domain rules

2.3.1 Search form

Search form consists of following data:

- Departure location
- Destination location
- Departure time
- Date of trip
- Selected P&R

2.3.2 Information

The information that will be shown:

- Departure time
- Arrival time
- Time per subroute
- P&R name
- Total time of trip

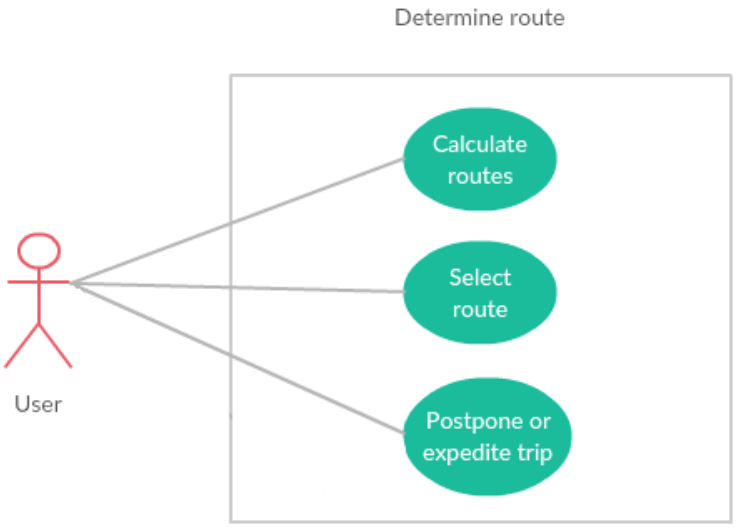
2.3.3 Detailed information

The information that will be shown:

- P&R name
- Amount of parking spots
- Ticketinfo
- Departure time
- Arrival time
- Time per subroute

- Total price
- Departure time per subroute
- Arrival time per subroute
- Delay time

2.4 Use case diagram



Figuur 2.1: De use cases worden allemaal door 1 actor uitgevoerd

Er is maar één actor in de applicatie, namelijk de gebruiker die naar de website surft. Te zien in figuur 2.1 kan deze gebruiker routes laten berekenen, een route selecteren voor meer gedetailleerde informatie te verkrijgen en de route een vast aantal minuten te laten vervroegen of te verlaten.

2.5 Feature list

Tabel 2.5 toont een lijst van alle features, geordend per sprint en elk met een complexiteit op basis van de story points die de bijhorende issues op GitHub kregen. Deze complexiteit kan volgende waarden aannemen, met hoe hoger de waarde, hoe moeilijker en hoe meer tijd een feature in beslag neemt: 0.5, 1, 2, 3, 5, 8, 13, 20 of 40.

Tabel 2.4: De features met de complexiteit

Issue nr.	Issue naam	Sprint	Complexiteit
1	API's leren kennen en documenteren	1	5
2	Werkomgeving opstellen (google drive, github, ...)	1	8
3	Use cases maken	1	8
4	Activity Diagram maken	1	8
5	Deployment diagram maken	1	8
6	Klassen diagram maken	1	8
7	UI mockups maken	1	8
8	Technologische analyse	1	8
9	Functionele specificaties	1	8
10	Programma opstellen met Maven	1	13
11	Inputvelden en andere UI opstellen	1	5
12	Inputvelden hebben autocomplete	1	2
13	Route via Google maps tonen	1	3
14	Route berekenen van start naar P&R	1	13
15	Route berekenen van P&R naar eindbestemming	1	13
16	Presentatie sprint 1 voorbereiden	1	8
17	Eerste deploy van server uitvoeren	1	8
18	klassen diagrammen implementeren	2	20
19	Nieuwe UI mockups	2	8
20	Nieuwe UI implementeren	2	13
21	Correcte route tonen	2	5
22	Korte info van routes tonen	2	13

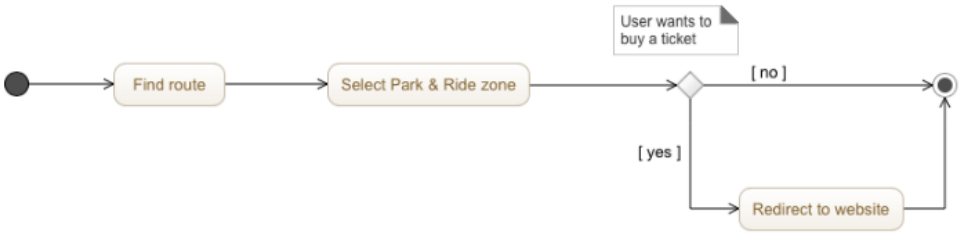
Tabel 2.4: De features met de complexiteit

