長崎バス情報サービス株式会社

SCRIPTIE

OV-routeplanner voor Nagasaki Bus

Datum:	15-03-2016
Naam:	Luuk Nass
Studentnummer:	1574695
Opleiding:	HBO Informatica (Software Engineering)
Eerste examinator:	Peter van Rooijen
Bedrijfsnaam:	Nagasaki Bus Information Service
Plaats:	Nagasaki, Japan
Versie:	1.0

MANAGEMENTSAMENVATTING

Dit document beschrijft het verloop van het afstudeerproject van Luuk Nass, uitgevoerd bij Nagasaki Bus Information Service (NBIS) in Nagasaki, Japan.

NBIS is het bedrijf dat de IT-zaken van Nagasaki Bus regelt, het busbedrijf van de stad Nagasaki, Japan. Nagasaki Bus maakte ten tijden van, en voor het afstudeerproject gebruik van een functioneel verouderde routeplanner, met als grootste tekortkoming het gebrek aan ondersteuning voor overstappen. De planner gaf alleen resultaten wanneer het vertrekpunt en de bestemming door een en dezelfde bus wordt aangedaan. Er zijn ook andere functionaliteiten waarop men graag verbetering wenste, zoals het kunnen zoeken naar niet alleen bushaltes, maar ook landmarks.

Na een verdieping in de materie en een analyse van wat er zoal beschikbaar is, en hoe OV-data in het algemeen zoal verwerkt wordt is er om een nieuwe routeplanner te implementeren gekozen om in eerste instantie de data van de dienstregelingen naar het gestandaardiseerde GTFS-formaat (General Transit Feed Specification) te converteren, en op basis van dit formaat een routeplanner te implementeren. Vervolgens is na een grondige vergelijking naar de mogelijkheden voor routeplanners uiteindelijk een geheel nieuwe routeplanner ontwikkeld. Als basis is hierbij gebruik gemaakt van de routing engine van de open source routeplanner OpenTripPlanner (OTP). Deze heeft niet alleen ondersteuning voor overstappen, maar ook voor onder andere looproutes in combinatie met OV. Als extra functionaliteit zijn er ook nog een aantal zaken toegevoegd, zoals het visueel weergeven van zoekresultaten en busroutes op kaarten, dichtstbijzijnde haltes via GPS kunnen selecteren, alsook het kunnen reizen van en naar landmarks.

Tijdens de ontwikkeling bleek dat het GTFS-formaat echter niet in alle gevallen compleet compatibel is met de eisen van Nagasaki Bus. Zo is het bij veel busbedrijven in Japan niet gebruikelijk dat een buslijn een lijnnummer heeft, maar in plaats hiervan een lijst via-bestemmingen heeft. Ook is de manier van tariefberekenen uniek voor Nagasaki Bus. Daarom is daarop als aanvulling een extra softwarelaag geïmplementeerd die aan deze eisen kan voldoen terwijl de basis, GTFS, ongewijzigd is.

De nieuw ontwikkelde routeplanner is goed ontvangen. Het uiteindelijke eindproduct heeft een grondige acceptatietest ondergaan en is daarmee goedgekeurd voor productie. Op moment van schrijven, eind februari 2016, is de applicatie intern online voor medewerkers van Nagasaki Bus, en is de planning om binnen enkele weken ook klanten uit te nodigen. Vervolgens is de planning om binnen afzienbare tijd de routeplanner geheel in productie te nemen.

INHOUDSOPGAVE

1. INLEIDING

Dit document is een afstudeerscriptie. De afstudeeropdracht en -scriptie dient voor de opleiding Informatica, afstudeerrichting Software Engineering en is bestemd voor zowel de afstudeerder, het afstudeerbedrijf en de hogeschool. De besproken onderdelen zijn als volgt.

In hoofdstuk 2 is er allereerst een inleiding tot het bedrijf gevolgd door en een beschrijving van de aanleiding, ofwel de kwestie, in hoofdstuk 3. In hoofdstuk 4 wordt de context van het afstudeerproject besproken.

Daarna volgt een theoretisch kader in hoofdstuk 5. Daarna de probleemstelling en daartoe behorende onderzoeksvragen en -methodieken in hoofdstuk 6. Vervolgens wordt in hoofdstuk 7 per onderzoeksvraag antwoord gegeven, en aan de hand van deze onderzoeksresultaten het eindproduct en de ontwikkeling daarvan in fasen gepresenteerd in hoofdstuk 8. Als laatste onderdeel worden discussies en aanbevelingen behandeld, in de vorm van een evaluatie van het project.

In hoofdstuk 10 worden bronvermeldingen en de bibliografie beschreven, en in hoofdstuk 11 volgen de bijlagen, onder andere het oorspronkelijke plan van aanpak en de student-evaluatie van de afstudeerperiode.

2. ORGANISATIE

Nagasaki Bus Information Service (NBIS) is onderdeel van Nagasaki Bus Group, de overkoepelende organisatie van een aantal bedrijven met als belangrijkste lid Nagasaki Bus (opgericht in 1936, het busbedrijf van de stad Nagasaki). Voorbeelden van andere bedrijven binnen de overkoepelende organisatie zijn:

- Winkelcentrum en busstation Mirai Nagasaki Cocowalk
- Hotels als Nikko Huis Ten Bosch en een viertal andere hotels
- De busbedrijven Saikaikotsu, het busbedrijf van de 30 km noordelijk geleden stad Saikai en Goto Bus, het busbedrijf van de Goto-eilanden
- De touroperatator Nagasaki Bus Kanko
- Nagasaki Bus Motor, een leasing-, verzekering- en reparatiebedrijf voor auto's

長崎バスグループ

長崎バス情報サービス株式会社

Nagasaki Bus Information Service (NBIS) is begin 2013 officieel afgesplitst van Nagasaki Bus als apart bedrijf door recente ontwikkelingen op ICT-gebied, maar bevindt zich nog wel in hetzelfde gebouw (NBIS, 2016). NBIS houdt zich voornamelijk bezig met het ontwikkelen, verkopen, onderhouden en introduceren van zowel software als hardware, consultancy en data-analyse en neemt steeds meer nieuwe projecten aan. Voorbeelden hiervan zijn de recent geïntroduceerde Nagasaki Smart Card; een “OV- chipkaart” voor de regio Nagasaki. Andere voorbeelden zijn beeldkranten voor advertentiedoeleinden en hardware- oplossingen als automatische omroepsystemen voor in de bus. Daarnaast bijvoorbeeld ook digitale vertrekstaten op busstations.

NBIS heeft, hoewel afgesplitst, nog altijd een directe relatie met Nagasaki Bus. Nagasaki Bus heeft 994 medewerkers met een aantal verschillende directeuren op verschillende niveaus (Nagasaki Bus, 2016). Binnen NBIS zijn er momenteel tien medewerkers; één algemeen directeur en een twee afdelingshoofden ICT. Verder zijn er software-ontwikkelaars, systeembeheerders en de overige medewerkers zijn voor kantoorwerkzaamheden zoals verkoop en telefoonsupport. De voertaal is Japans.

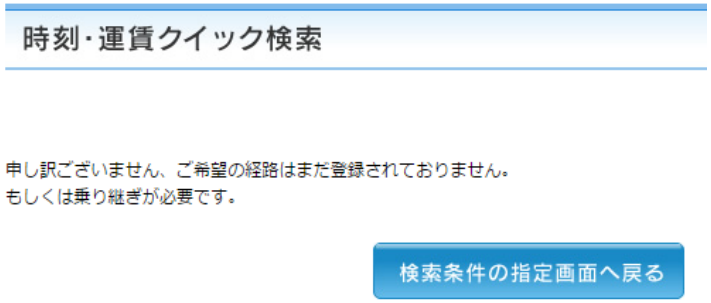
3. KWESTIE

De online routeplanner van Nagasaki Bus is verouderd en biedt weinig moderne functionaliteit (Nagasaki Bus, 2016). Het is oorspronkelijk ontwikkeld in 2002 en eenmalig herzien in 2007. Verder zijn er enkel her en der wat onderhoudsbeurten gedaan. Het grootste probleem is dat het geen zoekresultaten kan tonen voor zoekopdrachten waarbij de bestemming niet met 1 bus bereikt kan worden; dus wanneer er een overstap nodig is. Deze routeplanner alleen kan zoeken binnen busroutes (een reeks bushaltes in een bepaalde volgorde die door dezelfde bus worden aangedaan). Enkel in het geval de opstaphalte en de bestemming door dezelfde bus wordt aangedaan verschijnen er zoekresultaten. In de figuur hieronder wordt een dergelijke zoekopdracht gedaan.

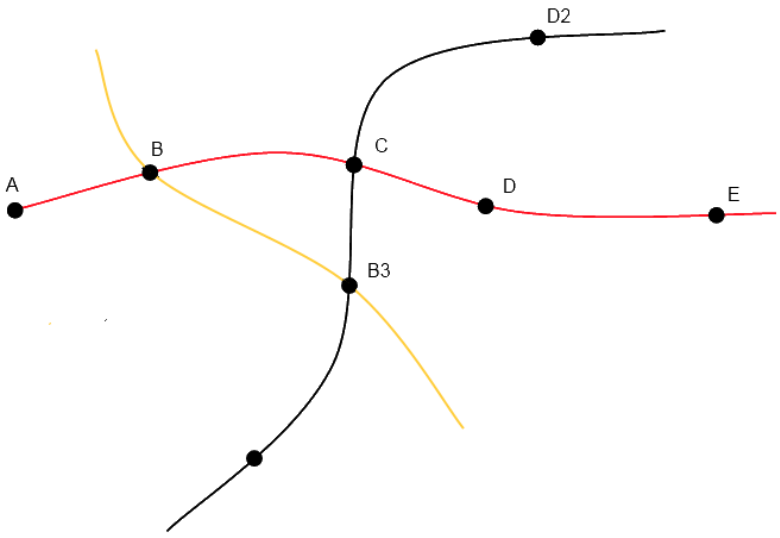
Figuur 1 Een voorbeeldroute van Wakabamachi (若葉町) naar Nagasaki-shinchi-terminal (長崎新地ターミナル)

Wanneer er gezocht wordt vanaf een bushalte die niet direct verbonden is met de eindbestemming verschijnt de volgende pagina:

Figuur 2: “Onze excuses, de door u gewenste reis is nog niet geregistreerd, ofwel mogelijk is er een overstap vereist.”



Routes zijn niet aan elkaar gekoppeld en men kan daarom geen informatie opvragen over diverse overstapmogelijkheden. Het schematisch overzicht hieronder laat dit zien.



Figuur 3: Schematisch voorbeeld

Een route van bushalte A naar bushalte E is geen probleem, maar van A naar B3 wel; dan is er geen beschikbare route. Een blik op het schema wijst uit dat er twee mogelijke routes zijn: (1) A > B > B3 en (2) A > C > B3. Welke route de meest optimale is kan bijvoorbeeld per tijdstip of per dag verschillen omdat de gele buslijn veel minder frequent kan rijden dan de rode en zwarte buslijn. Het koppelen van busroutes is dus niet enkel een kwestie van knooppunten creëren. Er zijn nog een aantal andere zaken die ook meegenomen moeten worden. Het is bijvoorbeeld mogelijk dat het lopen naar een andere bushalte waar meer of beter aansluitende bussen komen de algehele reiservaring kan verbeteren. Zo zou daardoor een extra overstap voorkomen kan worden, of kan de totale reistijd daarmee verkort worden.

4. CONTEXT EN PROBLEEMSTELLING

Er moet een oplossing worden gevonden om busroutes efficiënt aan elkaar te koppelen zodat ook reisinformatie kan worden gegeven over routes die een overstap vereisen zijn. Het doel van deze opdracht is om de online routeplanner te moderniseren, waarbij het mogelijk maken van overstappen binnen een zoekopdracht het belangrijkste onderdeel is.

Hiernaast zijn er nog een aantal andere wensen. Voorbeelden hiervan zijn het niet alleen kunnen zoeken van bushalte naar bushalte, maar ook vanaf bepaalde landmarks, dat wil zeggen bepaalde locaties waar men veel heen reist zoals een treinstation, toeristische attractie of een ziekenhuis. Daarnaast is er de wens om een route niet alleen van overstap naar overstap op tekstbasis weer te geven. Op een online kaartensysteem als Google Maps zou dan de gehele route, of tenminste de opstap- en eindhalte weergegeven kunnen worden. Een andere wens is het weergeven van de prijs van de nieuwe gecombineerde route en als laatste is er nog de wens om de webapplicatie ook compatibel te maken met mobiele apparaten zoals smartphones. Hierbij kan naast het optimaliseren van de user inferface ook gedacht worden aan het gebruik van zaken als automatische plaatsbepaling en vanuit die ingang zoekopdrachten uit te voeren.

Hieronder een tabel met MoSCoW-prioriteiten.

#	Naam	Prioriteit
1	Mogelijkheid tot overstappen	Must Have
2	Zoeken van en/of naar landmarks	Could Have
3	Visuele weergave van (een deel van) de zoekresultaten	Could Have
4	Prijsbereking van een route met overstap	Could Have
5	Website met ondersteuning mobiele apparaten	Could Have

Van deze onderdelen is alleen het introduceren van de mogelijkheid tot het vinden van routes met een mogelijkheid tot overstappen een requirement, de rest zijn nice to haves.

De roep om een handigere, gebruiksvriendelijkere routeplanner begint steeds dringender te worden. Nagasaki Bus Information Service wil als uiteindelijke doelstelling deze kans aangrijpen om klanten beter van dienst te zijn met een meer geavanceerde routeplanner die beter aansluit op wat klanten tegenwoordig verwachten. Tevens is dit een kans om intern vernieuwingen door te voeren. De opdracht is dus om bestaande software te verbeteren door nieuwe software of functionaliteit te introduceren om deze te vervangen of te verbeteren.

5. THEORETISCH KADER

Allereerst een inleiding tot de materie, routeplanners. De meest bekende routeplanner is wellicht Google Maps, maar ook routeplanners als die van de Nederlandse Spoorwegen en 9292.nl zijn bekende voorbeelden. Al deze routeplanners maken gebruik van het General Transit

Feed Specification (GTFS, oorspronkelijk Google Transit Feed Specification), een standaard op het gebied van het opslaan van informatie van OV-systemen (Google 2016). GTFS is een collectie .csv bestanden waarin volgens een vaste standaard een OV-systeem beschreven wordt. Het grote voordeel hiervan is dat bestaande routeplanners zoals Google Maps, maar ook open source routeplanners als OpenTripPlanner (OpenTripPlanner 2016) deze data in kunnen lezen en direct compatibel zijn. Steeds meer GTFS-feeds worden online aangeboden, zoals bij GTFS data exchange (GTFS Data Exchange, 2016) of bij Google Code (Google, 2016). Ook de meeste Nederlandse OV-bedrijven gebruiken GTFS (OV-api, 2016). Dit zijn uiteraard slechts de openbaar beschikbare feeds, het vermoeden bestaat dat GTFS ook intern veel gebruikt wordt.

Wat is GTFS?

In bijlage D staat een schematische weergave van het GTFS-formaat (Greenhorne & O'Mara 016). In eerste instantie lijkt het relationeel database modal (RDBM) erg ingewikkeld, maar in de basis bestaat GTFS uit slechts zes entiteiten.

(1) Agency: een lijst uitvoerders (naam, website, telefoonnummer)

(2) Stops: een lijst haltes (naam, locatie, beschrijving)

(3) Routes: een lijst routenummers (uitvoerder, lijnnummer, soort voertuig)

(4) Calendars: schema's voor beschikbaarheid

(5) Trips: een lijst ritten (routenummer, beschikbaarheidsschema, eindbestemming)

(6) Stop Times: een specificatie per rit (trip) waar er hoe laat gestopt wordt

Daarnaast heeft GTFS ook een behoorlijk aantal optionele velden, zowel in de bovenstaande zes entiteiten als in geheel optionele entiteiten. Zo kan bijvoorbeeld ook per voertuig worden aangegeven of men fietsen kan vervoeren, of dat een halte bijvoorbeeld wel of niet toegankelijk is voor rolstoelen.

Deze optionele entiteiten zijn:
Calendar Dates: Uitzonderingen op de dienstregeling (maandag als op zondag bijvoobeeld)
Fare Attributes en **Fare Rules**: Het specificeren van de prijs op een aantal manieren
Shapes: hoe de route op een kaart getekend moet worden d.m.v. een lijst GPS-coordinaten
Frequencies: Hergebruik van ritten vergemakkelijken door enkel de frequentie van een

bepaalde rit op te geven. De stoptijden van die rit worden dan berekend op basis van de tijden tussen haltes van een andere rit op dezelfde lijn.
Transfers: het aanmoedigen of ontmoedigen van overstappen op bepaalde locaties, of extra overstaptijd laten bijberekenen, bijvoorbeeld wanneer een metrostation zich ver onder de grond bevindt.
Feed Info: informatie over de feed zelf, voor als deze door anderen wordt ingelezen, bijvoorbeeld de publicatiedatum.

Het is in het kader van onderhoudbaarheid en transparantie een goed idee om gebruik te maken van open standaarden tegenover vendor lock-ins of een eigen implementatie die lastig te onderhouden is (FSFE, 2016). De keuze voor GTFS ligt dus voor de hand, enerzijds omdat het een open standaard is, anderszijds omdat het een veelgebruikte standaard is.

6. ONDERZOEKSVRAGEN EN –METHODIEK

Het doel van deze afstudeeropdracht is een antwoord en implementatie gebaseerd op de volgende hoofdvraag:

Hoe kan voor Nagasaki Bus een betere, modernere OV-routeplanner worden gerealiseerd?
Onder beter en moderner wordt verstaan de lijst requirements en nice to haves, die in de huidige routeplanner niet aanwezig zijn, uit hoofdstuk 4

Bij het beantwoorden van de hoofdvraag zijn antwoorden op de deelvragen hieronder vereist, deze bouwen op een gegrond resultaat.

(1) Hoe werkt de huidige routeplanner?

(2) Welke bestaande softwarepakketten voor routeplanners zijn er beschikbaar, zowel open-source als op de markt?

(3) Hoe is de dataopslag in het huidige systeem geregeld, en is dit te koppelen met andere, eventueel externe routeplanners?

(4) In hoeverre is de bestaande webapplicatie bruikbaar voor een softwareupgrade?

(5) Hoe kunnen landmarks worden geïntegreerd in een routeplanner?

(6) Welke mogelijkheden zijn er om een gevonden route weer te geven op een online kaartensysteem?

(7) Welke technieken en softwareomgevingen zijn geschikt om een nieuwe webapplicatie met nieuwe routeplanner te ontwikkelen?

(8) Welke mogelijkheden zijn er voor het berekenen van het tarief voor de gevonden gecombineerde route?

Er zal gebruik worden gemaakt van de volgende onderzoeksmethoden.

#	**Functie**	**Methode**
1	Beschrijvend	Observationeel
2	Beschrijvend, Vergelijkend	Literatuuronderzoek, experiment
3	Beschrijvend	Observationeel, literatuuronderzoek
4	Beschrijvend	Observationeel
5	Vergelijkend, Beschrijvend	Literatuuronderzoek, experiment
6	Vergelijkend	Literatuuronderzoek, experiment
7	Vergelijkend, Beschrijvend	Literatuuronderzoek, eventueel experiment
8	Vergelijkend	Literatuuronderzoek, experiment

Onder de gebruikte drie onderzoeksmethodes worden het volgende verstaan.

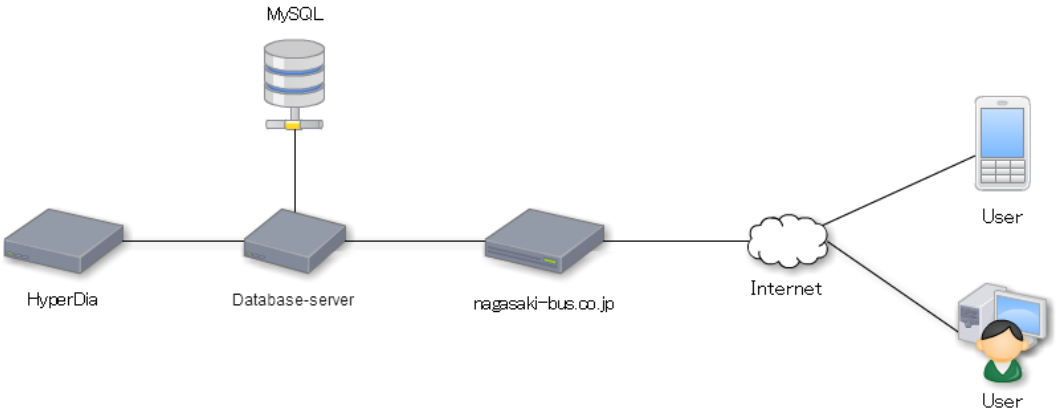
Observationeel: Een analyse van een bepaalde situatie gebruik makend van eigen expertise en beoordelingsvermogen.
Literatuuronderzoek: Geschreven informatie in de vorm van bijvoorbeeld vakliteratuur, wetenschappelijke literatuur, online bronnen en programmacode-documentatie en/of programmacode van voorbeeldimplementaties.
Experiment: Een testimplementatie, een proof of concept op zeer kleine schaal, van een aantal mogelijke oplossingen om hiermee te beoordelen of dit voor een grotere schaal wenselijk is en of de werking inderdaad zoals verwacht is.

7. ONDERZOEKSRESULTATEN

In dit hoofdstuk zal per deelvraag antwoorden worden gegeven op, en verantwoording worden geven per deelvraag. Uiteindelijk zullen deze antwoorden deel uitmaken van de invulling, en het antwoord op de hoofdvraag.

1. HOE WERKT DE HUIDIGE ROUTEPLANNER?

De huidige routeplanner is oorspronkelijk extern ontwikkeld in PHP en MySQL. De data van de dienstregeling bevindt zich in andere externe systemen (zie deelvraag 3). Deze data wordt door een weer een andere externe tool in een MySQL database ingelezen voor gebruik in de huidige routeplanner, de routeplanner roept deze vervolgens aan (zie hieronder). De tarieven voor de ritten komen vervolgens uit een ander fabrikantafhankelijk systeem welke op een vergelijkbare manier wordt geïmporteerd.



Figuur 4: Overzicht huidige serverstructuur

Hoewel schematisch gescheiden draait de homepage, de database-server en de huidige op dezelfde webserver. Er is documentatie aanwezig, maar dit gaat voornamelijk over hoe de data geimporteert moet worden. De code zelf is niet beschreven en er is geen toelichting op de architectuur (zie deelvraag 4). De zoekresultaten worden door middel van SQL-queries uit de database gehaald. Er zijn drie belangrijke tabellen: er is een tabel met informatie over haltes, en een tabel met alle haltes die gepasseerd worden per buslijn. Daarnaast is er nog wel een tabel met alle tarieven per buslijn, van iedere halte tot iedere andere halte binnen dezelfde buslijn.

Samenvattend is de huidige routeplanner een gesloten systeem dat data inzichtbaar maakt, op basis van exports uit andere gesloten systemen.

2. WELKE BESTAANDE SOFTWAREPAKKETTEN VOOR ROUTEPLANNERS ZIJN ER?

Welke routeplanners kunnen overweg met GTFS?

Onderzoek leert dat er een aantal softwarepakketten zijn die gebruik maken van GTFS, zoals beschreven in het Theoretisch Kader.

Google Maps

Google Maps vereist enkel een GTFS-feed en faciliteert daarmee dat die OV-data via Google Maps en alle daarvan afhankelijke applicaties werkt. Helaas wil men bij Nagasaki Bus door bedrijfsbeleid niet met Google in zee gaan en is deze optie niet mogelijk.

Node-GTFS

Node-GTFS is gebaseerd op Node.js en leest een GTFS-feed in en zet deze om naar een MongoDB-database. Vervolgens is er een Javascript API beschikbaar om de data te bevragen (Node-GTFS, 2016). Als toegevoegde waarde kan er ook gezocht worden naar dichtbijzijnde haltes of routes. Er is echter geen ondersteuning voor het zoeken van routes met eventuele overstap.

Graphserver

Graphserver is een op Python gebaseerde routing engine (Graphserver 2016) die gebruikt maakt van GTFS en OpenStreetMap (OSM, OpenStreetMap 2016), een open-source alternatief voor Google Maps. De software genereert een *graph*-tabel van de GTFS-data en gebruikt verschillende zoekalgoritmen om zoekresultaten te vinden. Het is mogelijk om zowel via haltes als via willekeurige GPS-locaties (zolang voor die gebieden een OSM-kaart ingelezen is) een zoekopdracht uit te voeren.

OpenTripPlanner

OpenTripPlanner (OTP) is een routeplanner ontwikkelt in Java die onder de motorkap gebruik maakt van het hierboven genoemde Graphserver, maar daar nog een aantal functionaliteiten aan toevoegt. Zo is OpenTripPlanner benaderbaar via een REST API en kan daarmee volledig extern, zelfs op een andere server draaien en blijft via de REST-calls nog steeds erg configureerbaar. Ook heeft OTP ondersteuning voor GTFS-realtime, een extentie voor GTFS die

kan voorzien in het registeren van realtime vertrektijden en realtime updates van de dienstregeling, zoals een tijdelijk onbruikbare halte. Andere functionaliteiten zijn ondersteuning voor het opnemen van fietsen in de route.

OneBusAway

OneBusAway (OBA) is een routeplanner ontwikkeld in Java en is speciaal gericht op bussen. Onder de motorkap maakt OBA gebruik van OpenTripPlanner. Het bestaat uit een aantal standaard applicaties zoals een smartphone app en een webversie. Er is ondersteuning voor overstappen via de ingebouwde OTP-server, maar deze functionaliteit is nog in beta en niet zo configureerbaar als via OTP stand-alone.

OSMSharp

OSMSharp is een routing engine gemaakt in C# met ondersteuning voor OpenStreetMap. Er is ondersteuning voor OV-systemen met GTFS gepland, maar dat is helaas op moment van schrijven nog niet beschikbaar.

Welke routeplanner is het meest geschikt?

De minimale eisen zijn:

- ✓ Ondersteuning voor GTFS
- ✓ Ondersteuning voor overstappen

Tabel 1: Vergelijking functionaliteiten van verschillende routeplanners

Functionaliteit	Node-GTFS	Graphserver	OpenTripPlanner	OneBusAway	OSMSharp
GTFS-ondersteuning	◯	◯	◯	◯	✕[1]
Overstappen	✕	◯	◯	✕[2]	✕
Zoeken vanaf GPS-locatie	✕	◯	◯	✕	◯
Web-API	◯	✕	◯	◯	✕
Realtime-OV informatie	✕	✕	◯	◯	✕
Dichtbijzijnde Haltes (GPS)	◯	✕	✕	◯	✕

◯ = aanwezig ✕ = niet aanwezig

Google Maps valt af door bedrijfsbeleid, en Node-GTFS en OSMSharp vallen beide af omdat ze respectievelijk nog geen overstappen of nog geen OV-informatie ondersteunen.

Bij alle hierboven genoemde routeplanners is er het probleem dat GTFS in het westen is ontwikkeld en er daarom weinig rekening is gehouden met andere manieren van het gebruik van OV-systemen en andere notaties van zaken. OBA lijkt dit het duidelijkst te laten zien; het is voor een specifieke situatie ontwikkeld en pas daarna open source geworden. Naast dat het erg afhankelijk is van routenummers (die in Japan weinig gebruikt worden), voegt het bovenop OpenTripPlanner weinig in de huidige situatie bruikbare functionaliteit toe. Wat wel wenselijk en in OBA aanwezig is, is het toevoegen van functionaliteit voor het vinden van bushaltes op basis van een locatie, maar daarmee is de overhead van het draaien van een gehele OBA-server niet verantwoord. Daarbij is de overstapfunctionaliteit nog in beta, en wordt dezelfde OTP-server aangeroepen met minder configuratiemogelijkheden dan een stand-alone OTP-server. OpenTripPlanner zorgt boven Graphserver ervoor dat de data extern benaderbaar is via een REST API, voegt toekomstelijke uitbreidingsmogelijkheden als GTFS-realtime toe.

[1] Gepland, maar nog niet officieel beschikbaar

[2] Nog in ontwikkeling, niet officieel beschikbaar

Bovenstaande tabel toont dat OpenTripPlanner de meest geschikte kandidaat is. Hoewel OneBusAway enkele functionaliteiten toevoegt, missen belangrijkste onderdelen zoals het kunnen zoeken via een GPS-locatie in plaats van enkel haltes.

3. HOE IS DE DATAOPSLAG IN HET HUIDIGE SYSTEEM GEREGELD?

Bij Nagasaki Bus wordt er voor het beheer van het busnetwerk en de buslijnen/ritten gebruikt gemaakt van een commerciële beheertool van Hitachi genaamd SmartDia (Hitachi, 2016). De tarieven voor de ritten komen vervolgens uit een ander fabrikantafhankelijk systeem welke op een vergelijkbare manier wordt geïmporteerd, waarover meer bij deelvraag 8.