Issue nr.	Issue naam	Sprint	Complexiteit
23	Dichtstbijzijnde P&R berekenen	2	20
24	Gedetailleerde informatie van route berekenen	2	20
25	Gedetailleerde informatie van route tonen	2	20
26	Unit testen schrijven	2	13
27	Integration testen schrijven	2	13
28	Use cases herwerken	2	8
29	De Lijn API laten werken	2	20
30	Bridge voor De Lijn API maken	2	8
31	API analyse	2	13
32	Documentatie in orde brengen	2	8
33	Bugs fixen	2	5
34	Overview UI	2	20
35	Presentatie voorbereiden	2	8
36	Totale tijd berekenen	3	3
37	Totale kost berekenen	3	3
38	Specifieke P&R selecteren	3	20
39	Korte info over route aangepast	3	5
40	Datum en tijd in berekeningen voor route opnemen	3	5
41	Verschillende subroutes een andere kleur geven op de map	3	8
42	Informatie in detailvenster van route aangepast	3	5
43	Usability testen	3	5
44	Wachttijden tonen tussen subroutes	3	5
45	Optie voor departure of arrival time te veranderen met 30 min	3	20
46	Bugs fixen	3	5
47	Foute input voor uur en datum opvangen	3	5

Hoofdstuk 3

Systeemarchitectuur

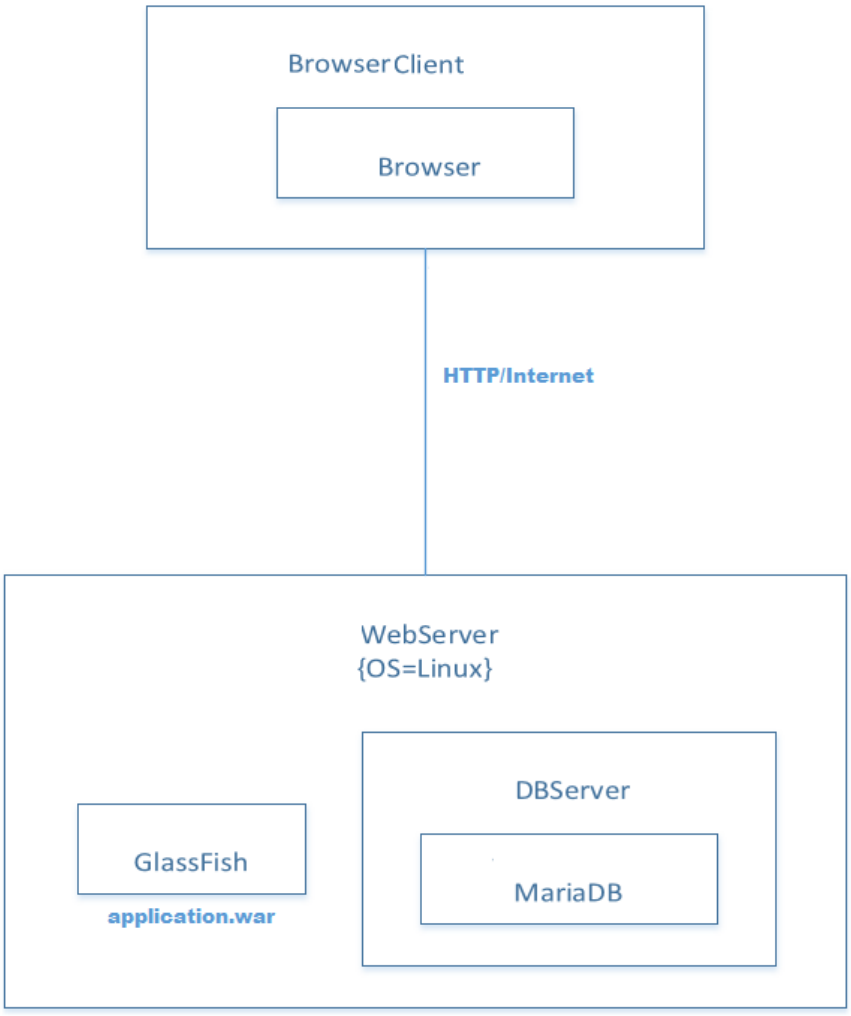
3.1 Activity diagram



Figuur 3.1: Het primaire verloop van de applicatie

Figuur 3.1 toont het activity diagram van de applicatie. De gebruiker zal eerst de mogelijke routes opvragen aan de hand van de ingevoerde gegevens. Het systeem berekent dan de optimale route en geeft die weer. Als de gebruiker niet tevreden is met de voorgestelde route kan hij één van de andere routes selecteren, of hij kan zijn zoekvoorwaarden aanpassen en een nieuwe set van routes laten berekenen. Wanneer een route geselecteerd is, krijgt de gebruiker de optie om doorverwezen te worden naar de site van De Lijn om meteen een ticket aan te schaffen.

3.2 Deployment diagram



Figuur 3.2: De verschillende componenten van de applicatie

De applicatie bestaat uit een eenvoudige browser- en webservercommunicatie, te zien in figuur 3.2. Voor het servergedeelte wordt er gebruik gemaakt van een Linux server waarop GlassFish draait in combinatie met MariaDB als relationele databank. Het cliëntgedeelte wordt volledig door de browser afgehandeld.

3.3 Class diagram

Het klassendiagram in figuur 3.3 beschrijft de interne structuur van de applicatie. Het beschrijft de onderlinge relaties tussen klassen en toont hun attributen en operaties.

We beginnen met de MainApp klasse. Hierin wordt de locatie van de stad waarvoor de app geschreven is bijgehouden. Hierdoor is de applicatie gemakkelijk herbruikbaar in andere steden. Om dezelfde reden heb je hier reeds een lijst met alle P&R's. Als laatste staan hier ook alle routes in, die het systeem heeft gegenereerd.

Een route bestaat uit meerdere attributen waarvan onder andere twee belangrijke. Een eerste attribuut is de P&R waar de route gebruik van maakt en een tweede is een lijst van subroutes (vertrekkende vanuit de P&R). De combinatie van een P&R en subroutes uit de lijst vormt een volledige route.

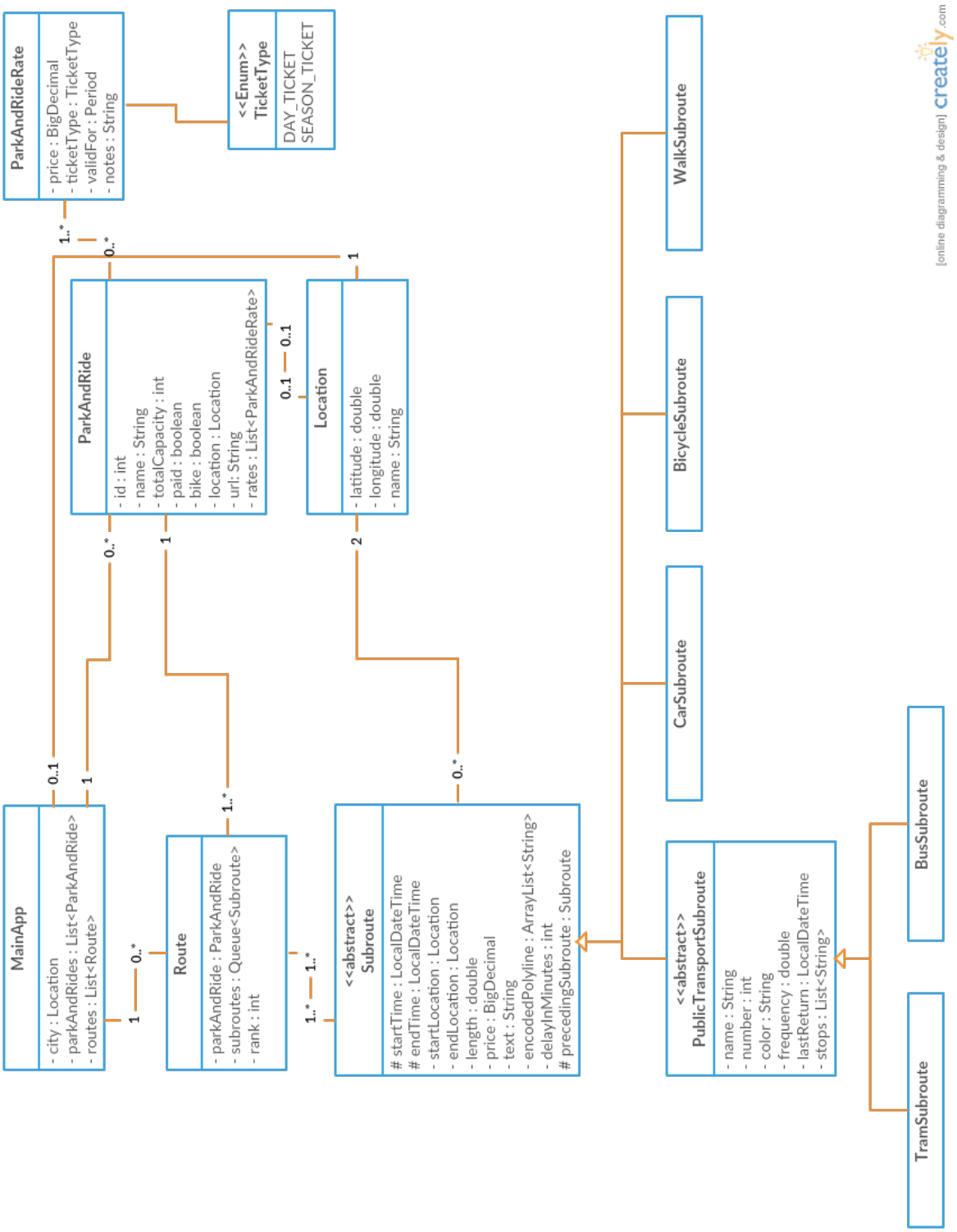
Een subroute is dus deel van een route, hiervan worden bepaalde gegevens bijgehouden: onder andere de start- en eindlocaties en start- en eindtijden. Een subroute heeft ook een bepaalde transportmethode zoals het openbaar vervoer of met de auto. Omdat sommige transportmethodes extra gegevens hebben is de subroute een abstracte klasse en kunnen de gegevens die elke transportmethode nodig heeft gemakkelijk overgeërfd worden.