SmartDia wordt gebruikt voor (1) het beheren van regio's en regiokantoren, remises, welke bushaltes waar zijn en welke faciliteiten ze eventueel hebben, welke routes er zijn, door welke regio's bepaalde routes gaan en het genereren van dienstregelingen per bushalte. Daarnaast is er (2) informatie over de dienstregeling, ritten, haltes per rit met aankomsttijden, en de diensten voor chauffeurs.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	改正ID	停留所コード	名称	ふりがな	英字名	1文字名称	2文字名称	3文字名称	4文字名称	5文字名称	駐在地フラグ	駐車場フラグ	待機場フラグ	休憩可フラグ	清掃可フラグ	納金可フラグ	設置場所コード	折返し加算有無	駐車可能台数	駐車停車時分	待機場所スターフ表示有無
2	29	00001	中央橋	ちゅうおうばし	CHUOBASHI	中	中央	中央橋	中央橋	中央橋	0	0	0	0	0	0	0	1	0	1	0
3	29	00002	浜の町	はまのまち	HAMANOMA	浜	浜町	浜の町	浜の町	十八銀行前	0	0	0	0	0	0	0	1	0	1	0
4	29	00005	親和銀行前	しんわぎんこ	SHINWAGINK	親	親和	親和銀	親和銀行	親和銀行前	0	0	0	0	0	0	0	1	0	1	0
5	29	00006	出島	でじま	DEJIMA	出	出島	出島	出島	出島	0	1	1	0	0	0	0	1	0	1	1
6	29	00007	長崎新地タ	ながさきしん	NAGASAKIST	新	新地	新地	新地TM	長崎新地	0	0	0	0	0	0	0	1	0	1	0
7	29	00008	大波止	おおはと	OHATO	大	大波	大波止	大波止	大波止	0	0	0	0	0	0	0	1	0	1	0
8	29	00009	元船町	もとふなまち	MOTOFUNA	船	船	元船町	元船町	元船町	0	0	0	0	0	0	0	1	0	1	0
9	29	00010	思案橋	しあんばし	SHIAMBASH	思	思案	思案橋	思案橋	思案橋	0	0	0	0	0	0	0	1	0	1	0
10	29	00011	興善町	こうぜんまち	KOZENMAC	興	興善	興善町	興善町	興善町	0	0	0	0	0	0	0	1	0	1	0
11	29	00012	県庁前	けんちょうま	KENCHOMA	県	県庁	県庁前	県庁前	県庁前	0	0	0	0	0	0	0	1	0	1	0
12	29	00013	市民病院前	しみんびょういん	SHIMINBYOI	市	市民	市民病	市民病院	市民病院前	0	0	0	0	0	0	0	1	0	1	0
13	29	00014	崇福寺入口	そうふくじいり	SOFUKUJIIR	ソ	ソ	崇福寺	崇福寺入	崇福寺入口	0	0	0	0	0	0	0	1	0	1	0
14	29	00015	常盤町	ときわまち		常	常盤	常盤町	常盤町	常盤町	0	0	0	0	0	0	0	1	0	1	0
15	29	00017	五島町	ごとうまち	GOTOMACH	五	五島	五島町	五島町	旭大橋	0	0	0	0	0	0	0	1	0	1	0
16	29	00018	出島ワーフ	でじまわーふ		出	出ワ	出島ワ	出島ワー	出島ワーフ	0	0	0	0	0	0	0	1	0	1	0
17	29	00019	市役所前	しやくしょまえ	SHIYAKUSH	市	市役	市役所	市役所前	市役所前	0	0	0	0	0	0	0	1	0	1	0
18	29	00020	桜町公園前	さくらまちこう	SAKURAMA	桜	桜町	桜町公	桜町公園	桜町公園前	0	0	0	0	0	0	0	1	0	1	0

Figuur 5: informatie over bushaltes uit SmartDia, hier staan gegevens als haltenaam, haltenummer, of het een enkel-opstap of -uitstap halte is, of het een plaats is waar de bus kan wachten als deze te vroeg aankomt, etc.

SmartDia is in functionaliteit zeer uitgebreid maar heeft tot verbazing geen tool die de data omzet naar het GTFS-formaat. Dit kan echter wel op aanvraag, en dit wordt dan met de hand gedaan door een medewerker bij Hitachi. Het is verstandig om hiervoor een eigen importtool te ontwikkelen om:

(1) niet afhankelijk te zijn van een externe partij

(2) de data met eigen validaties te kunnen valideren.

Er zijn echter wel een aantal problemen met GTFS in betrekking tot de beschikbare data vanuit Nagasaki Bus. Een analyse van een aangeleverd GTFS-bestand wees uit dat tarieven (zie deelvraag 8) niet geheel ondersteund worden door GTFS. Er zijn er ook problemen met aangeven van de route: In Japan wordt er veelal geen gebruikt gemaakt van een lijnnummer maar van een eindbestemming in combinatie met een aantal via-bestemmingen. Hiernaast een voorbeeld..

Figuur 6 *Een bus naar Nagasaki Shinchi Terminal via Togitsu, Kawahira en het stadshuis*

GTFS biedt hiervoor geen mogelijkheid, en de via-bestemming zullen of tussen haakjes na de eindbestemming moeten komen, of de via-bestemmingen moeten met behulp van een aanvullende API beschikbaar moeten worden gemaakt. Gezien er meer zaken zijn die niet, of niet in hun geheel door GTFS vervuld kunnen worden is het het verstandigst om hiervoor toch een eigen aanvullende API te ontwikkelen. De voorkeur gaat dan uit naar een Web-API gebaseerd op REST zodat zowel OTP als de eigen API op dezelfde manier extern benaderd kan worden.

Functionaliteit	Mate van ondersteuning binnen GTFS
Bushaltes	Aanwezig
Busroutes	Aanwezig, behalve het aangeven van via-haltes
Ritten	Aanwezig
Overstappen	Aanwezig
Tarieven	Aanwezig, behalve het specificeren van meer dan 1 tarief voor dezelfde aankomst en vertrekhalte. Ook geen ondersteuning voor een korting-tarief

Tabel 2: Aanwezigheid van functionaliteit binnen GTFS

Geanalyseerd is of de huidige data overgezet kan worden naar GTFS en in welke mate dat kan. Samenvattend is gekozen om de data te standaardiserennaar GTFS voor zover dat mogelijk is. De onderdelen die niet (direct) door GTFS ondersteund worden zullen worden toegevoegd door middel van een eigen-ontwikkelde API, waarover meer in de samenvatting aan het eind van dit hoofdstuk.

4. IS DE BESTAANDE WEBAPPLICATIE BRUIKBAAR VOOR EEN SOFTWAREUPGRADE?

In deelvraag 1 is de werking van het huidige algehele systeem behandeld. Een analyse van de webapplicatie zelf wijst uit dat er weinig tot geen rekening is gehouden met een goede software-architectuur; er is geen scheiding tussen de user interface (UI) en business logica, en ook is er geen database-abstractielaag. Er is geen mogelijkheid om te wisselen van UI, de domeinlogica is niet aanwezig en dus niet herbruikbaar, en de database is tevens niet uitwisselbaar. Daarnaast is de webapplicatie regelmatig herzien, bijvoorbeeld bugfixes en kleine nieuwe functies, met maar beperkte toelichting in de programmacode. De code staat ook vol met oude weggecommente code en is er in geen enkele vorm sprake van Clean Code (Martin, 2008). Er is documentatie aanwezig, maar dit zijn toelichtingen op de datastructuur, de werking van de omzettool en testplannen, maar niets over de code of architectuur. Veel hiervan komt wellicht door het feit dat de applicatie in 2002 ontwikkeld is, maar ook de gebruikte programmeertaal PHP maakt het organiseren van de architectuur niet makkelijker. Dit alles maakt het werken met dit systeem erg lastig, en is het, ook vanuit een architectureel standpunt verstandiger om een

nieuwe applicatie te ontwikkelen. Ook na overleg met collega's die regelmatig bugfixes of kleine toevoegingen hebben gedaan is naar voren gekomen dat ook zij enorm moeite hebben om de werking te begrijpen. Het antwoord op de deelvraag is dus een volmondig: nee, er zal niet worden voortgebouwd op de huidige applicatie. Dit ligt in het verlengde van het resultaat van deelvraag 1. Wat ontwikkeld zal worden wordt besproken in de samenvatting van de deelvragen.

5. HOE KUNNEN LANDMARKS WORDEN GEÏNTEGREERD IN EEN ROUTEPLANNER?

Landmarks zijn in Japanse context bepaalde locaties anders dan bushaltes waar men graag de optie voor wil hebben om heen te kunnen reizen. Voor het integreren van landmarks zijn er twee te onderscheiden stappen:

(1) **Welke data**: Het handmatig invoeren van landmarks dan wel zoeken via een externe bron, bijvoorbeeld een API

(2) **De Data gebruiken**: Hoe deze landmarks te benaderen, hiervoor zijn er twee manieren:

① Het koppelen van een landmark aan een bepaalde bushalte en de benodigde extra reistijd tijd hardcoden

② Een GPS-coordinaat koppelen aan een landmark en een zoekopdracht uitvoeren naar die GPS-locatie

Als voorbeeld het Nagasaki Peace Park.

Figuur 7: Een kaart van de omgeving van het Nagasaki Peace park

Mogelijkheid 1 is de halte Matsuyama-machi (de zwarte pijl links) toewijzen ongeacht waar men vandaan komt. “Het Nagasaki Peace Park is 3 minuten lopen van Matsuyama-machi”. Mogelijkheid 2 is een GPS-locatie aan het park toewijzen. Als de reiziger bijvoorbeeld niet van de grote gele hoofdweg, maar vanuit oostelijke richting komt kan men beter eerder uitstappen, bijvoorbeeld bij de halte bij de rode pijl zodat niet naar de toegewezen halte hoeft worden gereisd.

Aan beide scenario’s zitten nadelen wat betreft de brondata. Bij het zoeken via een externe API is de data niet in eigen beheer, en kunnen zoekresultaten onvoorspelbaar zijn. Bij het

handmatig invoeren van gegevens over landmarks is het eindresultaat voor gebruikers voorspelbaar, en kunnen slechte resultaten daarmee voorkomen worden, maar kost het tijd om deze data in te voeren, en ook regelmatig bij te werken.

Wat betreft het zoeken naar landmarks is het het meest wenselijk om via een GPS-coördinaat te zoeken tegen een vaste locatie, zoals de beschrijving van scenario 2, figuur 8, toont.

Nagasaki Bus heeft aangegeven geen mankracht te hebben om data in eigen beheer te houden, er is daarom besloten om de search-api van Google Maps te gebruiken. Gezien de te gebruiken routeplanner ook ondersteuning heeft voor het zoeken naar GPS-coordinaten is besloten om het zoeken ook via GPS te laten verlopen.

6. HOE KAN EEN GEVONDEN ROUTE WORDEN WEERGEGEVEN?

Voor het weergeven van een route is daar een specifieke mogelijkheid voor in Google's GTFS genaamd shapes.txt (Google 2016). Door middel van het opgeven van een reeks GPS-coördinaten voor een buslijn kan de software die de betreffende GTFS-feed inleest dan een lijn tekenen van die route op een willekeurige kaart, vanaf en tot waar van toepassing. De nauwkeurigheid van de route en grootte van de feed is hierbij een afweging van hoeveel coördinatener worden opgegeven, bijvoorbeeld om de 5 meter, om de 10 meter, om de 25 meter etc.

Helaas is het zo dat Nagasaki Bus geen data heeft over hoe de route in detail verloopt, anders dan GPS-coordinaten van de bushaltes. Wanneer deze gebruikt worden zouden worden zou het enkel een rechte lijn zijn van bushalte naar bushalte, en zorgt dit, zeker omdat Nagasaki een bergachtig gebied is met weinig rechte wegen, voor verwarring. Een andere mogelijke oplossing is om een autonavigatiesysteem te gebruiken, maar gezien de bus niet altijd de dezelfde route neemt als de route die zo een systeem voorspelt klopt ook deze benadering niet voldoende.

In de toekomst zijn er geen plannen om gedetailleerd te beschrijven hoe de routes gereden worden. Navraag over hoe nieuwe chauffeurs hun route dan leren leverde ook niets op, dit gaat via proefritten en papieren kaarten; het digitaliseren hiervan laat ook op zich wachten.

Als tussenoplossing is in overleg besloten om dan alle bushaltes te highlighten zonder een lijn te tekenen op een Google Maps-kaart zoals het voorbeeld hieronder.

Figuur 8: Voorbeeld van een route waarbij de gepasseerde bushaltes zijn aangegeven. Dit is een rit van ongeveer 8km over 30 minuten met 17 haltes.

Ondanks dat dit geen gedetailleerde lijn is hoe de route precies loopt, is het wel een goede indicatie. Zodra gedetailleerde data beschikbaar is kan deze functionaliteit uitgewisseld worden.

7. WELKE SOFTWAREOMGEVINGEN ZIJN GESCHIKT VOOR EEN NIEUWE WEBAPPLICATIE MET ROUTEPLANNER?

Zoals deelvraag 4 al uitgewezen heeft is er op dit moment geen sprake van een software architectuur. Het is binnen software engineering gemeengoed om software zo modulair en onderhoudbaar mogelijk op te zetten om toekomstige wijzigingen of toevoegingen makkelijk te maken. Mijn insteek is in dit project als volgt.

Het gebruikt van de GTFS zorgt ervoor dat

(1) De data abstract is van de software die deze gebruikt

(2) OTP zorgt ervoor dat het zoeken en het zoekalgoritme abstract kan worden gehouden via een REST API.

   ($\overline{\mathcal{A}}$) De onderdelen die OTP niet ondersteund kunnen het best worden aangevuld met een eigen ontwikkelde API zoals al eerder aangegeven. De meest flexibele oplossing is dus om de rest van de functionaliteit ook op dezelfde manier beschikbaar te maken zodat er geen afhankelijkheden ontstaan met bijvoorbeeld proprietaire bestandsformaten.

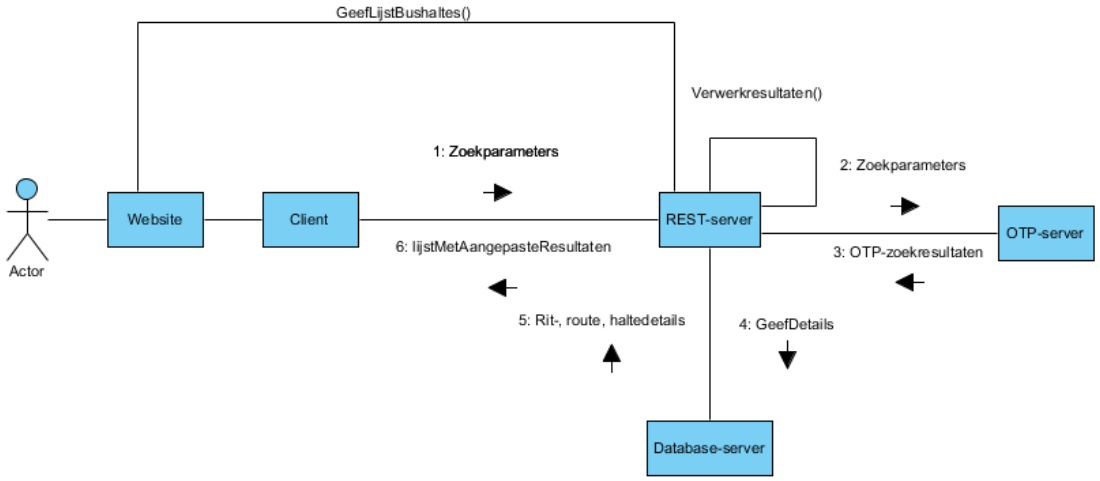
(3) De User Interface kan gescheiden worden ontwikkeld.

   ($\overline{\mathcal{A}}$) In eerste instantie is een webapplicatie ontwikkeld die deze API's aanroept om zo tot zoekresultaten te komen. Het gebruik van REST API's zorgt ervoor dat dezelfde data ook in andere applicaties gebruikt kan worden, en er aan de server kant niets hoeft te worden aangepast als er bijvoorbeeld voor een mobiel apparaat een eigen interface wordt ontwikkeld.

De situatie wijst uit dat OTP alleen lang niet in alle functionaliteiten kan voldoen. OTP dient aangevuld en bijgewerkt te worden aan de eisen. Dit kan op twee plaatsen gebeuren. (1) Door via de client een zoekopdracht naar OTP sturen, en de ontbrekende zaken van de eigen REST api ophalen. (2) De zoekopdracht dirigeren naar de eigen REST API, deze de zoekopdracht laten uitvoeren en corrigeren, en het eindresultaat door laten sturen.