Eén van de transportmethodes is de PublicTransportSubroute, deze kan ofwel een bus, ofwel een tram zijn en heeft nog extra gegevens nodig. Deze gegevens zijn onder andere de naam, het nummer van de bus of tram, hun frequency om te weten om de hoeveel tijd er een bus of tram rijdt op deze route, een lastReturn om te weten om hoe laat de laatste bus of tram rijdt en een lijst van alle tussenstops die de bus of tram maakt.

De klasse ParkAndRide houdt alle benodigde gegevens bij waarover een gebruiker moet kunnen beschikken. Dit zijn onder andere: de naam, de prijs, de totale capaciteit, het id en de locatie.

Een Locatie heeft een latitude en longitude. Deze worden bijgehouden om de exacte locatie te bepalen. Dit is nodig bij het berekenen van een route of bij het tonen van de locatie op de kaart. Er wordt ook een string met de naam van de locatie bijgehouden.

De laatste klasse die we bespreken is de ParkAndRideRate klasse. deze houd de ticketinformatie bij. ParkAndRideRate kent dus de prijs, het soort ticket, hoelang het ticket geldig is en eventueel extra opmerkingen die bij dit ticket horen.



Figuur 3.3: De nodige klassen van de applicatie

Hoofdstuk 4

Testplan

Er zijn 3 soorten testen beschikbaar: unittesten, integration testen en usability testen. Unit testen worden gebruikt om aparte klassen en modules te testen, terwijl integration testen net nagaan of die aparte modules ook kunnen samenwerken. Usability testen controleren de gebruiksvriendelijkheid van een applicatie.

4.1 Unit testen

Tabel 4.1: Een overzicht van de unit testen

Klasse	Test
Location	testLatitudeTooSmall, testLatitudeTooBig, testcorrectLatitude, testLatitudeMax, testLatitudeMin, testLongitudeTooSmall, testLongitudeTooBig, testcorrectLongitude, testLongitudeMax, testLongitudeMin, testNameNotNull, testFormattedCoordinates
ParkAndRide	testIdNotNegative, testIdNotZero, testCorrectId, testNameNotNull, testNameNotEmpty, testCorrectName, testUrlNull, testUrlEmpty, testCorrectUrl, testCapacityNotNegative, testCapacityZero, testCorrectCapacity, testLocationNotNull, testCorrectLocation, testRatesWhenNotPaid, testRatesNull, testRatesEmpty, testRatesCorrect, testPaidCorrectTrue, testPaidCorrectFalse, testBikeCorrectTrue, testBikeCorrectFalse

Tabel 4.1: Een overzicht van de unit testen

Klasse	Test
ParkAndRideRate	testPriceZero, testPriceNull, testPriceCorrect, testTicketTypeNull, testTicketTypeCorrect, testValidForNull, testValidForNegative, testValidForCorrect, testNotesNull, testNotesEmpty, testNotesCorrect
Route	testRouteGetPriceReturnsTheTotalPriceOfTheSubroutes, testGetTripDuration, testGetTotalPublicTransportTime, testGetWaitingTime, testGetPublicTransportCost
SubRoute	testStartTimeNotBeforeToday, testStartTimeNotNull, testCorrectStartTime, testStartTimeEarlierToday, testEndTimeNotBeforeToday, testEndTimeNotNull, testEndTimeNotBeforeBeginTime, testCorrectEndTime, testEndTimeEarlierToday, testStartLocationNotNull, testStartLocationCorrect, testEndLocationNotNull, testEndLocationDifferentFromStartLocation, testEndLocationCorrect, testLenghtNotNegative, testLengthNotZero, testLengthCorrect, testPriceNotNegative, testPriceCorrect, testPriceZero
PublicTransportSubroute	testNameNotNull, testNameEmpty, testNameCorrect, testNumberNotEmpty, testNumberCorrect, testColorNotEmpty, testColorCorrect, testFrequencyNotNegative, testFrequencyNotZero, testFrequencyCorrect, testLastReturnNotNull, testLastReturnNotBeforeToday, testLastReturnCorrect, testLastReturnCorrect, testStopsNotEmpty, testStopsCorrect

We beginnen met de klasse Locatie. Een locatie heeft om te beginnen een latitude en longitude. Hier wordt getest of alles correct verloopt wanneer een te groot getal, te klein getal, correct getal of een getal met maximale of minimale grootte wordt ingevoerd. Een locatie heeft ook een naam. Hiervoor wordt getest wat er gebeurt als er niets wordt ingegeven (waarde null).

De volgende reeks testen zijn die voor de klasse ParkAndRide. Als eerste wordt hier de id en de totalCapacity getest. Hierbij wordt er gecheckt dat ze niet 0 of negatief mogen zijn en of het programma correct werkt met juiste getallen. Bij de naam wordt er getest op het niet invullen (waarde null), het invoegen van de lege string en het correct ingeven van een naam. Bij het testen van de

location wordt het verloop getest met een correcte waarde en of de juiste fout wordt getoond wanneer niets wordt ingegeven. Ook wordt gecontroleerd dat het paid attribuut niet false mag zijn als er rates beschikbaar zijn. Als laatste worden de rates zelf getest. Opnieuw wordt er gecontroleerd op het ingeven van een correcte waarde, een null waarde en een lege tabel.

Net zoals in de vorige klassen, worden in de klasse van ParkAndRideRate alle attributen getest met een correcte waarde en met de null waarde. Nadien wordt er ook nog getest dat de prijs niet 0 mag zijn, dat de periode van het attribuut validFor niet negatief is en dat de notes niet de de lege string mogen bevatten.

Voor de klasse Route wordt er getest of de methode getPrice de correcte informatie teruggeeft bij geldig ingevoerde data. Ook wordt de werking van de logica achter het berekenen van de reistijden hier gecontroleerd.

De klasse Subroute heeft geen methodes. Hier worden opnieuw enkel attributen getest op correcte waardes. StartTime en endTime worden ook nog getest op niet null zijn. De datum mag niet in het verleden liggen en het uur mag nog niet langer dan vijf minuten verstreken zijn. De waarde van endTime is ook ongeldig indien het voor startTime valt. De startLocation en endLocation worden ook getest op niet null zijn en er wordt getest of ze verschillend zijn van elkaar. De price en length attributen worden getest op negatief zijn en length mag niet 0 zijn. Als laatste wordt er getest of price wel kan werken met waarde 0.

Als laatste wordt de klasse PublicTransportSubroute getest. Opnieuw worden alle attributen getest op een correcte waarde. Nadien wordt getest dat de name niet null of de lege string mag zijn, number en frequency mogen niet negatief of 0 zijn, lastReturn mag niet negatief zijn of voor de huidige datum liggen en stops mag geen lege lijst of null bevatten.

4.2 Integration testen

Tabel 4.2: Een overzicht van de integration testen

Pagina	Test
Home	WrongInputAddsErrorClass, NoInputAddsErrorClass, SearchRouteWorks, usingHourInputWorksViaUrl, usingHourInputWorks, wrongHourInputShowsError, selectSpecificPRWorks, usingPastDateInUrlShowsError, missingParametersInUrlShowsError
Overview	overviewShowsListOfPr

Er zijn een aantal integrationtesten die automatisch kunnen uitgevoerd worden. Eerst wordt uitgelegd wat deze testen precies doen en nadien wordt beschreven hoe men deze testen kan laten runnen.

De testen vertrekken van op de home- of overviewpaginas van de website. Op de homepagina wordt voor verschillende scenario's gecontroleerd of de 'Error' klasse wordt toegevoegd aan de inputbalk. Een voorbeeld van een scenario is wanneer een verkeerd adres wordt ingegeven in de inputbalk voor de start- of eindlocatie. Ook wordt getest of alles goed verloopt bij het ingeven van correcte adressen in de inputbalk voor de start- en eindlocatie. Het invullen van het tijdstip wordt zowel via de url als via de grafische component getest. Als laatste wordt ook nagegaan of alles correct verloopt wanneer de gebruiker een P&R selecteert bij de geavanceerde opties. Voor de overviewpagina wordt gecontroleerd of de lijst met P&R's correct wordt geïnitialiseerd.