In het kader van eerdergenoemd modulair ontwerpen van software is het de verstandigste keuze om voor optie (2) te gaan en hiermee de client geen domein-logica te laten uitvoeren. De client is puur alleen een doorgeefluik voor de zoekparameters (vanaf waar, hoelaat…) en moet vervolgens zorgen voor een representatie van de zoekresultaten.

Systeemstructuur



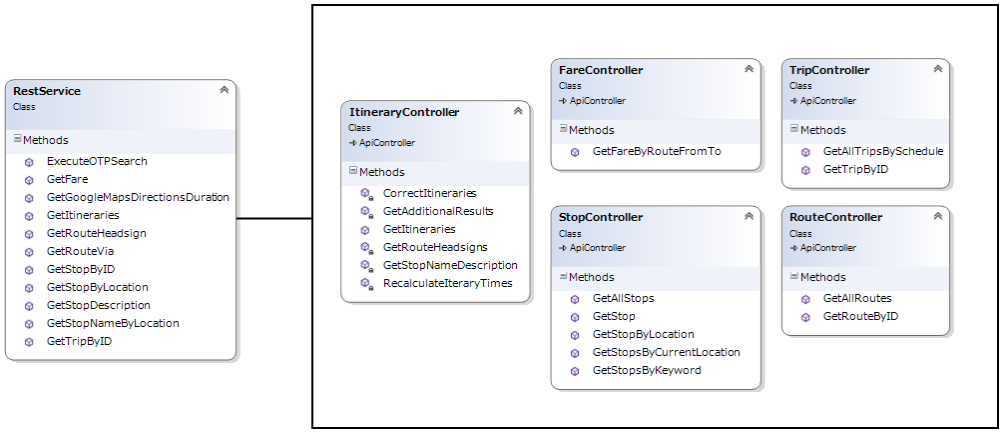
Figuur 9: Communicatiemodel

De zoekresultaten van OTP worden ingelezen in het domeinmodel, daar aangepast aan de business requirements van Nagasaki Bus, en vervolgens naar de client gestuurd via REST-objecten. De verantwoordelijkheden zijn als volgt:

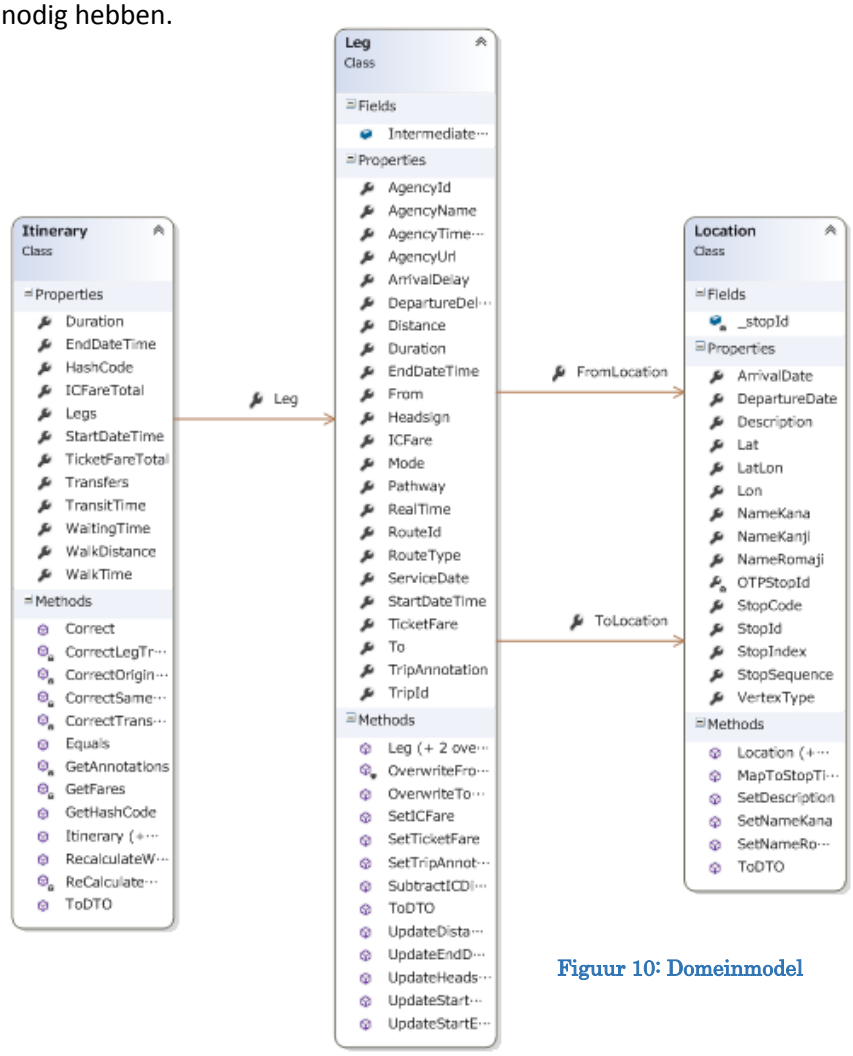
Server: Zoeken en corrigeren, omzetten van domein naar DTO
De rol van de server is om de zoekresultaten en de aanvullende data te combineren en te valideren. Daarna zullen deze tot Data Transfer Objects (DTO) worden omgezet en via de REST-service aangeboden worden.

Client: Een Interface bieden om een zoekopdracht samen te stellen, en weergave van de van de ontvangen DTO
De rol van de client is enkel om de zoekparameters te verzamelen, deze te verpakken in een SearchParameters-object, en vervolgens de DTO die de server terugstuurt in HTML op de website weer te geven.

De server-architectuur is als volgt:



De server biedt een enkele interface, in dit geval de RestService die vervolgens voor Itinaries (zoekresultaten) OTP aanroept en deze bijgewerkt aan de business requirements terugstuurt. Voor de andere aanvragen verwijst de service door naar de eigen databaseserver. Het domeinmodel bestaat alleen uit Itinary omdat de overige objecten slechts DTO's zijn (volgende pagina) en geen interne logica nodig hebben.



Figuur 10: Domeinmodel

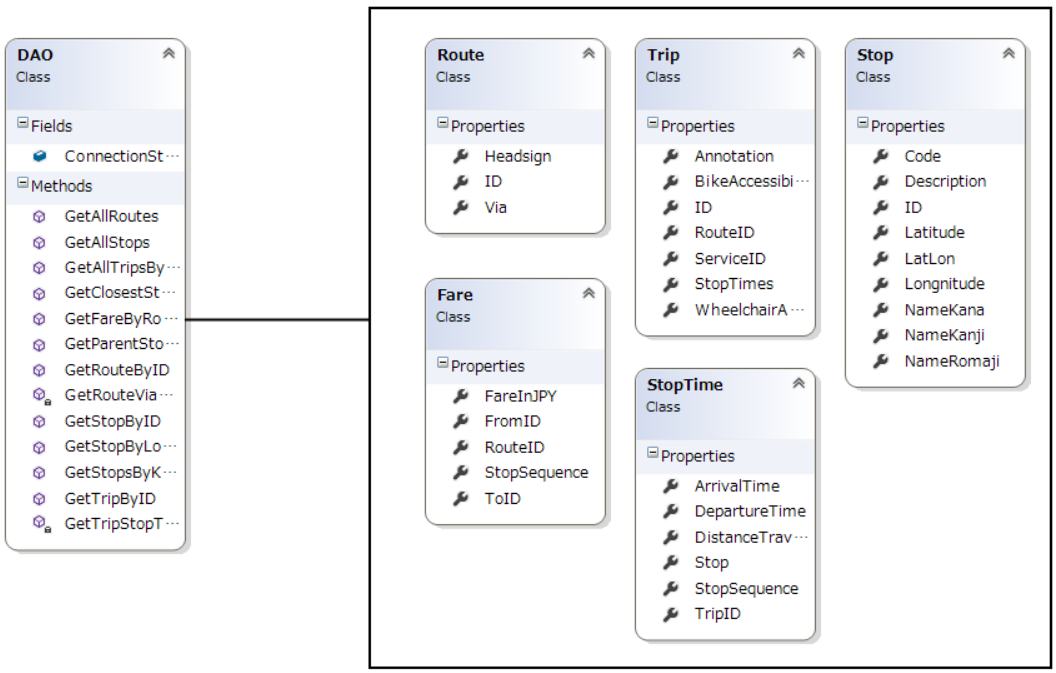
Een Itinerary is het hele reisadvies, een lijst **routes** en informatie over het geheel.

Figuur 11: Voorbeeld van een zoekoverzicht, vertrekpunt, bestemming en tijd, en een aantal opties met vertrek, aankomst, reistijd, tarief en aantal overstappen.

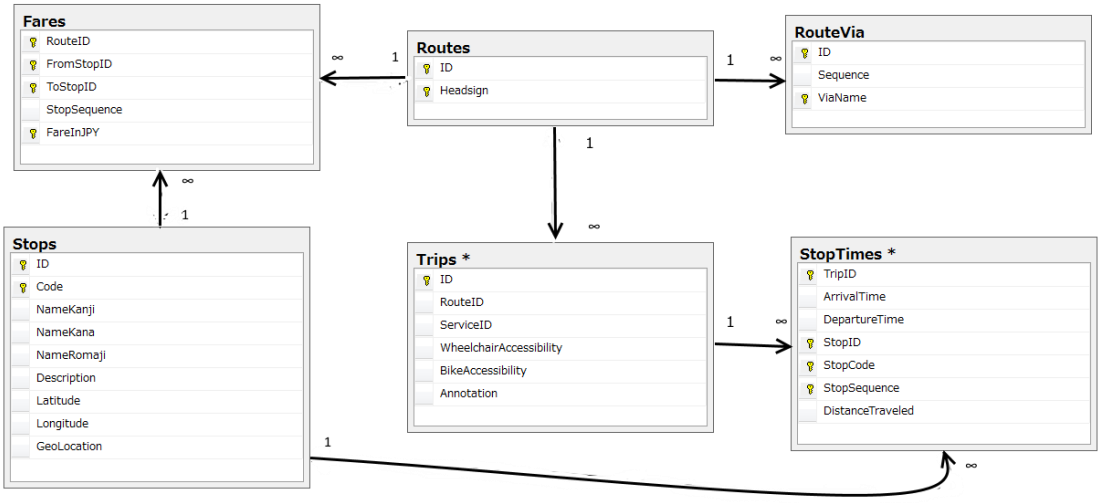
Een leg is een ‘stap’ binnen een route, een busreis, een stukje lopen etc

Een Location is een vertrek- of aankomstlocatatie via een GPS-locatie of een bushalte.

Figuur 12: Een voorbeeld van een route, in dit geval met 2 legs

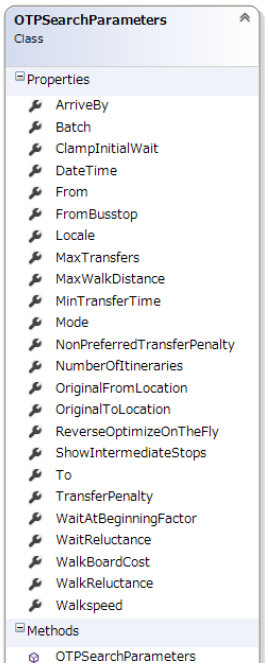


Figuur 13: Databasemodel



Figuur 14: Database ERD

Om zoekresultaten te krijgen zoals in figuur 11 en 12 is er enkel een OTPSearchParameters-object nodig (figuue 15) die met de REST-request moet worden meegestuurd. De server stuurt vervolgens een lijst Itineraries terug in DTO-vorm. Via het parameterobject kan ook extern invloed worden uitgeoefend over met welke voorkeur gezocht moet worden.



Figuur 15: Zoekparameter-object

Platform

Er zijn geen beperkingen op het gebied van bijvoorbeeld bedrijfsbeleid over welke platformen gebruikt moeten worden. Een veelgebruikt platform voor server-side REST API's is Java, maar ook andere platformen zoals ASP.NET zijn hier bijzonder goed voor geschikt.

Qua platform is besloten om de server-side in ASP.NET te ontwikkelen, en de client side ook in ASP.NET te ontwikkelen. Gezien de client geen domeinlogica bevat kan deze ook makkelijk worden uitgewisseld voor een andere technologie, of kan er zonder aan de server-kant iets aan te passen een nieuwe interface voor bijvoorbeeld mobiele apps worden ontwikkeld.

Om te faciliteren in de wens om ook een website voor mobiele apparaten te ontwikkelen is gekozen om gebruik te maken van het mobile-first framework Bootstrap (Bootstrap, 2016). Dit zorgt ervoor dat er slechts een enkele website hoeft te worden ontwikkeld zonder veel onder te doen voor dedicated-mobiele websites. Tegelijkertijd is het dan gezien de beperkte tijd mogelijk om zowel een desktop- als smartphonewebsite te ontwikkelen.

8. WELKE MOGELIJKHEDEN ZIJN ER VOOR HET BEREKENEN VAN HET TARIEF VOOR DE GEVONDEN GECOMBINEERDE ROUTE?

Voor het berekenen van het tarief voor een route is daar een specifieke mogelijkheid voor binnen GTFS genaamd fares.txt en fares_rules.txt (Google, 2016). Deze mogelijkheid is echter wel beperkt tot een aantal specifieke situaties. Voorbeelden hiervan zijn:

(1) een zone-systeem zoals in Londen gebruikelijk is, het tarief gaat omhoog bij het betreden

(2) Hetzelfde tarief waar je ook opstapt of uitstapt (wordt gebruikt bij de tram in Nagasaki)

De variatie die het meest dicht bij Nagasaki Bus in de buurt komt is een vast tarief van bushalte naar bushalte.

Voor elke busroute is er van elke halte naar elke andere halte een tarief vastgesteld. Het probleem is echter dat er ook buslijnen zijn die dezelfde bushalte vaker dan 1 keer aandoen omdat ze een woonwijk ingaan, op een bepaald punt keren en via dezelfde route weer terug gaan. Op de terugweg uitstappen is soms duurder omdat op dat moment net een zone gepasseerd kan worden. Ook zijn er routes die de ene keer ‘linksom’ gaan en de andere keer ‘rechtsom’, dat wil zeggen soms komt het voor dat dezelfde halte dichtbij is, en soms duurt het bijvoorbeeld een half uur langer omdat het een behoorlijke omweg is. Ook hier verschilt het tarief. Het probleem is echter dat het externe systeem waaruit de tarieven worden geëxporteerd slechts 4 velden geeft:

```
"系統番号","出発地停留所","到着地停留所","普通運賃"
"0100","0007","0001",015
"0100","0007","0012",015
"0100","0007","0008",015
"0100","0007","0017",015
"0100","0007","0027",015
"0100","0007","0026",015
"0100","0007","0035",015
"0100","0007","0040",015
"0100","0007","0049",015
"0100","0007","0056",015
"0100","0007","0066",015
"0100","0007","0082",016
"0100","0007","0088",016
"0100","0007","0095",016
"0100","0007","0098",016
"0100","0007","0101",016
"0100","0007","0108",016
"0100","0007","1218",019
"0100","0007","0184",019
```

Figuur 16: Een voorbeeld van een csv-**export van het tarievensysteem**

De velden zijn (1) routenummer (eerste 3 cijfers) en richting (laatste cijfer), (2) opstaphaltenummer, (3) uitstaphaltenummer en (4) tarief. Voor de eerste regel is dat route 010, heenweg, Opstap: Nagasaki Shinchi Terminal, Uitstap: Chuobashi, tarief ¥150 (€ 1,17). De volgende is hetzelfde maar dan tot Kenchomae. Bij regel 13, vanaf Asahimachi-Sanbashimae verandert het tarief naar ¥160 (€ 1,25).

Op dit moment is het dus zo dat wanneer een bushalte vaker wordt aangedaan er meer dan een tarief naar voren komt. Er is een aanvraag gedaan naar de ontwikkelaar van het tarievensysteem om een extra veld, het volgnummer, ook in de export op te nemen. Zodra deze beschikbaar is kunnen tarieven in hun geheel opgenomen worden. Het is onwaarschijnlijk dat het zoekalgoritme met opzet de lange variant waar geen tarief voor is selecteert, derhalve voldoet het (voorlopig) voor Nagasaki Bus om deze extra tarieven niet op te nemen.

Binnen GTFS is overstappen en overstapkorting helaas ook erg beperkt. Voor overstappen is alleen mogelijk om aan te geven of dat helemaal niet, 1 keer, 2 keer of oneindig kan. Bij Nagasaki Bus kan overstappen niet, en moet het opstaptarief ook voor elke overstap opnieuw betaald worden. Er is echter wel 5% korting als binnen 30 minuten met dezelfde smartcard opnieuw ingecheckt wordt, maar ook weer niet 2 keer achter elkaar. Dit geldt weer niet voor mensen die contant betalen.

Korting met de smartcard geldt wanneer:

A) De overstap binnen 30 minuten is

B) De overstap op dezelfde halte is (lopen naar een andere halte mag niet)

C) Geen korting-op-korting, niet constant korting (na een keer korting heeft de daaropvolgende bus nooit korting)

D) Maximaal 3 keer korting korting per kalenderdag

E) Korting is 5% van het tarief van bus 1 en bus 2, afgerond op ¥ 10

De berekening werkt bijvoorbeeld als volgt:

Stap 1	Bus V: 11:50 A: 12:10	Bus V: 18:10 A: 18:12	Bus V: 13:01 A: 13:23	Bus V: 13:01 A: 13:09
Stap 2	Bus V: 12:25 A: 13:00	Lopen V: 18:13 A: 18:20	Bus V: 13:41 A: 14:00	Bus V: 13:42 A: 14:00
Stap 3	×	Bus V: 18:41 A: 20:00	Bus V: 14:10 A: 14:45	Bus V: 14:32 A: 14:44
Stap 4	×	×	Bus V: 15:15 A: 16:00	Bus V: 15:13 A: 16:00
Stap 5	×	×	Bus V: 16:04 A: 16:11	Bus V: 16:34 A: 16:11
Korting	5% van Bus 1 en 2	Geen korting	5% van Bus 1 en 2 5% van Bus 4 en 5	5% van Bus 3 en 4
Voldoet niet aan		B	C	A

Tabel 3; Voorbeelden tariefberekening

GTFS biedt geen toelaas voor de situatie die bij Nagasaki Bus van toepassing is. De tarieven zullen dus via de eigen API toegevoegd worden aan de zoekresultaten. Een totaaltarief zal zowel voor een normaal ticket als voor een smartcard per zoekresultaat, en per bus worden weergegeven. Verder is in overleg besloten om bussen zonder tarief of met dubbele tarieven worden niet in de routeplanner op te nemen.

SAMENVATTING

Aan de hand van de onderzoeksresultaten is er samengevat het volgende besloten.

De bestaande webapplicatie zal niet gebruikt worden, de routeplanner zal nieuw ontwikkeld worden (deelvraag 1 en 4) Deze zal gebruik maken van enkel de routing engine van de open source routeplanner OTP (2). De data van dienstregelingen uit huidige systemen is hierbij bruikbaar (3), en zal worden opgeplist in:

A GTFS zodat deze universeel bruikbaar is, in dit geval met OTP.
B Een database voor de eigen ontwikkelde API ter aanvulling van GTFS en OTP

Landsmarks (5) zullen bruikbaar zijn in de vorm van

A de AutoComplete-API van Google Maps en
B vrij ingeefbaar via een kaart (hiermee wordt ook direct het reizen van/naar adressen ondersteund)

De gevonden route kan niet in detail weergegeven worden omdat de data daarvan ontbreekt, daarvoor in de plaats wordt door middel van het markeren van iedere bushalte de route in grote lijnen weergegeven (6).

De tarieven voor ritten kunnen niet via GTFS worden geconfigureerd, hiervoor zal een eigen API worden ontwikkeld die ook enkel andere zaken aanvult waarin GTFS niet kan voorzien(8).

Hiermee zijn de deelvragen beantwoord en heeft dit geleid tot het eindproduct (antwoord op de hoofdvraag).

8. EINDPRODUCT

In dit hoofdstuk wordt het gerealiseerde eindproduct en het verloop van de ontwikkeling hiervan toegelicht.

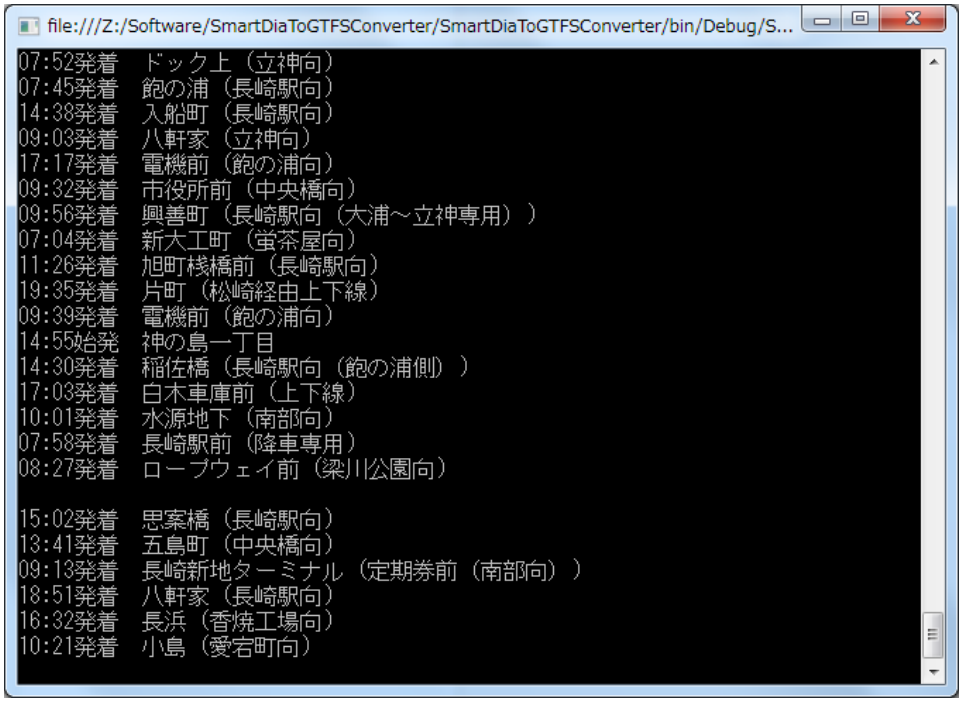
Het eindproduct is in eerste instantie als proof of concept tegelijkertijd met het onderzoeken van de deelvragen ontwikkeld. Ongeveer iedere twee weken is er verslag gemaakt van de voortgang en vergaderd over wat de een goede volgende stap zal zijn. Soms was dat een wijziging, of vraag om een verdere uitwerking in dezelfde richting. De projectmethode lijkt op een soort scrum; iteraties met iedere keer een feedback-loop.

Het project is onder te verdelen in twee onderdelen: (A) data bruikbaar maken (B) data doorzoekbaar maken.

A DATA-IMPORTTOOL

Voor het inlezen van de bestaande data is gekozen om in eerste instantie een commandline tool te ontwikkelen. De reden daarvoor is dat deze enkel gebruikt wordt bij een wijziging van de dienstregeling wat slechts sporadisch voorkomt, twee of drie keer per jaar. Ook zal dit door technische mensen gedaan worden en is een UI niet noodzakelijk.

De eerste iteratie is gedaan direct na de start van het project, zodat er een basis was om de verschillende routeplanners op te testen.



Figuur 17: De importool: de voortgang wordt constant in de console getoond. Hier worden alle vertrektijden ingelezen.

Men dient de *master*-data (data over buslijnen en routes) en vervolgens 3 keer een dienstregeling op te geven, voor doordeweeks, zaterdagen en zondagen. De tool leest de data vervolgens automatisch in, valideert deze (o.a. bestaat de betreffende bushalte van de rit wel) en output dit naar een GTFS-bestand, en genereert vervolgens SQL-scripts om de data van de eigen REST API beschikbaar te maken in een SQL-server.

Er is nog geen ondersteuning voor het toevoegen van speciale dienstregelingen voor vakantieperioden, en zo moeten bijvoorbeeld nachtbussen en andere niet-standaard bussen volgens bedrijfsbeleid niet in de reguliere routeplanner verschijnen terwijl ze wel in de data-export van SmartDia aanwezig zijn.

TWEEDE ITERATIE

De eerste iteratie was redelijk compleet, maar voldeed op enkele details nog niet helemaal aan de eisen. Deze zijn aangevuld in de tweede iteratie, die na het tweede protype heeft plaatsgevonden.

Bij het opgeven van dienstregelingen kunnen aanvullend nog drie dienstregelingen opgegeven worden voor feestperioden, vaste perioden met een afwijkende dienstregeling. Men kan dan zelf aangeven in het GTFS calendar_dates.txt bestand welke dienstregeling op welke datum moet gelden.

Het probleem met nachtbussen en andere uit te sluiten bussen is opgelost door deze in het systeem niet op reguliere haltes te laten stoppen, maar daarvoor speciale onzichtbare dummies in de plaats te gebruiken. Bij het converteren van de data worden deze dummies genegeerd. Men hoeft dus bij het in de toekomst mogelijk wel willen opnemen alleen maar die bushaltes naar de niet-dummy versie terug te veranderen.

PROTOTYPE 1

Eind december is de eerste bruikbare testversie opgeleverd, zie figuur 12.

Figuur 18: Het zoekscherm van het eerste prototype

Bij dit prototype is direct de meest belangrijke functionaliteit in de basis aanwezig: het is mogelijk om routes te vinden inclusief waarbij daar een overstap nodig is. Daarnaast is de lijst met beschikbare bushaltes volledig, en doorzoekbaar. De vertrek- of aankomstdatum en tijd kan worden opgegeven, maar de software maakt nog enkel onderscheid tussen doordeweeks, zaterdag en zon- feestdagen. Verder kan het aantal resultaten en de volgorde van weergave, bijvoorbeeld benodigde tijd, vertrektijd of het aantal overstappen aangegeven worden, alsook

is er een optie voor extra overstaptijd. Ook kunnen de gepasseerde haltes worden getoond per bus.

In het voorbeeld hiernaast wordt gezocht van Hongawachi (本川内) naar Mie (三重), waarbij in alle gevallen een overstap vereist is, de huidige routeplanner geeft bij deze zoekopdracht geen resultaten. In dit geval is dat bij de bushalte Rokujizomae (六地蔵前) De 1 minuut lopen (徒歩 1 分) is het verwisselen van richting bij de bushalte.

Figuur 19; Voorbeeld van een zoekresultaat

FEEDBACK

Dit prototype is bijzonder goed ontvangen door het bedrijf, men was zeer onder de indruk van de ontwikkelingen in een korte termijn. In dit stadium zijn echter wel een aantal problemen naar voren gekomen. Het grootste probleem hierbij is het valideren van de uitkomsten van de routeplanner. Het probleem is hier niet zo zeer dat er valide resultaten uitkomen, maar dat deze data ook semantisch correct moet zijn. Hiermee wordt specifiek bedoeld dat een route die theoretisch correct is en ook bestaat in de dienstregeling, maar voor de klant vreemd aanvoelt, als onjuist moet worden gezien. In Japan is het gebruikelijk dat zodra iets voor de klant beschikbaar is het ook volledig moeten kunnen worden gewaarborgd dat dit ook juist is.

Om dit probleem op te lossen zijn een aantal voorstellen gedaan, bijvoorbeeld een beta-versie online zetten, of een testversie voor gebruik binnen het bedrijf opzetten, maar geen van deze werden goedgekeurd. Uiteindelijk is besloten om door middel van een logmechanisme zoekopdrachten op de huidige zoekmachine te loggen en deze nogmaals uit te voeren op de nieuwe, en daarmee een goed inzicht te krijgen of de zoekmachine inderdaad goede resultaten geeft. Het gebruik van realistische data is hierbij zeer belangrijk. Dit vormt dan ook gelijk het belangrijkste onderdeel voor de acceptatietest.

Ook is er een probleem (2) dat een bepaalde bushaltes door Nagasaki Bus als overstaphalte zijn aangemerkt, maar het objectief gezien niet uitmaakt welke halte wordt gebruikt als er verschillende overstapmogelijkheden van en naar dezelfde bus zijn. Ook komt het voor dat het zoekalgoritme de passagier langer in een bus laat zitten om de wachttijd buiten te verkorten. Dit is inderdaad wenselijk als het buiten koud is, maar als daardoor het tarief omhoog gaat of een overstap erg krap wordt is dat al minder het geval.

PROTOTYPE 2

Het tweede prototype is opgeleverd rond de eerste conceptversie van de scriptie eind januari, en is uiteraard een doorontwikkeling van het eerste prototype.

Figuur 20: Het zoekscherm van prototype 2: een zoekopdracht van een bushalte naar een dichtbijzijde bushalte.

Er zijn veel wijzigingen aangebracht in voornamelijk het uiterlijk en hoe de applicatie aanvoelt bij gebruik. Daarbij is er ondersteuning voor het zoeken vanaf/naar landmarks via de Google Maps Locations API, is er ondersteuning voor het bepalen van de huidige positie, en het vinden van bushaltes in de buurt. Ook is het zoeken van bushaltes aangepast naar een realtime zoekfunctie via AJAX met een eigen REST API. Daarnaast kan een zoekopdracht ook gestart worden door willekeurig ergens op de kaart te klikken.

Figuur 21 Zoekresultatenscherm prototype 2

Tevens is er in het zoekresultatenscherm (figuur 15) ondersteuning voor het bekijken van de busroute, en is er navigatie voor de ene naar de andere bushalte wanneer een overstap nodig is.

FEEDBACK

Ook dit prototype is goed ontvangen met enkel een aantal aanvullende nice to haves. Deze waren: het eerst tonen van een overzicht van alle zoekresultaten in plaats van direct alle details, en het kunnen aangeven of men (1) liever niet wil lopen maar dan langer onderweg is of vaker moet overstappen, of (2) het niet erg vinden om een stuk te lopen als men daarmee veel sneller op de eindbestemming kan zijn.

PROTOTYPE 3/EINDPRODUCT

Ook bij dit prototype, dat uiteindelijk het eindproduct is geworden, is veel aandacht besteed aan gebruiksgemak, maar ook vooral gebruikerservaring. Het gewenste zoekoverzicht is

eïmplementeerd en de data is gevalideerd en goedgekeurd. Alle vooraf bepaalde, enanvullende eisen zijn ook tot waar mogelijk gerealiseerd.

結果一覧

#	出発	到着	所要時間	乗車時間	待ち時間	徒歩時間	運賃	乗り継ぎ回数
1	15:04 発	**15:52 着**	00:48	**00:24**	—	00:23	**290円**	**0回**
2	15:25 発	16:13 着	00:49	00:32	00:12	**00:05**	480円	1回
3	16:04 発	16:52 着	00:48	**00:24**	—	00:23	**290円**	**0回**
4	16:25 発	17:05 着	00:41	00:27	00:05	00:09	440円	1回
5	16:47 発	17:22 着	**00:36**	00:30	—	**00:05**	310円	**0回**

もっと見る

Figuur 22: Resultatenoverzicht: Vertrek, Aankomst, reistijd, tijd in voertuig, wachttijd, looptijd, tarief en aantal overstappen.

Ook is er ondersteuning voor smartphones toegevoegd, zie figuur 17 hiernaast.

Figuur 23: Smartphone-weergave van een zoekresultaat

In dit prototype zijn ook alle resterende problemen, ofwel zorgpunten opgelost. Om het probleem van het overstappen bij niet-aangewezen haltes te voorkomen is gekozen om de zoekresultaten van het algoritme achteraf te overschrijven en altijd de eerste mogelijke gezamenlijke halte van de twee ritten te gebruiken.

Een ander probleem dat soms voorkwam zijn fouten in de kaarten, soms leek het voor het algoritme zo dat het lopen naar de volgende halte sneller is dan wachten op de bus (als men toch moet wachten), ondanks dat diezelfde bus de vertrekhalte ook aandoet. Ook kwam het soms voor dat de halte voor de heenweg en terugweg zo ver uit elkaar lagen dat de volgende halte qua loopafstand dichterbij was. Beide problemen zijn opgelost door te forceren dat altijd vanuit de vertrekhalte wordt gereisd tenzij de bus deze niet aandoet.

Het overschrijven van overstaphaltes en de vertrekhaltes heeft het eerder genoemde probleem met mogelijk niet-semantisch correcte routes tot een minimum beperkt. Het enige probleem dat soms voorkomt is dat de kaartgegevens een kleine binnendoorweg niet bevat en de reistijd enkele minuten langer wordt aangegeven dan deze eigenlijk hoeft te zijn. Ook zijn sommige bushaltes niet helemaal correct gepositioneerd, en kan op de kaart niet gevonden worden vanaf waar de zoekopdracht precies wordt gestart. Dit zijn echter problemen die extern opgelost moeten worden.

SAMENVATTING

Het eindproduct is een nieuwe routeplanner die aan alle eisen voldoet die zijn gesteld bij in hoofdstuk 2. De ontwikkelde functionaliteiten zijn:

Zoeken

- Realtime zoeken naar bushaltes
- Dichtstbijzijnde bushaltes via GPS zoeken
- Zoeken via een adres, of door op een kaart op een locatie te klikken
- Het kunnen opgeven van extra overstaptijd

- Het kunnen aangeven te willen lopen om reistijd te verkorten

Zoekresultaten

- Een vergelijkend resultatenoverzicht
- Specificatie van reistijd, afstand, benodigde tijd, tarieven en automatische berekening van korting
- Het kunnen bekijken van de gehele rit (van startpunt tot eindpunt), het kunnen bekijken van de hele dienstregeling van een bushalte
- Weergaven van de route op een kaart
- Het kunnen bekijken waar de bus onderweg stopt, en waar precies opgestapt of uitgestapt wordt
- Een printervriendelijke versie van de zoekresultaten

9. DISCUSSIE EN AANBEVELINGEN

In het vorige hoofdstuk is de ontwikkeling en uiteindelijk het eindproduct gepresenteerd. Hierbij zijn bepaalde ontwerpkeuzes gemaakt die eventueel openstaan voor een andere implementatie. Er is gekozen om zoveel mogelijk gebruik te maken van open standaarden (GTFS) en data extern beschikbaar te maken via REST Web-services.

Wat niet ontwikkeld is, is een native app voor smartphones. De gekozen tussenoplossing, het mobile-first framework Bootstrap is echter niet perfect. Met mobiele apparaten is er altijd het probleem dat er veel schermformaten en –resoluties zijn, dat is bij desktops een minder groot probleem. Daarnaast kan de user-interface qua gebruiksgemak en layout nog verbeterd worden, maar dat is een ander vakgebied.

Bij het zoeken naar routes zijn er een aantal zaken die vooraf bepaald moeten worden, zoals hoe krap of ruim overstappen mogen zijn, hoe ver de gebruiker wil lopen (of juist helemaal niet), reist men liever om of wacht men liever etc. Deze instellingen zijn in het configuratiebestand van de webapplicatie toegankelijk gemaakt zodat ook tijdens runtime

aangepast kan worden aan welke zoekresultaten de voorkeur wordt gegeven, dit kan na publicatie nog uitgebreid geëvalueerd worden.

Er is ondersteuning ingebouwd voor andere OV-bedrijven dan de huidige twee, Nagasaki Bus en Saikai-kotsu (een zuster-busbedrijf waarvan het beheer ook bij Nagasaki Bus ligt). Het bleek echter lastig om data van andere bedrijven te verkrijgen, en ook al zou deze verkregen zijn, is het betrouwbaar bijwerken een hekel punt. Er is bijvoorbeeld gedacht om dienstregelingen van ferry's tussen verschillende de eilanden op te nemen, of bijvoorbeeld de kabelbaan naar Mt. Inasa of de trams die alleen het stadscentrum aandoen.

Het eindproduct is volledig, bruikbaar en boven verwachting, maar zou nog ietswat uitgebreid kunnen worden op enkele ovengenoemde zaken die in dit project, door de beschikbaarheid van alleen mijzelf voor slechts enkele maanden, te veel werk was ook nog te realiseren.

10. BIBLIOGRAFIE

Bootstrap. Opgevraagd op 3 maart 2016, van

http://getbootstrap.com/about/

FSFE. Open Standards. Opgevraagd op 1 maart 2016, van

https://fsfe.org/activities/os/def.en.html

GraphServer, *GraphServer* Opgevraagd op 18 januari 2016, van

http://graphserver.github.io/graphserver/

Greenhorne & O'Mara. *Google Transit Feed Specification.* Opgeroepen op 18 januari 2016, van

http://kb.g-and-o.com/wiki/images/GTFS_Schema.png

Google. *General Transit Feed Specification Reference*. Opgeroepen op 18 januari 2016, van

https://developers.google.com/transit/gtfs/reference

Google. PublicFeeds.wiki. Opgeroepen op 3 maart 2016, van

https://code.google.com/archive/p/googletransitdatafeed/wikis/PublicFeeds.wiki

GTFS Data Exchange. Transit Agencies Providing GTFS Data. Opgeroepen op 3 maart 2016, van http://www.gtfs-data-exchange.com/agencies

Hitachi. *HyperDia:* クラウド型ダイヤ作成システム (*cloud-based systeem voor het beheer van dienstregelingen*). Opgeroepen op 18 januari 2016, van http://www.hitachi-systems.com/solution/s006/smartdia/

Robert C. Martin: Clean Code: A Handbook of Agile Software Craftsmanship, Pearson Education, 2008

Nagasaki Bus. 時刻・運賃クイック検索 (zoekmachine voor het snel vinden van tijd en tarieven). Opgevraagd op 18 januari 2016, van http://nagasaki-bus.co.jp/dia-search/index.php

Nagasaki Bus. 会社概要 (bedrijfsoverzicht). Opgevraagd op 16 februari 2016, van https://nagasaki-bus.co.jp/bus/company/index.html

Nagasaki Bus Information Service 会社概要 (bedrijfsoverzicht). Opgevraagd op 16 februari 2016, van http://nbis.jp/index.php/company/about

Node-GTFS. *GitHub – Node-GTFS.* Opgevraagd op 18 januari 2016, van https://github.com/brendannee/node-gtfs

OpenStreetMap, *OpenStreetMap*, Opgevraagd op 18 januari 2016, van https://www.openstreetmap.org/

OpenTripPlanner. *OpenTripPlanner – Home*. Opgevraagd op 18 januari 2016, van http://www.opentripplanner.org/

OV-api. *OpenTripPlanner – Home*. Opgevraagd op 3 maart 2016, van http://gtfs.ovapi.nl/

11. BIJLAGEN

BIJLAGE A. ACCEPTATIETEST EINDPRODUCT (JAPANS)

Op de huidige routeplanner is een logger geplaatst die voor een aantal weken alle zoekopdrachten gelogt heeft. De acceptatietest bestaat uit een vijftigtal zoekopdrachten die daaruit willekeurig gekozen zijn. Deze zijn vervolgens vergeleken wanneer uitgevoerd op de huidige routeplanner, en wanneer uitgevoerd op de nieuw ontwikkelde routeplanner, en zijn de resultaten daarbij geanalyseerd.

De velden zijn: nummer, zoekopdracht, resultaat oud, resultaat nieuw, toelichting, en beoordeling. Het meerendeel van de gevonden resultaten zijn zelfs beter, dat wil zeggen meer dan wel efficiëntere reismogelijkheden d.m.v. het invoegen van een overstap of een gedeelte om te zetten naar een looproute. Alleen bij zoekopdracht (2) is het resultaat precies gelijk.

Het eindresultaat is goedgekeurd door de bedrijfsbegeleider en ondertekend.

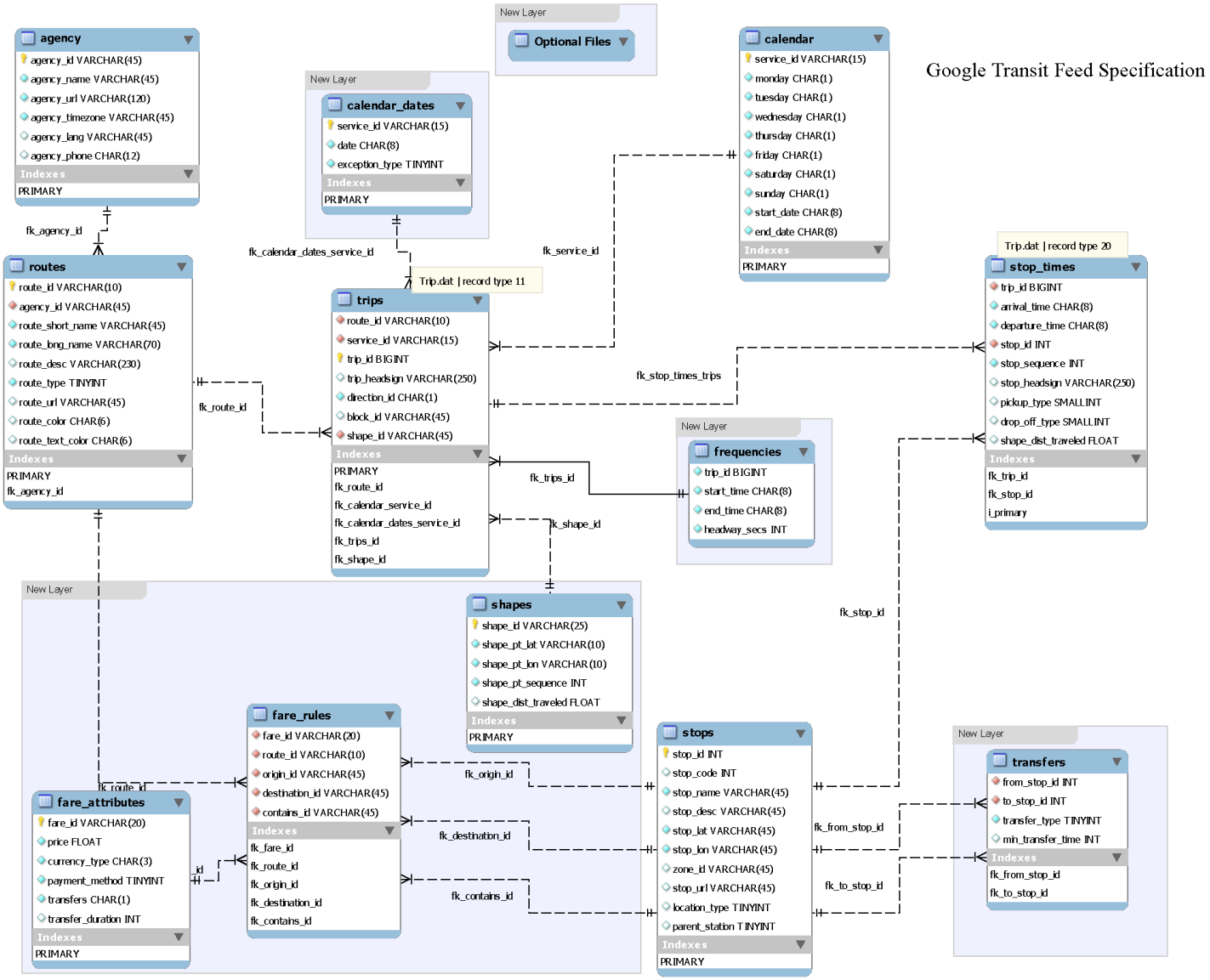
長崎バス ダイヤデータ検証

#	検索条件	時刻検索	乗り継ぎ検索	備考	判定
1	長崎駅前東口 番所橋 土曜 7 時 00 分 発	申し訳ございません、ご希望の経路はまだ登録されておりません。 もしくは乗り継ぎが必要です。	# 出発 到着 運賃 1 08:33 発 09:07 着 310 円 2 08:48 発 09:22 着 310 円 3 09:13 発 09:47 着 310 円 4 09:33 発 10:10 着 310 円	乗 継	より良
2	小瀬戸町 興善町 平日 14 時 10 分 発	1 14:11 発 14:38 着 260 円 2 14:21 発 14:48 着 260 円 3 14:31 発 14:58 着 260 円 4 14:41 発 15:08 着 260 円 5 14:51 発 15:18 着 260 円	1 14:11 発 14:38 着 260 円 2 14:21 発 14:48 着 260 円 3 14:31 発 14:58 着 260 円 4 14:41 発 15:08 着 260 円 5 14:51 発 15:18 着 260 円		良
3	池田 栄上 日曜祝日 15 時 00 分 発	# 出発 到着 運賃 1 16:58 発 17:08 着 160 円	# 出発 到着 運賃 1 15:51 発 16:08 着 150 円 2 16:26 発 16:43 着 150 円 **3 16:58 発 17:08 着 160 円** 4 17:36 発 17:53 着 150 円 5 18:21 発 18:38 着 150 円		より良
4	長崎駅前南口 馬場 日曜祝日 20 時 23 分 発	1 20:26 発 20:59 着 320 円 2 21:26 発 21:59 着 320 円 3 22:26 発 22:59 着 320 円	1 20:26 発 20:59 着 320 円 2 20:51 発 21:30 着 320 円 3 21:26 発 21:59 着 320 円 4 21:51 発 22:30 着 320 円		より良

			5 22:26 発 22:59 着 320 円		
5	長崎駅前 文教町 平日 18 時 37 分 発	1 18:53 発 19:08 着 160 円 2 19:36 発 19:48 着 190 円	1 18:53 発 19:08 着 160 円 2 18:43 発 19:11 着 190 円 3 18:43 発 19:11 着 190 円 4 18:51 発 19:20 着 190 円 5 19:11 発 19:38 着 190 円		より良
6	住吉 昭和町 平日 23 時 00 分 発	ご指定の時間の運行はございません。	1 (1) 22:58 発 23:13 着 0 円 2 23:03 発 23:24 着 150 円 3 23:06 発 23:27 着 150 円 4 23:15 発 23:36 着 150 円 5 23:17 発 23:38 着 150 円	(1)徒歩	より良
7	飽の浦神社前 住吉台口 日曜祝日 5 時 00 分 発	申し訳ございません、ご希望の経路はまだ登録されておりません。 もしくは乗り継ぎが必要です。	1 06:55 発 07:30 着 310 円 2 06:50 発 07:38 着 310 円 3 07:00 発 07:38 着 310 円 4 06:45 発 07:40 着 310 円 5 07:24 発 07:58 着 210 円		より良
8	ココウォーク茂里町 国立長崎病院 平日 10 時 20 分 発	申し訳ございません、ご希望の経路はまだ登録されておりません。 もしくは乗り継ぎが必要です。	1 10:18 発 10:52 着 320 円 2 10:20 発 11:01 着 360 円 3 10:25 発 11:01 着 360 円 4 10:20 発 11:02 着 400 円 5 10:20 発 11:02 着 400 円		より良

濱本 剛一

BIJLAGE B: RDBM GTFS



BIJLAGE C. PLAN VAN AANPAK

HOGESCHOOL UTRECHT

OV-routeplanner voor Nagasaki Bus

Plan van Aanpak

Luuk Nass 1574695
Versie: 1.0
Herzien: 15-12-2015

Inhoudsopgave

1. Inleiding

Dit document is een plan van aanpak. Deze beschrijft diverse onderwerpen die betrekking hebben op het afstuderen voor de opleiding Informatica. Deze onderwerpen zijn als volgt: de context van het afstudeerproject, de opdrachtomschrijving, een kernachtige formulering van de opdracht, een aantal inhoudelijke trefwoorden waarmee de opdracht wordt gekarakteriseerd, producten die naar verwachting worden opgeleverd, kennis vereist om de opdracht uit te voeren, een globale planning, relatie van de opdracht met de hoofdfase van de studie, de uitdaging, bedrijf- en persoonsgegevens, en bronvermeldingen.

2. Organisatie

Dit hoofdstuk bevat een beknopte beschrijving van de organisatie en de organisatiestructuur.

2.1 Algemeen

Nagasaki Bus Information Service (NBIS) is onderdeel van Nagasaki Bus Group, de overkoepelende organisatie van een aantal bedrijven met als belangrijkste lid Nagasaki Bus (opgericht in 1936, het busbedrijf van de stad Nagasaki, Japan); maar ook hotels als Nikko Huis Ten Bosch, winkelcentrum en busstation Mirai Nagasaki Cocowalk, een centrum voor verkeersveiligheid alsook een vastgoedtak, en nog enkele andere bedrijven.

De Information Service is begin 2013 officieel afgesplitst van Nagasaki Bus als apart bedrijf door recente ontwikkelingen op ICT-gebied, maar bevindt zich nog wel in hetzelfde gebouw. NBIS houdt zich voornamelijk bezig met het ontwikkelen, verkopen, onderhouden en introduceren van zowel software als hardware, consultancy en data-analyse en neemt steeds meer nieuwe projecten aan. Voorbeelden hiervan zijn de recent geïntroduceerde Nagasaki Smart Card; een “OV- chipkaart” voor de regio Nagasaki. Andere voorbeelden zijn beeldkranten voor advertentiedoeleinden en hardware-oplossingen als automatische omroepsystemen voor in de bus. Daarnaast bijvoorbeeld ook digitale vertrekstaten op busstations.

2.2 Interne organisatie

Nagasaki Bus Information Service heeft, hoewel afgesplitst, nog altijd een directe relatie met Nagasaki Bus. Nagasaki Bus heeft ongeveer 1000 medewerkers met een aantal verschillende directeuren op verschillende niveaus. Binnen NBIS zijn er momenteel tien medewerkers; één algemeen directeur, een twee afdelingshoofden ICT. Verder zijn er software-ontwikkelaars, systeembeheerders en de overige medewerkers zijn voor kantoorwerkzaamheden zoals verkoop en telefoonsupport. De voertaal is Japans.

2.3 Infrastructuur en ondersteuning

Op het hoofdkantoor van Nagasaki Bus is een kantoorwerkplek inclusief benodigde hardware beschikbaar gesteld, evenals de ontwikkelomgeving en eventuele server-hardware. Ook is er gezorgd voor een (werk)visum, verblijfplaats en een voldoende salaris om rond te kunnen komen.

3. Projectbeschrijving

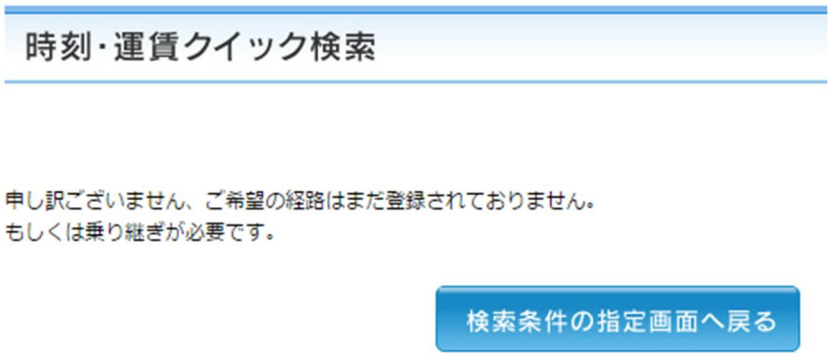
3.1 Aanleiding

De online routeplanner[1] van Nagasaki Bus is verouderd en biedt weinig moderne functionaliteit. Het is oorspronkelijk ontwikkeld in 2002, herzien in 2007, en verder enkel her en der wat onderhoudsbeurten. Het grootste probleem is dat deze routeplanner alleen kan zoeken binnen busroutes. Enkel in het geval de opstaphalte en de bestemming door dezelfde bus wordt aangedaan verschijnen er zoekresultaten. In figuur 1 hieronder wordt een dergelijke zoekopdracht gedaan.

Figuur 1: Een voorbeeldroute van Wakabamachi (若葉町) naar Nagasaki-shinchi-terminal (長崎新地ターミナル)

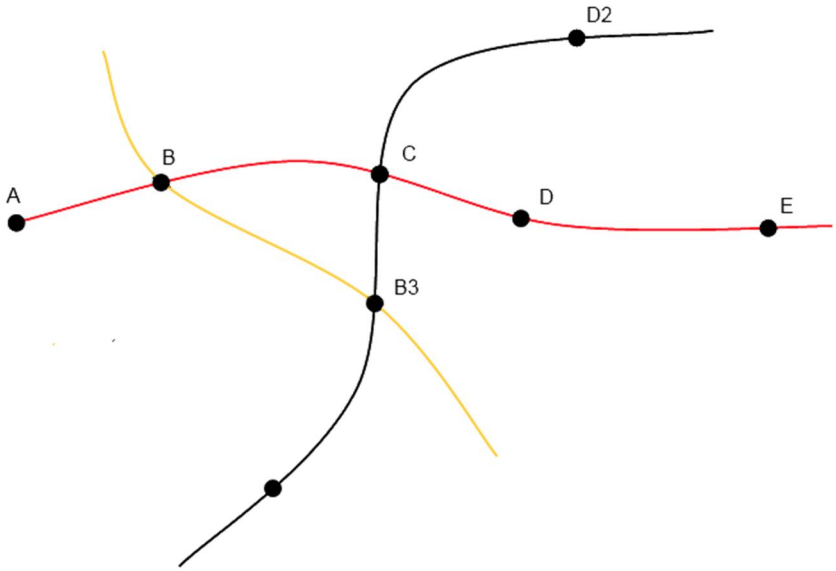
[1] http://www.nagasaki-bus.co.jp/dia-search/index.php (Japans) en de mobiele versie: http://www.nagasaki-bus.co.jp/k-tai/index.php (Japans)

Wanneer er gezocht wordt vanaf een bushalte die niet direct verbonden is met de eindbestemming verschijnt de volgende pagina:



Figuur 2: “Onze excuses, de door u gewenste reis is nog niet geregistreerd, ofwel mogelijk is er een overstap vereist.”

Routes zijn niet aan elkaar gekoppeld en men kan daarom geen informatie opvragen over diverse overstapmogelijkheden. Het schematisch overzicht hieronder laat dit zien.



Figuur 3: Schematisch voorbeeld

Een route van bushalte A naar bushalte E is geen probleem, maar van A naar B3 wel; dan is er geen beschikbare route. Een blik op het schema wijst uit dat er twee mogelijke routes zijn: (1) A > B > B3 en (2) A > C > B3. Welke hiervan de meest optimale is kan verschillen omdat bijvoorbeeld de gele buslijn veel minder frequent kan rijden dan de rode en zwarte buslijn.

3.2 Doelstelling en opdrachtformulering

Het doel van deze opdracht is om de online routeplanner te moderniseren, waarbij het mogelijk maken van overstappen binnen een zoekopdracht het belangrijkste onderdeel is. De bestaande webapplicatie is ontwikkeld met PHP5 i.c.m. MySQL. Er is een optie om daarin deze nieuwe functionaliteit te introduceren, ofwel om een nieuwe applicatie te ontwikkelen.

Naast het mogelijk maken van overstappen zijn er nog een aantal andere wensen. Voorbeelden hiervan zijn het niet alleen kunnen zoeken van bushalte naar bushalte, maar ook vanaf bepaalde landmarks, dat wil zeggen bepaalde locaties waar men veel heen reist zoals een treinstation, toeristische attractie of een ziekenhuis. Daarnaast is er de wens om een route niet alleen van overstap naar overstap op tekstbasis weer te geven. Op een online kaartensysteem als Google Maps zou dan de gehele route, of ten minste de opstap- en eindhalte weergegeven kunnen worden. Een andere wens is het weergeven van de prijs van de nieuwe gecombineerde route en als laatste is er nog de wens om de webapplicatie ook compatibel te maken met mobiele apparaten zoals smartphones. Hierbij kan naast het optimaliseren van de user inferface ook gedacht worden aan het gebruik van zaken als automatische plaatsbepaling en vanuit die ingang een zoekopdracht uit te voeren.

Hieronder een tabel met MoSCoW-prioriteiten.

#	Naam	Prioriteit
1	Mogelijkheid tot overstappen	Must Have
2	Zoeken van en/of naar landmarks	Could Have
3	Visuele weergave van (een deel van) de zoekresultaten	Could Have
4	Prijsbereking van een route met overstap	Could Have
5	Website met ondersteuning mobiele apparaten	Could Have

Van deze onderdelen is alleen het introduceren van de mogelijkheid tot het vinden van routes met een mogelijkheid tot overstappen een requirement, de rest zijn nice to haves.

De roep om een handigere, gebruiksvriendelijkere routeplanner begint steeds dringender te worden. Nagasaki Bus Information Service wil als uiteindelijke doelstelling deze kans aangrijpen om klanten beter van dienst te zijn met een meer geavanceerde routeplanner die beter aansluit op wat klanten tegenwoordig verwachten. Tevens is dit een kans om intern vernieuwingen door te voeren. De opdracht is dus om bestaande software te verbeteren door nieuwe software of functionaliteit te introduceren om deze te vervangen of te verbeteren.

3.3 Trefwoorden

PHP—MySQL—OV-routeplanner—Zoekalgoritmen—Mobiele Integratie—Online kaartsystemen

3.4 Theoretisch kader

Hieronder wordt het theoretisch kader bij de opdrachtformulering uiteengezet.

Betreft de routeplanner zijn er een drietal te onderscheiden onderdelen:

(1) Data: de dataopslag (data-laag)
(2) Zoeken: het zoeken naar routes (business logica)
(3) Weergeven: de presentatie aan de gebruiker (user interface (UI))

De functionaliteit voor het kunnen overstappen (2) is het belangrijkste, hier zal daarom ook de meeste aandacht aan besteed worden. De overige vier wensen, het kunnen zoeken vanaf andere locaties dan bushaltes, het weergeven van een route op een online kaart, het berekenen van het tarief en het ondersteunen van smartphones zullen mogelijk niet, of beperkt in dit project worden opgenomen.

Een oriëntatie op het type data leert dat, hoewel een standaard SQL-benadering zoals in de huidige routeplanner technisch mogelijk is, een graph-database als Neo4J in dit soort situaties sneller en efficiënter resultaten kan leveren.[2] In het geval van een SQL-implementatie zal in theorie, optimalisaties zoals het vooraf opdelen in geografische groepen niet meegenomen, voor alle mogelijke routes tussen alle haltes berekend moeten worden wat de 'kosten' in afstand en tijd zijn. Hiermee moet dan worden bepaald welke route daarvan de snelste of beste is.

De schaal van de data is hierbij belangrijk. Het busnetwerk van Nagasaki Bus heeft ongeveer 1100 bushaltes waarvan er in de huidige dienstregeling 800 ook daadwerkelijk aangedaan worden. Per week zijn er ongeveer 12,000 ritten verdeeld over bijna 1000 verschillende buslijnen van gemiddeld een half uur per rit. Vanaf elk van deze bushaltes naar elk van de andere bushaltes zal dus vooraf de route berekend moeten worden.

[2] http://neo4j.com/why-graph-databases/

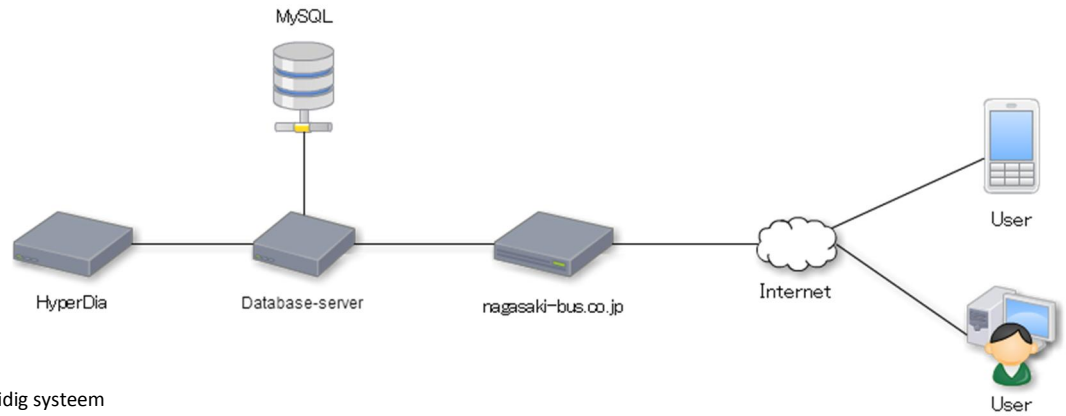
Hier zitten een aantal grote nadelen aan.

- De snelste route is niet voor elk moment van de dag de snelste route; sommige bussen rijden bijvoorbeeld vaker in de spits, of soms helemaal niet
- De gehele dataset moet opnieuw worden naberekend als er bij een van de busritten of routes iets veranderd: de data is niet dynamisch
  - Het implementeren van uitbreidingen zoals realtime vertrektijden is daarom niet mogelijk
  - In het geval er naar de eindbestemming meer dan 1 route mogelijk is zal vooraf bepaald moeten worden welke daarvan gekozen wordt

Het is dus wenselijk om dit probleem op een andere manier op te lossen, waarbij de data wel dynamisch te bevragen is. Vooronderzoek wijst uit dat er een wereldwijd veelgebruikte standaard is voor het opslaan en gebruiken van OV-informatie: GTFS (General Transfit Feed Specification)[3]. GTFS is een collectie .csv bestanden waarin volgens een vaste standaard een OV-systeem beschreven wordt. Het grote voordeel hiervan is dat bestaande dynamische routeplanners zoals Google Maps, maar ook open source routeplanners als OpenTripPlanner[4] deze data kunnen inlezen en ook direct compatibel zijn.

Een nadeel is dat GTFS niet alle mogelijke configuraties van systemen kan bevatten. In Japan is het bijvoorbeeld gebruikelijk om een rit niet te voorzien van een lijnnummer. In plaats daarvan wordt met een reeks via-bestemmingen de route uiteengezet. Bijvoorbeeld eindbestemming B via wijk A en wijk D. Met een omweg is dit wel te implementeren, maar een universele routeplanner interpreteert deze data dan mogelijk onjuist, of heeft daar in de user interface geen rekening mee gehouden.

Bij Nagasaki Bus wordt er voor het beheer van het busnetwerk en de buslijnen/ritten gebruikt gemaakt van een commerciële beheertool van Hitachi genaamd SmartDia[5]. Deze data wordt gebruik in de huidige routeplanner, en door een externe tool in een MySQL database ingelezen. De routeplanner roept deze vervolgens aan (zie figuur 4). De tarieven voor de ritten komen vervolgens uit een ander fabrikantafhankelijk systeem welke op een vergelijkbare manier wordt geïmporteerd.



Figuur 4: huidig systeem

[3] https://developers.google.com/transit/gtfs/?hl=nl

[4] http://www.opentripplanner.org/

[5] http://www.hitachi-systems.com/solution/s006/smartdia/

Een analyse van de huidige webapplicatie wijst uit dat er weinig tot geen rekening is gehouden met een goede software-architectuur; er is geen scheiding tussen de user interface (UI) en business logica, en ook is er geen database-abstractielaag. Er is geen mogelijkheid om te wisselen van UI, de domeinlogica is niet herbruikbaar, en de database is niet uitwisselbaar. Daarnaast is de webapplicatie regelmatig herzien, bijvoorbeeld bugfixes en kleine nieuwe functies, met maar beperkte toelichting in de programmacode. De code staat ook vol met oude weggecommente code en is er in geen enkele vorm sprake van Clean Code. Er is documentatie aanwezig, maar dit zijn toelichtingen op de datastructuur, de werking van de omzettool en testplannen, maar niets over de code. Veel hiervan komt wellicht door het feit dat de applicatie in 2002 ontwikkeld is, maar ook de gebruikte programmeertaal PHP maakt het organiseren van de architectuur niet makkelijker. Dit alles maakt het werken met dit systeem erg lastig, en is het, ook vanuit een architectureel standpunt waarschijnlijk verstandiger om een nieuwe applicatie te ontwikkelen.

In overleg met de bedrijfsbegeleider is in een vroeg stadium besloten dat het, voornamelijk in het kader van onderhoudbaarheid en uitbreidbaarheid verstandig is om de data uit SmartDia en het tarievensysteem compatibel te maken met GTFS. Afgesproken is hier dat er een tool moet worden ontwikkeld die de data uit SmartDia inleest, valideert, en omzet naar GTFS. Zodra deze data in GTFS beschikbaar is, is het in de toekomst mogelijk om makkelijker extensies toe te voegen zoals bijvoorbeeld realtime vertrekdata met bijvoorbeeld de RT(Real Time)-extentie. Tegelijkertijd kunnen de huidige vertrouwde beheertools gebruikt blijven worden en is er geen introductie van een andere software noodzakelijk.

Door het gebruik van GTFS is het tevens veel makkelijker om een bestaande oplossing voor het vinden van routes te implementeren en deze aan te roepen via een API. Dat betekent dat in deze vorm de data-laag (GTFS), de business logica (de API) los komen te staan van de UI (loose coupling) waardoor onderhoudbaarheid en portabiliteit vergroot worden. Door de verschillende softwarelagen los te koppelen is er ook de mogelijkheid om de webapplicatie, mobiele- en/of smartphone-versie buiten de scope van dit project te plaatsen gezien de onderliggende data gelijk blijft.

3.5 Hoofdvraag en deelvragen

Hoe kan voor Nagasaki Bus een betere, modernere OV-routeplanner worden gerealiseerd?
Onder beter en moderner wordt verstaan de lijst requirements en nice to haves uit hoofdstuk 3.2

(1) Hoe werkt de huidige routeplanner?
(2) Welke bestaande softwarepakketten voor routeplanners zijn er beschikbaar, zowel open-source als op de markt?
(3) Hoe is de dataopslag in het huidige systeem geregeld, en is dit te koppelen met andere, eventueel externe routeplanners?
(4) In hoeverre is de bestaande webapplicatie bruikbaar voor een softwareupgrade?
(5) Hoe kunnen landmarks worden geïntegreerd in een routeplanner?
(6) Welke mogelijkheden zijn er om een gevonden route weer te geven op een online kaartensysteem?
(7) Welke technieken en softwareomgevingen zijn geschikt om een nieuwe webapplicatie met nieuwe routerplanner te ontwikkelen?
(8) Welke mogelijkheden zijn er voor het berekenen van het tarief voor de gevonden gecombineerde route?

3.6 Onderzoeksmethoden

Om onderzoek uit te voeren is het bepalen van de onderzoeksmethode een belangrijke keuze. Per deelvraag zal worden aangeven hoe dat onderzoek uitgevoerd zal worden.

#	Functie	Methode
1	Beschrijvend	Observationeel
2	Beschrijvend, Vergelijkend	Literatuuronderzoek, experiment
3	Beschrijvend	Observationeel, literatuuronderzoek
4	Beschrijvend	Observationeel
5	Vergelijkend, Beschrijvend	Literatuuronderzoek, experiment
6	Vergelijkend	Literatuuronderzoek, experiment
7	Vergelijkend, Beschrijvend	Literatuuronderzoek, eventueel experiment
8	Vergelijkend	Literatuuronderzoek, experiment

Voor een aantal van de deelvragen, met name (1) tot en met (4) is reeds vooronderzoek gedaan ten tijde van het Plan van Aanpak, en zal dit verder worden uitgebreid. Voor deelvraag (5) en (6), het vinden van routes van/naar landmarks, en weergeven van de gehele gevonden zal er waarschijnlijk een externe navigatie API moeten worden aangeroepen, bijvoorbeeld Google Maps of OpenStreetMaps. Voor deelvraag (7), welke ontwikkelsoftware, zal een afweging moeten worden gemaakt wat hiervoor de beste oplossing is en aan de andere kant ook met het oog op onderhoudbaarheid, de gebruikte omgeving moet wel bekend zijn, of in een korte tijd te introduceren zijn bij het team voor toekomstig onderhoud. Voor het berekenen van het tarief (8) is de werking van het huidige systeem de bepalende factor.

3.7 Vereiste kennis

Voor dit project is vakkennis nodig voor de volgende onderdelen:

- OV-systemen, GTFS
  - Het bedrijf heeft ruim 80 jaar ervaring met OV-systemen
  - Er is kennis over GTFS, en de werking daarvan
- Software-architectuur
  - Binnen het bedrijf niet aanwezig, maar de student heeft dit vanuit de opleiding meegekregen
- Databases, in het bijzonder Graph-databases
  - Binnen het bedrijf niet tot minimaal aanwezig, maar de student heeft dit vanuit de opleiding meegekregen
- Universele koppelingen met externe systemen
  - Het koppelen en omzetten van fabrikantafhankelijke systemen
    - Binnen het bedrijf is er voldoende expertise over deze systemen gezien de jarenlange ervaring daarmee
- Web development en mobiele websites
  - Zowel bij het bedrijf als bij de student is kennis aanwezig

3.8 Producten

Tijdens het afstudeerproject worden voortdurend producten gemaakt. Aan het einde van het project worden de volgende producten opgeleverd:

Product	**Criteria, goedgekeurd wanneer:**
Plan van Aanpak	Goedgekeurd en ondertekend door - student - bedrijfsbegeleider - docentbegeleider
Onderzoeksrapport	Er dient gedegen onderzoek te worden gedaan, de resultaten moeten een voldoende en bruikbaar antwoord geven op de deelvragen, en daarmee uiteindelijk antwoord op de hoofdvraag
Eindproduct, eventueel beperkt tot proof of concept	- Voldoet minimaal aan de requirements (het kunnen overstappen) - Nice to haves (landmarks, weergave route, prijsberekeningen) zijn optioneel
Scriptie	- twee maal inleveren van een conceptversie - voldoet aan de eisen zoals gesteld in de afstudeerleidraad

Deze producten worden ontwikkeld aan de hand van de planning in hoofdstuk 4.

4. Globale Planning

Hieronder staat een globale planning van de te verrichten werkzaamheden met de tijdsverdeling en eventuele inleverdata.

Activiteit	**Startdatum**	**Einddatum**	**Deadline**
Fase -1			
- Inventariseren van wensen en eisen	16 november	24 november	—
- Vooronderzoek	19 november	4 december	—
- Simpel prototype	7 december	14 december	—
- Plan van aanpak	24 november	14 december	15 december
- Afstudeercontract	—	—	15 december
Fase-2			
- Functioneel ontwerp (use cases)	16 december	22 december	—
- Architectuurdocument (technisch ontwerp)	23 december	30 december	—
- Ontwikkeling	4 januari	19 februari	—
- Testen	22 februari	26 februari	—
Fase 3			
- Acceptatietesten	29 februari	4 maart	—
- Verwerken feedback, uitloop	7 maart	15 maart	—
Scriptie, conceptversie 1	21 december	—	29 januari
Scriptie, conceptversie 2	—	—	26 februari
Scriptie, definitieve versie	—	—	15 maart
Schriftelijke beoordeling	—	—	15 maart
Voorbereiding afstudeerzitting	21 maart	25 maart	—
Afstudeerzitting	29 - 31 maart	—	—

5. Relatie Project met studie

De relatie tussen het project en de gevolgde studie is het volgende:

1. Analyse van een bestaand softwaresysteem
2. Requirements verzamelen en tot een degelijk ontwerp komen
3. Functioneel- en technisch ontwerp, m.n. softwarearchitectuur
4. Zelfstandig opgedane kennis toepassen in een systeem

Diverse onderdelen zijn tijdens de studie zo realistisch mogelijk toegepast, maar nooit is er echt druk vanuit het werkveld geweest, en nooit is een product daadwerkelijk live gegaan.

6. Risico's en Uitdagingen

Bij deze opdracht zijn er verschillende uitdagingen:

- Werken in een compleet verschillende bedrijfscultuur
- Dagelijks communiceren in het Japans, een taal die op geen enkele manier lijkt op Nederlands, Engels of andere westerse talen
- Leren werken met de bestaande productieomgeving
- De schaal van het eindproduct, er zijn een groot aantal buslijnen, ritten, en passagiers; er is veel data waarbij efficiëntie essentieel is
- Het eindproduct wordt daadwerkelijk gebruikt en zal het bedrijfsimago schaden wanneer er onnodige fouten in de software zitten
- Mogelijke toekomstige koppeling met andere route-informatiesystemen

Tevens zijn er een aantal risico's. Hieronder wordt ook de te ondernemen actie aangegeven.

#	**Risico**	**Actie**
1	Planning loopt uit	Planning bijstellen, eventueel requirements aanpassen of nice to haves schrappen
2	Technische kennis ontbreekt	Bij het bedrijf informeren voor mogelijkheden tot bijscholing door collega's of informeren naar mogelijkheden voor cursussen, of planning aanpassen voor tijd voor zelfstudie
3	Wijziging in requirements	De inpact onderzoeken en de planning bijstellen, eventueel andere requirements ook aanpassen. Docentbegeleider inlichten.
4	Eindproduct voldoet niet aan de eisen	In overleg met bedrijfsbegeleider de te ondernemen acties bespreken, eventueel requirements bijstellen.
6	Geld en/of middelen ontbreken	Overleggen met de bedrijfsbegeleider wanneer dit opgelost zal worden, eventueel verantwoordelijkheid van de student weghalen, direct docentbegeleider inlichten.

7. Bedrijfsbegeleider en communicatie

De bedrijfsbegeleider is Koichi Hamamoto is:

- k.hamamoto@nbis.jp
- T +81 95 826 1119
- M +81 (0) 80 2737 3238

Mijn persoonlijk begeleider is Shuichi Kashiyama:

- s.kashiyama@nagasaki-bus.jp
- T +81 95 826 1116
- M +81 (0) 80 2737 221

Mocht er zich een probleem voordoen zal ik mij ten eerste richten tot de bedrijfsbegeleider. Als het om criteria van het eindproduct gaat is in overleg altijd het een of ander aan te passen. Mocht het een cultuurprobleem of een probleem in benodigde middelen zijn is er nog mijn persoonlijke begeleider.

Iedere ochtend is er een bijeenkomst waarbij alle medewerkes hun taken voor die dag vertellen aan de rest van het bedrijf, en daarmee is direct de voortgang duidelijk voor anderen. Daarnaast is de ontwikkelafdeling nauw betrokken bij het project, en zijn er dagelijks werkbesprekingen. Uitgebreide voortgangsbesprekingen zullen ongeveer 1 a 2 keer per week plaatsvinden, waarbij ook waar nodig de koers bijgesteld zal worden.

8. Bedrijfsinformatie

Nagasaki Bus Information Service

3-17 Shinchi-machi, Nagasaki City, Nagasaki Prefecture, Japan

+81 95 826 1119

http://nbis.jp/

9. Bronnen

GTFS – uitleg over de standaard
https://developers.google.com/transit/gtfs/?hl=nl
geraadpleegd op 16-11-2015

General Transit Feed Specification
http://www.transitwiki.org/TransitWiki/index.php?title=General_Transit_Feed_Specification
geraadpleegd op 30-11-2015

HyperDia クラウド型ダイヤ作成システム (HyperDia: een cloud-based beheertool voor dienstregelingen)
http://www.hitachi-systems.com/solution/s006/smartdia/
geraadpleegd op 14-12-2015

Transit schedule data demystified - using GTFS – extra toelichting over de werking
http://cheesehead-techblog.blogspot.jp/2014/01/transit-schedule-data-demystified-using.html
geraadpleegd op 29-11-2015

Why Graph Databases?
http://neo4j.com/why-graph-databases/
geraadpleegd op 10-12-2015

時刻・運賃クイック検索 長崎バス (Snel zoeken naar tijden en prijzen – Nagasaki Bus)
http://www.nagasaki-bus.co.jp/dia-search/index.php (Japans)
geraadpleegd op 29-11-2015

長崎バスモバイル (Nagasaki Bus Mobiel)
http://www.nagasaki-bus.co.jp/k-tai/index.php (Japans)
geraadpleegd op 29-11-2015