De integration testen gebeuren met Selenide, wat gebruik maakt van de chrome webdriver. Deze webdriver moet in het pad aanwezig zijn voordat de testen kunnen draaien. Op een OSX systeem (waarop Homebrew geïnstalleerd is) kan dit zeer gemakkelijk door het commando "brew install chromedriver" uit te voeren in de console. Windows gebruikers kunnen de laatste release downloaden via de chromedriver website[1]. Hierna moet het exe-bestand aan het pad worden toegevoegd.

[1] https://sites.google.com/a/chromium.org/chromedriver/getting-started

4.3 Usability testen

In sprint drie werd een kleinschalig onderzoek gedaan naar de usability en user experience van de webapplicatie. Een groep van 10 personen, bestaande uit zowel mannen als vrouwen, met een leeftijd variërend tussen 18 en 59 jaar experimenteerden met de finale versie van de webapplicatie. Elke persoon kreeg dezelfde opdracht voorgeschoteld waarbij twee scenario's zonder externe hulp moesten uitgevoerd worden. Tijdens de opdracht werden de personen wel gemonitord door één van de projectleden. Nadien werd ook gevraagd om feedback te geven op de webapplicatie. Een voorbeeldexemplaar van de opdracht samen met het feedbackformulier zijn bijgesloten in bijlage A.

De algemene reacties van de testgebruikers waren alvast zeer positief. Sommige personen vinden de benodigde informatie in een oogopslag, anderen hebben wat meer tijd nodig. Alle personen vinden echter de gevraagde informatie zonder hulp en leren het gebruik van de interface snel kennen. De testgebruikers geven de webapplicatie een score van 4/5 voor design, 3.8/5 voor gebruiksvriendelijkheid en 4.2/5 voor inhoud.

Zelf merkten de projectleden op dat de gebruikers initieel wel eens op een verkeerde knop durfden klikken, maar met behulp van de voorziene foutboodschappen vond men snel de juiste weg. Eens men de correcte manier van werken had ontdekt, verliep alles zeer vlot. Er werd ook meermaals opgemerkt dat de gebruikers soms niet meteen doorhadden dat zij naar onder konden scrollen op de pagina, dit zou beter in beeld moeten gebracht worden. Opvallend was ook dat enkele gebruikers sneller gebruik maakten van de externe links om informatie op te zoeken in plaats van op de overzichtspagina te blijven.

Enkele suggesties die de gebruikers hadden:

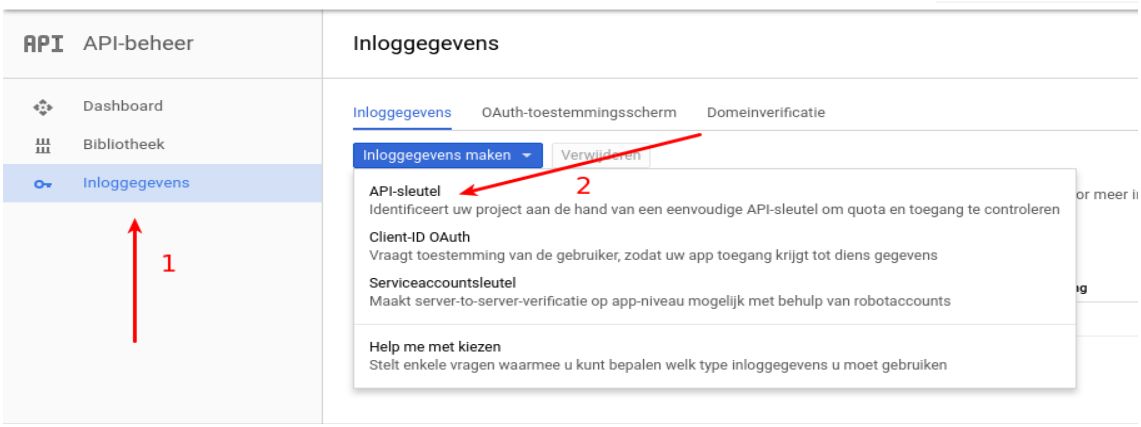
- Het zou handig zijn moest er ook informatie over andere parkeermogelijkheden beschikbaar zijn op deze website.
- De geselecteerde P&R op de overzichtspagina mag wat meer in de verf gezet worden op het kaartje.
- De tarieven springen niet meteen in het oog.
- Het perron vermelden bij de gedetailleerde reisweg is een leuke extra.
- De optie om een specifieke P&R te selecteren zit wat verstopt.
- Het adres (straatnaam) van de P&R zou ook kunnen vermeld worden bij in het infovenster.

Hoofdstuk 5

Installatiehandleiding

5.1 Configuratie

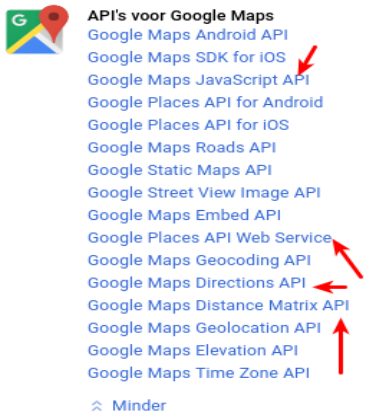
Om de applicatie correct te laten draaien moeten een aantal stappen gevolgd worden. Als eerste is een Google API key vereist. Dit is een unieke code die via de Google Developers Console[1] kan aangevraagd worden. Om de code aan te vragen dien je aan de linkerkant op inloggegevens te klikken, waarna je op de knop inloggegevens aanmaken en API-sleutel klikt.



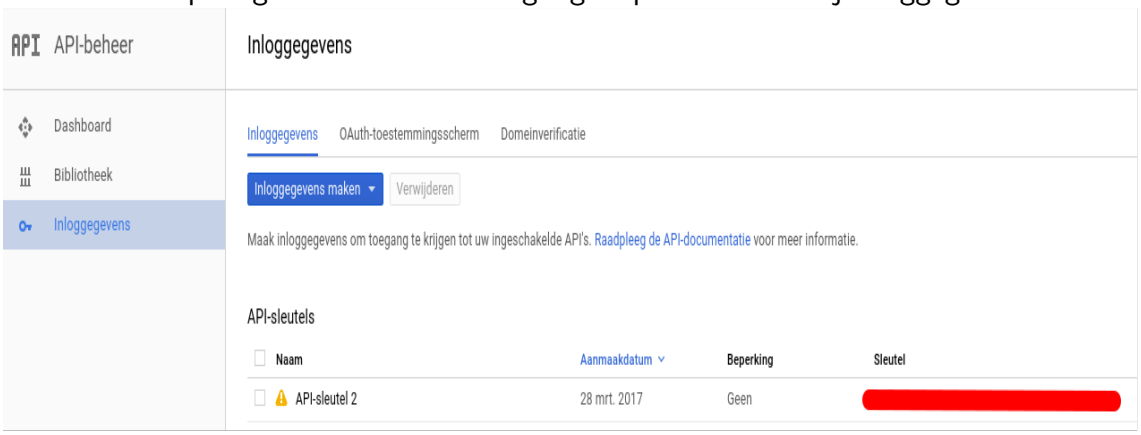
Vervolgens zal via de Google Developers Console de Directions API, Distance Matrix API, JavaScript API en Google Places API Web Service geactiveerd moeten worden. Dit kan door op het dashboard op "API Inschakelen" te klikken

[1] https://developers.google.com/maps/documentation/javascript/get-api-key

en de juiste te selecteren uit de lijst, onder "API's voor Google Maps", en op "Inschakelen" te klikken.



Eenmaal een Google code verkregen is, moet een aanpassing gebeuren in de map "ParkAndRideGent/src/main/resources" van het project. Hier zal een bestand beschikbaar zijn, namelijk "config.example.properties". Dit bestand moet gedupliceerd worden met de naam "config.properties". Nadien kan dit nieuwe bestand met een tekstverwerker zoals Kladblok aangepast worden. Hier zal de tekst "GMAPS.KEY=" staan. Hierachter moet de unieke Google key geplaatst worden, zodat het volgende verkregen wordt: "GMAPS.KEY=code". De code bevindt zich op volgende locatie in het google api dashboard bij inloggegevens:



5.2 Build project

Het builden wordt gedaan met Maven. Er zal een bestand in de map "path/to/pr-01/ParkAndRideGent/ target/ParkAndRideGent-1.0-SNAPSHOT.war" verkregen worden. Dit bestand moet dan geüpload worden naar de web server. Voor instructies hoe dit precies moet, moet gekeken worden in de documentatie van uw server. Een aantal veel voorkomende servers zijn Glassfish[2], Tomcat[3] en Jetty[4].

5.2.1 Maven

Om het project te builden met Maven[5], dien je met een commandovenster naar volgende locatie gaan: "path/to/pr-01/ParkAndRideGent". Hierna voer je volgend commando uit: "mvn package -Dmaven.test.skip=true". Dit commando zal gebruikmaken van het pom.xml buildscript. In dit script kan ook geconfigureerd worden waar het .war bestand geplaatst wordt na het builden.

5.3 Deploy project

Om het project effectief te kunnen gebruiken dien je de .war-file die je verkregen hebt vanuit het builden eerst te uploaden naar een webserver. Instructies om dit te doen hangen af van server tot server.

[2]https://docs.oracle.com/cd/E19798-01/821-1757/6nmni99aj/index.html
[3]https://tomcat.apache.org/tomcat-6.0-doc/deployer-howto.html
[4]http://www.eclipse.org/jetty/documentation/current/configuring-deployment.html
[5]https://maven.apache.org/download.cgi

Hoofdstuk 6

Conclusie

Achteraf gezien hebben sommige delen meer tijd in beslag genomen dan we eerst hadden gepland. Vooral het project opstarten in de eerste sprint met een server in Java was voor velen een onderdeel dat ver in het achterhoofd verdwenen was. Het duurde langer dan verwacht om dit op te frissen en dat vertraagde de ontwikkeling van de applicatie. Daarnaast werden een aantal taken slecht ingeschat, waardoor ook daar meer tijd gespendeerd werd dan eerst verwacht. Door deze verkeerde tijdsinschattingen moesten enkele taken meegenomen worden naar de tweede sprint.

In sprint 2 ging alles al een heel stuk vlotter. Naast het inspelen op de gekregen feedback, waren de meeste geplande features voor deze tweede sprint dan ook volledig afgerond. De enkele taken die niet volledig gelukt zijn, hingen vooral af van een bepaalde API, die we niet gemakkelijk konden gebruiken. Uiteindelijk werd deze vervangen door een andere API, die wat minder goed is, maar toch een gelijkaardig resultaat gaf.

De derde en laatste sprint werd gebruikt voor het afwerken van de laatste issues. Hierbij waren onder andere kleine UI fixes, bugfixes en het verbeteren van het route-berekenings-algoritme. Hier en daar zijn er echter kleine features, zoals de keuze van het transportmiddel, die niet in het eindproduct geraakt zijn. Desondanks hebben we toch een mooie werkende demo beschikbaar en kan er op het einde van deze projectperiode een goede conclusie getrokken worden.

De opstart van een project is nooit eenvoudig, zo leert de ervaring. Alle groepsleden dienen op dezelfde lijn te zitten en alle aspecten van de applicatie moeten besproken worden alvorens er aan concrete use cases of andere analysedocumenten gewerkt kan worden. Taken werden in het begin niet altijd goed ingeschat, waardoor nieuw werk zich opstapelde en enkele issues moesten meegenomen wor-

den naar een volgende sprint. Eens deze problemen echter weggewerkt waren, verliep alles vlot. Uiteindelijk zijn we tevreden met het eindresultaat. Het was een leerrijke ervaring en een spannende uitdaging om voor de stad Gent een opdracht uit te werken.

Bijlage A

Blanco usability test

◯ man ◯ vrouw _____ jaar

Opdracht

Situatie 1: U wenst morgen Gent te bezoeken. U gaat met de auto en maakt gebruik van de Park & Ride zones. Via de webapplicatie zoekt u op hoe u van uw woonplaats in het stadscentrum van Gent geraakt.

Vragen:

1. Van welke Park & Ride zou u in deze situatie gebruik maken? _____
2. Wat is de totale reistijd en kostprijs van deze reis? _____
3. Indien u gebruik maakt van De Lijn: welke bus/tram moet u nemen? _____
4. Heeft u bovenstaande informatie gemakkelijk gevonden? _____

Situatie 2: U wenst meer informatie over Park & Ride Galveston.

Vragen:

1. Waar is P+R Galveston ongeveer gesitueerd? _____
2. Over hoeveel plaatsen beschikt P+R Galveston? _____
3. Is P+R Galveston betalend? Zoja, wat zijn de tarieven? _____
4. Heeft u bovenstaande informatie gemakkelijk gevonden? _____

Feedback

Gelieve de webapplicatie een score te geven op volgende aspecten:

(schaal: 1 = slecht, 2 = onvoldoende, 3 = voldoende, 4 = goed, 5 = uitstekend)

Design

1 2 3 4 5

Gebruiksvriendelijkheid

1 2 3 4 5

Inhoud

1 2 3 4 5

Indien u bij het gebruik van de webapplicatie op fouten bent gestoten, gelieve hieronder de fout te beschrijven:

Hieronder kan u algemene opmerkingen kwijt: