Data Analysis Portal

Bijlagen

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Lead	Tim van Egmond
Opdrachtgever	CityTraffic
Plaats	Alphen aan den Rijn
Datum	1 Juni 2017
Versie	1.0
Bijlagen	A. Tim van Egmond afstudplan_2017-1.1 B. Plan van Aanpak C. Voortgangsverslag D. Onderzoeksrapport SPA E. Onderzoeksrapport Akka.NET F. Requirements Discipline G. Design Discipline H. Testrapport I. Evaluatieformulier afstuderen

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2017-1.1 (start uiterlijk 6 februari 2017)
Startdatum uitvoering afstudeeropdracht: 6 februari 2017
Inleverdatum afstudeerdossier volgens jaarrooster: 2 juni 2017

Studentnummer: 13078550
Achternaam: dhr van Egmond
Voorletters: T.
Roepnaam: Tim
Adres: Steenbokstraat 51
Postcode: 2394 PE

Woonplaats: Hazerswoude Rijndijk
Telefoonnummer:
Mobiel nummer: (+316) 22327974
Privé emailadres: timvngmnd@gmail.com

Opleiding: Informatica
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: Anneke Wieman
Naam begeleidend examinator: Gerda in 't Veld
Naam tweede examinator: Cobie van der Hoek

Naam bedrijf: CityTraffic
Afdeling bedrijf:
Bezoekadres bedrijf: De Schans 23
Postcode bezoekadres: 2405 XX
Postbusnummer: 2065
Postcode postbusnummer: 2400 CB
Plaats: Alphen aan den Rijn
Telefoon bedrijf: (+31172) 435901
Telefax bedrijf: (+31172) 422271
Internetsite bedrijf: http://citytraffic.nl/

Achternaam opdrachtgever: dhr. Schmitz
Voorletters opdrachtgever: B.
Titulatuur opdrachtgever:
Functie opdrachtgever: Directeur
Doorkiesnummer opdrachtgever:
Email opdrachtgever: bart.schmitz@pfm-intelligence.com

Achternaam bedrijfsmentor: dhr. Floor
Voorletters bedrijfsmentor: C.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Software Developer
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: christiaan.floor@pfm-intelligence.com

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Ontwikkelen van een data-analysesysteem bij CityTraffic

Opdrachtomschrijving

1. Bedrijf

CityTraffic is een jong en innovatief bedrijf dat in de afgelopen jaren in meer dan 120 Nederlandse en Belgische steden een netwerk van automatische sensoren aangelegd heeft, om volcontinu de passantenstromen in winkelstraten mee te meten. Gemeentes en steden willen graag weten wat de drukke punten in hun winkelcentra(s) zijn, om zo bijv. beter de huurprijzen van winkelpanden te kunnen bepalen. CityTraffic is onderdeel van het moederbedrijf PFM-Intelligence. PFM beschikt over 50+ medewerkers, verspreid over Nederland, België, Duitsland en Groot-Brittannië.

Telsystemen op basis van camera metingen genereren tellingen in de vorm van CSV-bestanden. Deze bestanden hebben de totaal tellingen van een camera. De sensoren die deze tellingen realiseren variëren van beeld-, infrarood- tot laser camera's. Naast deze sensoren wordt ook gebruik gemaakt van zogenaamde hotspots. Deze hotspots scannen de wifi en bluetooth mac-adressen van passanten. De data, die de sensoren genereren, wordt nu in een database verwerkt tot rapportages. Indien een telsysteem niet gefunctioneerd heeft is dit terug te zien in de rapportage, omdat er een data reeks ontbreekt. CityTraffic beschikt op dit moment over vijf werknemers, wiens werk het is deze data te corrigeren.

De IT-afdeling maakt gebruik van Jira en git.

2. Probleemstelling

De verschillende vestigingen van PFM-Intelligence werken met verschillende softwarepakketten, de directeur wil graag dat iedereen met één standaardpakket werkt. Deze afstudeeropdracht zal zich richten op één onderdeel van dit softwarepakket, namelijk het datacorrectie gedeelte. Dit gedeelte heeft de hoogste prioriteit om gecreëerd te worden, omdat de datacorrectie op dit moment veel manueel werk vereist. Het is de wens dat dit zo veel mogelijk automatisch gebeurt.

3. Doelstelling van de afstudeeropdracht

CityTraffic wil een oplossing voor de volgende vraagstukken:

1. Hoe kunnen we TEL-data en wifidata correcties ondervangen in één programma, wat wel beide datastromen controleert en daar waar nodig repareert?

2. Hoe maak je het detectie algoritme dusdanig robuust dat dit automatisch leidt tot een zeer betrouwbaar filter, waarmee de datamanager (werknemer) overbodige controles voorkomt, maar anderzijds zeker geen data afwijkingen/ onderbrekingen over het hoofd ziet?

3. De reactiesnelheid van deze te creëren software zal goed moeten zijn voor het op operationeel niveau te laten slagen. De hoeveelheid data van met name de wifisensoren is substantieel en zal oplopen tot miljoenen records per dag, per locatie. Hoe wordt straks de snelheid en schaalbaarheid in het oog gehouden en wat zijn hierin de problemen waar het design rekening dient te houden?

Om het probleem op te lossen moet een webapplicatie gemaakt worden, waar de data in gecorrigeerd kan worden, dat niet door het algoritme opgelost kan worden. Deze webapplicatie zal gemaakt worden in C#. CityTraffic wil dat de UI gemaakt wordt m.b.v. een framework. Er moet onderzocht worden wat de mogelijkheden hiervoor zijn.

Ook zal onderzocht moeten worden hoe de applicatie schaalbaar gemaakt kan worden, zodat de applicatie om kan gaan met de hoeveelheid data dat de applicatie te verwerken krijgt. Hiervoor zal onder andere gekeken worden naar frameworks als Akka.NET en Orleans. Deze frameworks lossen dit probleem mogelijk op door gebruik te maken van het actor model.

Voor de webapplicatie is het van belang dat deze uitbreidbaar is met de ondersteuning van andere bedrijfsprocessen. De webapplicatie zal een belangrijk bedrijfsproces ondersteunen, welke omgaat met gevoelige informatie, zoals gescande mac-adressen. Hierdoor is het van belang dat er rekening gehouden wordt met de beveiliging.

4. Resultaat

Nadat de opdracht is uitgevoerd zal een werkende demo van de webapplicatie met de datacorrectie functie opgeleverd worden. Als besloten wordt dit project voort te zetten of in gebruik te nemen, zal het, het werk van de medewerkers van CityTraffic verlichten.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Afstudeeropdracht

1. Het opstellen van een plan van aanpak (3 dagen)
2. Onderzoek naar een framework voor de UI (5 dagen)
3. Onderzoek hoe de applicatie om kan gaan met de hoeveelheid data dat de camera's en hotspots genereren. Kan Orleans of Akka.NET hiervoor gebruikt worden? (5 dagen)
4. Het opstellen van requirements, in bespreking met de stakeholders (8 dagen)
5. Het ontwerpen en bouwen van de software (34 dagen)
6. Het testen van de software (15 dagen)

Afstudeerdossier

1. Eindverslag (15 dagen)

6. Op te leveren (tussen)producten

- Plan van aanpak
- Onderzoeksrapporten
- Use cases
- Requirements
- Ontwerp diagrammen
- Testontwerp
- Testrapport

7. Te demonstreren competenties en wijze waarop

1.4 Uitvoeren analyse door definitie van requirements (niveau 3)

Voordat er aan de webapplicatie gewerkt kan worden, zal door middel van interviews, requirements opgesteld worden. Voor de webapplicatie zijn i.i.g. vijf stakeholders, die naar verwachting tegengestelde wensen hebben betreffende de userinterface. Nadat de requirements opgesteld zijn zullen deze gesorteerd worden op prioriteit. Vervolgens worden de requirements getest op volledigheid, duidelijkheid en consistentie. Het product dat hieruit volgt is een use-case diagram en beschrijving. Dit zal alleen van toepassing zijn, als het toegevoegde waarde biedt.

3.2 Ontwerpen systeemdeel (niveau 3)

Voor de webapplicatie zal een klassendiagram gemaakt worden. Het klassendiagram wordt gemaakt naar aanleiding van de requirements. Er zal een sequensdiagram gemaakt worden voor het algoritme dat de data corrigeert. Tijdens het ontwerp zal gebruik gemaakt worden van design patterns, om te zorgen dat het goed uitbreidbaar is. Ook wordt rekening gehouden met de beveiliging. Nadat het klassendiagram opgesteld is zal aan de bouw van de applicatie begonnen worden.

3.3 Bouwen applicatie (niveau 4)

De webapplicatie zal gemaakt worden in C# en maakt gebruik van frameworks, welke frameworks moet blijken uit het vooronderzoek. Voor versiebeheer zal git gebruikt worden. Verder maakt CityTraffic gebruik van een ontwikkel-, test- en productieomgeving.

3.5 Uitvoeren van en rapporteren over het testproces (niveau 3)

De applicatie ondersteunt een belangrijk bedrijfsproces, dus zal het goed getest moeten worden. De code zal getest worden met behulp van unit testen. Naast de code is het voor de applicatie van belang dat de performance getest wordt. Voor de tests zullen testontwerpen opgesteld moeten worden. Voor de testscripts is het van belang dat het in de toekomst hergebruikt kan worden, zodra ervoor gekozen wordt andere bedrijfsprocessen in de applicatie te creëren. De resultaten van de uitgevoerde tests zullen in het testrapport

Ontwikkelen van een data-analysesysteem

Plan van Aanpak

Dit document is bedoeld om de lezer inzicht te geven op het data-analyse project. Dit document beschrijft de opdracht, de aanpak en geeft een globale planning.

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Leider	Tim van Egmond
Opdrachtgever	Citytraffic
Plaats	Alphen aan den Rijn
Datum	10 februari 2017
Versie	1.0

Inhoudsopgaven

1. Inleiding

Dit plan van aanpak is opgesteld in opdracht van CityTraffic. Het geeft de lezer inzicht in het Data Analysis Portal project en beschrijft hoe dit traject zal lopen. Hierbij zal onder andere ingegaan worden op de opdrachtomschrijving, de scope en risico's, de ontwikkeling en de uit te voeren onderzoeken. Als laatst zal een globale planning geven worden van de uit te voeren activiteiten.

2. 1. Opdrachtomschrijving

2.1. 1.1. Aanleiding

Telsystemen op basis van camera metingen genereren tellingen in de vorm van CSV-bestanden. Deze bestanden bevatten de totaal tellingen van een camera. De sensoren die deze tellingen realiseren variëren van beeld-, infrarood- tot laser sensoren. Naast deze sensoren wordt ook gebruik gemaakt van zogenaamde hotspots. Deze hotspots scannen de wifi en bluetooth mac-adressen van passanten. De data, die de sensoren genereren, wordt nu in een database verwerkt tot rapportages. Indien een telsysteem niet gefunctioneerd heeft is dit terug te zien in de rapportage, omdat er een data reeks ontbreekt.

2.2. 1.2. Probleemstelling

De verschillende vestigingen van PFM-Intelligence werken met verschillende softwarepakketten, de directeur wil graag dat iedereen met één standaardpakket werkt, om de datacorrecties uit te voeren. In Nederland wordt met behulp van PFM totaal en Advantage II gewerkt en in Engeland met Advantage I. PFM totaal is opgesplitst in twee tools, één voor wifi en één voor tel data.

2.3. 1.3. Doelstelling

CityTraffic wil een oplossing voor de volgende vraagstukken:

1. Hoe kunnen we TEL-data en wifidata correcties ondervangen in één programma, wat beide datastromen controleert en daar waar nodig repareert? Deze acties worden niet automatisch uitgevoerd.

2. Hoe maak je het detectie algoritme dusdanig robuust dat dit automatisch leidt tot een zeer betrouwbaar filter, waarmee de datamanager (werknemer) overbodige controles voorkomt, maar anderzijds zeker geen data afwijkingen/onderbrekingen over het hoofd ziet?

3. De reactiesnelheid van deze te creëren software zal goed moeten zijn om het op operationeel niveau te laten slagen. De hoeveelheid data van met name de wifisensoren is substantieel en zal oplopen tot miljoenen records per dag, per locatie. Hoe wordt straks de snelheid en schaalbaarheid in het oog gehouden? En wat zijn op dit gebied de problemen waar het design rekening mee dient te houden?

Om het probleem op te lossen moet een webapplicatie gemaakt worden, waar de data in gecorrigeerd kan worden. Deze pagina zal beschikken over een functie welke een suggestie maakt aan de gebruiker voor reparatie mogelijkheden. CityTraffic wil dat de UI gemaakt wordt als een single page applicatie. Er moet onderzocht worden wat de mogelijkheden hiervoor zijn.

Ook zal onderzocht moeten worden hoe de applicatie schaalbaar gemaakt kan worden, zodat de applicatie om kan gaan met de hoeveelheid data dat de applicatie te verwerken krijgt. Hiervoor zal gekeken worden naar frameworks als Akka.NET en Orleans. Deze frameworks lossen dit probleem mogelijk op door gebruik te maken van het actor model.

Voor de webapplicatie is het van belang dat deze uit te breiden is met de ondersteuning van andere bedrijfsprocessen. De webapplicatie zal een belangrijk bedrijfsproces ondersteunen, welke omgaat met gevoelige informatie, zoals gescande mac-adressen. Hierdoor is het van belang dat er rekening gehouden wordt met de beveiliging.

De webapplicatie moet gemaakt worden met de nieuwste standaarden en dient gebruiksvriendelijk te zijn op de volgende display types: Desktop PC (16:9 & 21:9), tablet en smartphones.

3. 2. Project

3.1. 2.1. Scope

Deze opdracht zal zich richten op één aspect van het uiteindelijke softwarepakket, namelijk het datacorrectie en detectie gedeelte. Dit gedeelte heeft de hoogste prioriteit om gecreëerd te worden, omdat de datacorrectie op dit moment veel manueel werk vereist.

Op dit moment is het nog niet bekent of de webapplicatie een eigen database krijgt, of wat de structuur hiervan zal worden. Dit project zal in eerste instantie gebruik maken van de database van Advantage II. In het ontwerp dient rekening gehouden te worden met de mogelijke wijziging van de database. Het ontwerpen van de nieuwe database behoord niet tot de scope van dit project.

3.2. 2.2. Risico's

Het grootste risico is de beveiliging van de webserver. Doordat de webapplicatie in aanraking komt met gevoelige gegevens is het van uiterst belang dat deze gegevens niet in de verkeerde handen belanden. Ook mogen alleen geselecteerde gebruikers bij de gegevens komen. Om dit risico zo goed mogelijk te tackelen zal een plan opgesteld worden, welke beschrijft hoe zowel de front-end als backend omgaat met de beveiliging. Hierin zal ook ingegaan moeten worden op de auditing, autorisatie en authenticatie. Indien de stakeholders dit probleem op een later moment willen tackelen, zal dit buiten de scope van het project vallen. Wel zal in de toekomst goed naar dit probleem gekeken moeten worden. Het product kan NIET live gaan voordat dit uitgewerkt is.

Een ander groot risico is de performance van de webserver. De server zal te maken krijgen met de verwerking van grote hoeveelheid data. In de toekomst zal de webserver ook uitgebreid worden met andere bedrijfsprocessen, waardoor het verkeer naar de server kan stijgen. Tijdens de ontwikkeling dient rekening gehouden te worden met de schaalbaarheid van de applicatie. Dit risico kan mogelijk verholpen worden met het gebruik van Akka.NET. Mocht dit niet het geval zijn zal naar andere oplossingen gezocht moeten worden.

Het detectie/reparatie algoritme is nog niet bekend en moet nog worden onderzocht. Dit kan een grote impact hebben op de database dat gebruikt zal worden. Zodra het algoritme bekend is zal onderzocht moeten worden wat voor impact dit zal hebben op de gebruikte database(s).

3.3. 2.3. Stakeholders

De stakeholders van dit project zijn de data reparatie werknemers van CityTraffic, zij zullen met dit programma werken. De directeur zal altijd het laatste oordeel behouden als er conflicterende wensen zijn. Voor technische aspecten zal de projectleider, in overleg met andere ontwikkelaars, het laatste oordeel geven.

3.4. 2.4. Documentatie

Om dit project te realiseren worden de hieronder genoemde documenten opgeleverd. In de loop van het traject zal bepaald worden uit welke deelproducten deze rapporten bestaan.

- Onderzoeksrapporten
- Requirementsrapport
- Designrapport
- Testrapport

4. 3. Ontwikkeling

4.1. 3.1. Ontwikkelmethodiek

De ontwikkelmethode die in dit project gebruikt wordt is Kanban. Kanban is de gebruikte ontwikkelmethodiek bij CityTraffic. Tijdens de ontwikkeling zal om de twee weken een vast contactmoment met de stakeholders zijn, er zijn meer contactmomenten mogelijk. Het bedrijf wil agile werken en andere methodieken als scrum hebben te veel overhead in een project dat door één man wordt uitgevoerd. Er is gekozen om agile te werken, zodat snel feedback gevraagd kan worden van de stakeholders. Nog een reden is dat de stakeholders snel resultaat willen zien, hoewel dit ook mogelijk is met andere methodes, heeft agile hier de beste ondersteuning voor.

De manier van werken lijkt erg op dat van waterval. Dit word mede veroorzaakt doordat de Directeur al een idee in zijn hoofd heeft hoe de applicatie eruit moet zien en wat het uiteindelijk allemaal moet kunnen. Een andere oorzaak is dat de IT afdeling van tevoren al een beeld wil hebben hoe de uiteindelijke applicatie technisch in elkaar zit, zoals welke frameworks er allemaal gebruikt worden. Hierdoor zullen de onderzoeken voor zowel front- als backend eerst uitgevoerd worden.

Het Kanban bord zal bestaan uit vijf rijen:

1. Backlog
   Op de backlog staan alle cards die nog uitgewerkt moeten worden. De exacte wens van de stakeholders is nog niet bekent en moet nog besproken worden.

2. To do
   De to do lijst beschikt over cards waarvan de wens bekend is, maar nog niet in ontwikkeling is.

3. In progress
   De lijst met alle cards die momenteel in ontwikkeling zijn.

4. Acceptance review
   De cards die door de ontwikkelaar uitgewerkt en getest zijn, maar nog niet zijn geaccepteerd door de stakeholders.

5. Done
   Alle cards die klaar zijn om uitgegeven te worden tijdens de volgende release van de software.

4.2. 3.2. Versiebeheer

Git is de gebruikte versiebeheer tool, hierdoor zal er met Git gewerkt worden. De interne Git server van PFM zal gebruikt worden als host.

In Git zal met drie branches gewerkt worden:

1. Master
   De master branch beschikt over de laatst opgeleverde versie van de software.

2. Development
   De development branch wordt gebruikt voor Kanban cards die nog in progress zijn. Zodra deze ontwikkeld is zal het met de acceptatie branch gemerged worden.

3. Acceptatie
   De acceptatie branch wordt gebruikt voor Kanban cards die ontwikkeld zijn, maar nog niet door de opdrachtgever zijn goedgekeurd. Zodra deze zijn goedgekeurd wordt het op de master branch gezet. Als het wordt afgewezen moet het terug naar de development branch.

4.3. 3.3. Ontwikkelomgeving

De ontwikkelomgeving is een virtuele Windows omgeving. Deze omgeving zal door de ontwikkelaar gebruikt worden om de applicatie te testen. Er zal een test database beschikbaar worden gesteld, wat een kopie is van de Advantage II database. De acceptatie testen zullen plaatsvinden op een server, waar Windows Server 2016 als O.S. gebruikt zal worden. Deze omgeving zal ook gebruikt worden voor de performance tests. De server zal gebruik maken van dezelfde test database.

4.4. 3.4. Tools

Tijdens het project zal gebruik worden gemaakt van de volgende tools:

Tool	Beschrijving
Jira	Ticketsysteem dat zal worden gebruikt als Kanban bord
Draw.io	Webtool voor het maken van diagrammen
Microsoft Office	Alle documentatie wordt hiermee opgesteld
Visual Studio 2015	De Integrated development environment

4.5. 3.5. Gebruikte Taal en Libraries

Voor PFM is het van belang dat er gebruik wordt gemaakt van de nieuwste standaarden.

- De webserver zal gemaakt worden in C# met behulp van ASP.NET Core. Om de mogelijkheid te behouden op linux te werken, zal gebruik worden gemaakt van de core library van .NET.

- De backend van de webserver zal mogelijk gebruik maken van Akka.NET.

- De client/UI zal gemaakt worden met een SPA framework. Uit onderzoek zal blijken welke het is. Er is besloten met een spa te werken, omdat op deze manier zo veel mogelijk werk van de webserver weggehaald wordt. Een spa draait volledig in de browser.

- Om de ontwikkeling te stroomlijnen zal er gebruik worden gemaakt van andere plugins:
  - Node.js: Een JavaScript runtime gemaakt op Chrome's V8 JavaScript engine.
  - NPM: Een plugin manager. (Onderdeel van Node.js)
  - Webpack: Een module builder voor javascript applicaties.

  Let op: In de loop van het project kan deze lijst wijzigen.

4.6. 3.6. Definition of Done

Voor elke Kanban card geldt dat deze moet voldoen aan de definition of done voordat deze op klaar gezet kan worden. Hier volgt de algemene definition of done, voor afwijkende items zal de definition of done in Jira bijgehouden moeten worden. Nadat de requirements voor een card bekend zijn kan deze ontwikkeld worden. Voor elke feature moet gekeken worden of deze uitgewerkt moet worden in UML diagrammen. Nadat de feature is ontwikkeld moet deze getest worden. Pas nadat de feature

door de ontwikkelaar is goedgekeurd zal de feature door de stakeholders getest worden. Zodra de stakeholders de feature goedgekeurd heeft zal de bijbehorende card als done gezien worden.

5. 4. Onderzoek

5.1. 4.1. Spa framework

Er zijn vele Javascript frameworks/libraries beschikbaar welke spa kunnen ondersteunen in combinatie met ASP.NET Core. Deze frameworks zijn al meerdere malen met elkaar vergeleken. Om die reden zal de keuze voor een SPA framework gemaakt worden aan de hand van deze onderzoeken.

5.2. 4.2. Actor model framework

Er zijn twee frameworks beschikbaar, met een actieve community, welke het actor model implementeren (in C#) om zo de performance van applicaties te verbeteren, Akka.NET en Microsoft Orleans. Op dit moment heeft Orleans nog geen comptabiliteit met de .NET Core library, deze wordt verwacht in de 2.0 release. Om deze reden kan alleen getest worden met Akka.NET. Uit dit onderzoek moet blijken of het baat heeft het actor model te implementeren in de webserver.

6. 5. Globale Planning

Week	Datum	Beschrijving
1	09-02-2017	Oriëntatie huidige situatie.
2	15-02-2017	Onderzoek SPA framework.
	23-02-2017	Opzet project in Visual Studio & Inrichting verschillende omgevingen (ontwikkel, test en database).
4	27-02-2017	Studeer over Akka.NET
	03-03-2017	Go / No Go gebruik Akka.NET
5	06-03-2017	Eerste gesprek met alle stakeholders. Opstellen en uitwerken van initiële requirements.
	09-03-2017	Ontwikkeling basis layout website.
7	20-03-2017	Acceptatietest layout website & verwerken van feedback.
	23-03-2017	Begin ontwikkeling data-correctie pagina’s.
10	10-04-2017	Acceptatietest data-correctie pagina’s & verwerken van feedback.
11	17-04-2017	Ontwikkeling automatische data-correctie methode.
13	01-05-2017	Performancetest automatische data-correctie methode.
14	08-05-2017	Implementeer Akka.NET in de backend van de website.
15	15-05-2017	Performancetest automatische data-correctie methode. Met Akka.NET implementatie.

Voortgangsverslag

Tim van Egmond
29 Maart 2017
Alphen aan den Rijn

De afstudeerperiode is nu 7 weken onderweg en ik heb het nog steeds erg naar mijn zin. Volgens de globale planning, te vinden in het plan van aanpak, loop ik op dit moment een week achter op de planning. Deze achterstand is veroorzaakt, door een 'uitdaging' een aantal weken geleden, hier later meer over.

Desondanks de achterstand op de planning, is er de afgelopen weken wel veel gedaan. Het Single Page Application (user interface) framework onderzoek is succesvol afgerond, waaruit geconcludeerd is dat Angular 2 gebruikt zal worden. Het Akka.NET onderzoek verliep iets minder soepel. Na onderzoek gedaan te hebben, blijkt dat Akka op meerdere manieren geïntegreerd kan worden in het systeem. Eén van deze manieren is het gebruik van het framework bij functionaliteiten met taken die tegelijktijdig uitgevoerd kunnen worden. Een voorbeeld van zo'n functionaliteit, passend bij de opdracht, is het berekenen van de correcte data van scanners, dit kan voor vele scanners gelijktijdig gedaan worden. Er is ervoor gekozen deze manier, van het gebruik van het framework, toe te passen binnen de afstudeeropdracht.

Nadat het SPA framework onderzoek afgerond was is aan de opzet van het project gewerkt, hier is de vertraging ontstaan. Tijdens het maken van de planning heb ik twee dagen uitgerekend voor de opzet van het project, uiteindelijk zijn het er negen geworden. De webapplicatie maakt gebruik van Webpack voor script bundeling en algemene compilatie workflow voor de front-end. Het was mijn veronderstelling dat het opzetten van Webpack niet meer dan een dag zou kosten, maar dit bleek meer werk te zijn dan verwacht.

Zodra het project opgezet was is een gesprek ingepland met de stakeholders, om de basis layout en design filosofie van de webapplicatie door te nemen. In dit gesprek is afgesproken Angular Material te gebruiken voor alle frontend design. Ook is aan de hand van een wireframe en online voorbeelden besloten hoe de pagina's eruit zien. Nadat er een eerste implementatie was gemaakt van de globale opzet van de website is er een acceptatie test uitgevoerd. De stakeholders waren positief over het eerste resultaat, wel merkte Bart dat de gebruikte kleuren nog niet overeenkomen met de wensen.

De komende periode zal zich allereerst richten op het detecteren van foutieve data. Het is de bedoeling dat de webserver periodiek checkt voor fouten en dat deze fouten opgeslagen worden in de database. Vervolgens zal aan de reparatie gewerkt worden. In overleg met CityTraffic is afgesproken dat de opdracht afgebakend wordt tot alleen camera's, wifiscanners vallen af. Voor wifiscanners is op dit moment nog veel onbekend, ook binnen het bedrijf.

Gerda in t' Veld, begeleidend examinator, is op 10 maart langsgekomen op bedrijfsbezoek. Tijdens dit gesprek is afgesproken dat de manier hoe wordt omgegaan met risico's binnen het project beschreven zou worden. Deze missende informatie is inmiddels verwerkt in desbetreffende documenten.

Naar mijn mening verloopt de afstudeeropdracht goed. Ik weet zeker dat niet de gehele opdracht afgerond kan worden binnen de afstudeerperiode, dus het is van belang dat de wensen goed geprioriteerd worden in overleg met de stakeholders. De beroepstaken en op te leveren producten worden vooralsnog nageleefd en ik verwacht dat ik ze allemaal kan afronden.

Single Page Applications

Onderzoeksrapport

Dit document geeft de lezer inzicht op het onderzoek naar een SPA framework voor het Data Analysis Portal. Hierin wordt naar drie aspecten gekeken: de populariteit, performance en de onderhoudbaarheid.

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Lead	Tim van Egmond
Opdrachtgever	CityTraffic
Plaats	Alphen aan den Rijn
Datum	16 februari 2017
Versie	1.0

Samenvatting

Dit onderzoek is uitgevoerd in opdracht van CityTraffic en moet een geschikt SPA framework selecteren voor het Data Analysis Portal project. In dit onderzoek worden drie aspecten gebruikt om een geschikt framework te selecteren: de populariteit, prestaties en de onderhoudbaarheid. Naast deze aspecten moet het framework voldoen aan een aantal eisen. Omdat er al vele onderzoeken naar SPA frameworks zijn gedaan, zal dit onderzoek kijken naar bestaande onderzoeken.

De geselecteerde SPA frameworks waar dit onderzoek naar kijkt zijn: React, Angular 2, Ember, Vue.js en Aurelia. Van deze frameworks heeft React de meeste ontwikkeling en grootste community. Als rekening wordt gehouden met de leeftijd van het framework blijkt dat Angular 2 het grootst is, zowel in ontwikkeling als community.

De performance van de frameworks ligt dicht bij elkaar, maar er zijn twee uitblinkers. Ember is veruit de langzaamste en heeft het meeste ram gebruik. Vue.js aan de andere kant is het snelst en heeft het minste ram gebruik.

Zowel React als Aurelia voldoen niet aan de eisen. React is meer een library dan een framework, het is gemaakt ter ondersteuning van de view en ondersteunt geen 2-way databinding. Aurelia is nog erg nieuw en heeft gelimiteerde ondersteuning voor third party libraries. Ember is ook een afvaller, de performance is het slechts en de community het kleinst.

Om de onderhoudbaarheid van Angular 2 en Vue.js te bepalen is gekeken naar de structuur, toekomstplannen en de ondersteuning van TypeScript. Angular 2 heeft strengere regels met betrekking tot de structuur dan Vue.js. Naast dat heeft Angular 2 een strak release plan voor toekomstige versies, Vue.js heeft vooral plannen om betere test tooling te ontwikkelen. Angular 2 is gemaakt in TypeScript, terwijl Vue.js nog JavaScript gebruikt. Hoewel het gebruik van JavaScript flexibeler is, is het gewenst TypeScript te gebruiken voor het Data Analysis Portal project. In dit project zal Angular 2 gebruikt worden als SPA framework.

Inhoudsopgaven

1. Inleiding

Dit onderzoek is uitgevoerd in opdracht van CityTraffic. Uit dit onderzoek moet blijken welk SPA framework gebruikt zal worden voor het Data Analysis Portal. Van de lezer wordt verwacht dat hij/zij enige technische kennis heeft.

1.1. Wat is een Single Page Application?

Een single page application is een website dat op de client wordt uitgevoerd en niet op de webserver. Traditionele websites vragen aan een webserver een pagina op en krijgen vervolgens die pagina terug. Een SPA zal in één keer de applicatie (website) ophalen en vervolgens op de client de verschillende pagina's inladen. Dit heeft als gevolg dat er stress van de webserver afgehaald zal worden en dat de gebruiker een betere, maar vooral snellere user experience zal hebben.

"Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app. SPAs use AJAX and HTML5 to create fluid and responsive Web apps, without constant page reloads. However, this means much of the work happens on the client side, in JavaScript." – Mike Wasson; November 2013; Microsoft;

2. Opdracht

2.1. Probleemstelling

Voor het Data Analysis Portal project is een single page applicatie framework vereist. Op dit moment zijn er vele van dit soort frameworks in ontwikkeling zoals: Anglular2, React en Ember, om er een aantal te noemen. Het is niet bekent welk van deze frameworks het meest geschikt is voor dit project.

2.2. Doelstelling

Het doel van dit onderzoek is een geschikt single page application framework te vinden, welke gebruikt kan worden in het Data Analysis Portal project.

2.3. Eisen

Voor dit project zijn een aantal eisen opgesteld waar het framework aan moet voldoen. Mocht deze functionaliteit niet native beschikbaar zijn, maar wel via een plugin is dit voldoende.

1. Er is gedegen documentatie beschikbaar.
2. Er moet ondersteuning zijn voor lazy loading.
3. Er is ondersteuning voor routing met variabelen.
4. Er is minimaal ondersteuning voor 2-way data binding.
5. Er is compatibiliteit met andere JS frameworks, zoals datatables.
6. Er is ondersteuning voor TypeScript.
7. IFrames moeten geopend kunnen worden.

2.4. Vragen

Om de doelstelling van dit onderzoek te bereiken en de beste keus te maken, zijn er een aantal vragen opgesteld. Aan de hand van de deelvragen zal de hoofdvraag beantwoord kunnen worden.

Hoofdvraag:

> *"Welk SPA framework kan het best gebruikt worden voor het data-analyse systeem project?"*

Deelvragen:

1. Welke SPA frameworks voldoen aan de eisen?
2. Welk SPA framework heeft de beste ondersteuning qua development en community?
3. Welk SPA framework heeft de beste performance?
4. Welk SPA framework is het best te onderhouden?

3. Werkwijze

Er zijn al vele onderzoeken en artikelen geschreven door mensen met meer ervaring van SPA frameworks. Om deze reden zal dit onderzoek kijken naar de bevindingen van andere mensen en deze bevindingen op een overzichtelijke manier weergeven. Vervolgens zal per framework gekeken worden of deze aan de eisen kan voldoen.

4. Overzicht frameworks

Voor dit onderzoek zijn vijf SPA frameworks geselecteerd. Deze frameworks behoren tot de meest bekende en gebruikte SPA frameworks van dit moment. Hieronder een tabel met algemene gegevens, deze data is verzameld op 13 februari 2017. Bij het opstellen van deze gegevens zijn de volgende bronnen gebruikt: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20.

	React	Angular 2	Ember	Vue.js	Aurelia
Bedrijf / sponsors	Facebook	Google	(meerdere)	(meerdere)	Blue Spire
Source	Open	Open	Open	Open	Open
Versie	15.4.2	2.4.7	2.11.0	2.1.10	1.0.8
Eerste uitgaven (volledige 1.0)	29 mei 2013	15 sep 2016	1 sep 2013	27 okt 2015	27 jul 2016
Taal	JSX	TypeScript	JavaScript	JavaScript	JavaScript
Data binding	1-way	2-way	2-way	2-way	2-way
Architectuur	Geen (alleen V)	MVC	MVC	MVC	MVC
Lazy loading	Wel	Wel	Wel	Wel	Wel
Server side rendering	Wel	Wel	Wel	Wel	Wel
Git Watchers	4.152	2.274	1.086	2.484	525
Git Stars	59.550	20.564	17.512	42.815	9.012
Git Fork	10.873	5.307	3.643	5.342	516

4.1. Community & Populariteit

Al de geselecteerde frameworks zij open source en worden voornamelijk door de community onderhouden. Om er zeker te zijn van het langdurige onderhoud van een framework is het van belang dat deze populair is. Aan de hand van de hoeveelheid git gebruikers, te zien in bovenstaande tabel, is af te leiden hoe groot de community van een framework is. Het is wel belangrijk om te weten dat deze gebruikers niet actief hoeven te zijn. Omdat de repositories verschillende eerste uitgaven datums hebben is de formule als volgt:

Totale hoeveelheid **/** *Aantal dagen sinds eerste uitgave*

Angular 2 en AngularJS (de originele) moet en als twee aparte frameworks gezien worden, er zijn ook aparte repositories. Toch zullen veel mensen Angular 2 gevolgd hebben door het feit dat het de naam deelt. Om deze reden is er een extra kolom welke deze twee frameworks combineert. Deze kolom is berekend op de volgende manier:

(Totale hoeveelheid Angular 2 + Totale hoeveelheid AngularJS)
/
(Aantal dagen sinds eerste uitgaven Angular 2 + Aantal dagen sinds eerste uitgave AngularJS)

	React	Angular 2	AngularJS + Angular 2	Ember	Vue.js	Aurelia
Git Watchers	3.06	15,06	3.63	0,86	5,23	2,61
Git Stars	43,92	136,19	40,53	13,89	90,14	44,83
Git Fork	8,02	25,15	17.53	2,89	11,25	2,57

Aan de hand van deze gegevens kunnen de frameworks gerangschikt worden op drie aspecten.

	Git Watchers	Git Stars	Git Fork
1	Angular 2	Angular 2	Angular 2
2	Vue.js	Vue.js	AngularJS + Angular 2
3	AngularJS + Angular 2	Aurelia	Vue.js
4	React	React	React
5	Aurelia	AngularJS + Angular 2	Ember
6	Ember	Ember	Aurelia

Uit de rangschikking hierboven is gelijk te zien dat Angular 2 op dit moment het populairst is, gevolgd door Vue.js. De populariteit van Vue.js is te verklaren door het feit dat veel mensen van AngularJs zijn overgestapt naar Vue.js. Angular 2 is niet populair onder de JavaScript community en richt zich volgens hun teveel op de Java en C# OO programmeur. [2]

5. Performance

In dit hoofdstuk is te lezen welk van deze frameworks het snelst is en het minste geheugen gebruikt. Deze performance testen zijn uitgevoerd door Stefan Krause [12], zijn resultaten staan in [14].

5.1. Keyed & non-keyed

Niet elk framework werkt op dezelfde manier. Er is een splitsing te maken tussen keyed en non-keyed frameworks. Het gebruik van keyed of non-keyed heeft impact op de performance en usability van het framework. Zodra non-keyed gebruikt wordt zal de ondersteuning van 3rd-party javascript beperkt worden. Deze methodes zijn vooral interessant bij het gebruik van lijsten van data. [15]

Het verschil tussen keyed en non-keyed is het beheren van de DOM nodes op een pagina. Zodra iets inde controller veranderd, zal de keyed methode alle DOM nodes opnieuw creëren. De non-keyed methode zal de bestaande Dom nodes updaten. [15]

De keyed methode heeft een één op één relatie tussen de data in de controller en de DOM node. Dit zorgt ervoor dat wanneer de DOM node buiten het framework veranderd, dit geen impact heeft op de controller. Een voordeel is dat zodra de data in de controller van volgorde veranderd, de DOM nodes mee veranderen. Een ander voordeel is dat wanneer de styling buiten het framework veranderd dit wordt overschreven zodra het framework de DOM node update. Het gebruik van de non-keyed methode heeft als voordeel dat het vaak sneller is en minder geheugen gebruikt. [14]

Zowel Angular 2, React en Vue ondersteunen op dit moment beide manieren. Ember ondersteund alleen keyed en Aurelia ondersteund alleen non-keyed. [14]

5.2. Scenario's

De performance testen zijn gebaseerd op de volgende scenario's.

S1	**Create rows** Duration for creating 1000 rows after the page loaded.
S2	**Replace all rows** Duration for updating all 1000 rows of the table (with 5 warmup iterations).
S3	**Partial update** Time to update the text of every 10th row (with 5 warmup iterations).
S4	**Select row:** Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).
S5	**Swap rows** Time to swap 2 rows on a 1K table (with 5 warmup iterations).
S6	**Remove row** Duration to remove a row (with 5 warmup iterations).
S7	**Create many rows** Duration to create 10,000 rows
S8	**Append rows to large table** Duration for adding 1000 rows on a table of 10,000 rows.
S9	**Clear rows** Duration to clear the table filled with 10.000 rows.
S10	**Slowdown geometric mean**

5.3. Resultaten

Al de onderstaande resultaten zijn in milliseconden.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
React v15.4.2-keyed	184.29	198.68	15.44	6.86	13.42	50.04	1824.01	327.64	410.94	**1.26**
React v15.4.2-non-keyed	181.98	77.90	15.61	4.97	11.31	66.01	1805.81	326.55	423.06	**1.35**
Angular v2.4.3-keyed	178.12	188.54	9.86	3.89	11.34	48.68	1832.15	288.12	342.39	**1.21**
Angular v2.4.3-non-keyed	179.20	54.62	10.40	8.65	7.85	37.36	1857.48	292.79	336.63	**1.18**
Ember v2.10.0-beta.3-keyed	309.09	283.64	14.91	5.94	15.26	53.36	2496.88	457.75	251.29	**1.43**
Vue v2.1.10-keyed	151.97	158.72	15.62	8.37	16.81	55.36	1551.08	369.41	248.823.33	**1.15**
Vue v2.1.10-non-keyed	152.92	62.61	15.51	7.66	13.60	46.09	1514.01	361.10	246.073.18	**1.16**
Aurelia v1.0.7-non-keyed	178.81	83.52	9.97	11.87	10.94	61.72	1880.92	299.99	291.635.14	**1.29**

De volgende resultaten zijn in megabytes.

	Ready memory Memory usage after page load.	**Run memory** Memory usage after adding 1000 rows.
React v15.4.2-keyed	4.73	11.66
React v15.4.2-non-keyed	4.97	11.64
Angular v2.4.3-keyed	5.86	12.46
Angular v2.4.3-non-keyed	5.89	12.46
Ember v2.10.0-beta.3-keyed	10.76	21.20
Vue v2.1.10-keyed	3.65	8.89
Vue v2.1.10-non-keyed	3.65	9.25
Aurelia v1.0.7-non-keyed	7.42	14.15

De resultaten van de performance testen liggen dicht op elkaar, maar er zijn twee duidelijke uitschieters. Ember is in de meeste scenario's veruit het langzaamst. Aan de andere kant is Vue.js juist weer een stuk sneller. De resultaten van de performance test zijn terug te zien in het geheugen gebruik. Ember gebruikt het meest en Vue gebruikt het minst.

6. Voldoening eisen

In het volgende hoofdstuk is te lezen welk van de frameworks aan al onze eisen voldoen. Voor de frameworks die niet aan de eisen voldoen zal beschreven worden waarom dit is. Vervolgens zal beschreven worden met welk framework het onderzoek zal worden voortgezet.

Een volledige beschrijving van de eis kan gevonden worden in 2.3.

	React	**Angular 2**	**Ember**	**Vue.js**	**Aurelia**
1. Gedegen documentatie	✓	✓	✓	✓	✓
2. Lazy loading	✓	✓	✓	✓	✓
3. Routing	✓	✓	✓	✓	✓
4. 2-Way databinding	✘	✓	✓	✓	✓
5. Compatibiliteit JS frameworks	✓	✓	✓	✓	✘
6. Typescript	✓	✓	✓	✓	✓
7. Iframes	✓	✓	✓	✓	✓

- **React**
  Zoals in bovenstaande diagram is te zien heeft React geen ondersteuning voor 2-way databinding. Dit wordt veroorzaakt doordat React alleen de user interface verzorgt. Deze functionaliteit kan wel worden ingebouwd door voor elke binding een onChance handler te maken, maar dit is voor ons niet voldoende, te veel onnodig werk.

- **Aurelia**
  Aurelia heeft, naar onze mening, niet voldoende comptabiliteit met andere JS frameworks. Zoals in hoofdstuk 5.1. is te lezen heeft Aurelia op dit moment geen ondersteuning voor de keyed methode. Na te weten te zijn gekomen wat dit inhoud, is besloten dat voor dit project een framework in gebruik wordt genomen welke deze functionaliteit wel beschikt.

Op dit moment van het onderzoek is het zeker dat er meerdere frameworks zijn, die niet gebruikt zullen worden genomen. Twee daarvan zijn simpel, React en Aurelia voldoen niet aan de eisen. Ember voldoet wel een de eisen, maar is ook een afvaller. Naar onze mening is de community en populariteit te klein in vergelijking met Angular 2 en Vue.js. Ook is de performance en geheugen gebruik in de meeste scenario's slechter in vergelijking met de andere frameworks. Tot slot heeft Ember geen ondersteuning voor de non-keyed methode Angular 2 en Vue.js hebben dit wel. Om deze redenen zal dit onderzoek worden voortgezet met Angular 2 en Vue.js.

7. Onderhoudbaarheid

Zoals in hoofdstuk 6. Is uitgelegd zal dit hoofdstuk alleen kijken naar Angular 2 en Vue.js. Uit dit hoofdstuk moet blijken welk van de frameworks de langste levensduur en ondersteuning zal hebben.

7.1. Filosofie & Structuur

Angular 2 lijkt zich te richten op de professionele markt. Dit wordt voornamelijk veroorzaakt door de harde eis voor het gebruik van TypeScript. Daarnaast raat het Angular team sterk aan met een bepaalde design filosofie te werken. Elk aparte module staat in een eigen folder en de html template, typescript code en css styling hoort in aparte bestanden. [16] [17]

Vue.js richt zich op een breder publiek. Typescript wordt ondersteund, maar is niet verplicht en ondersteund niet de complete API [22]. De structuur van de applicatie staat ook minder vast, het Vue.js team wil hier ook geen vaste regels voor maken. Het team wil de ontwikkelaar juist zo veel mogelijk flexibiliteit geven en in staat stellen om een applicatie te maken in zo min mogelijk regels code. [16]

7.2. Roadmap

Vue.js is wil in 2017 richten op de ontwikkeling van test tools. Naast deze ontwikkeling willen ze de documentatie een fixe opknapbeurt geven. Angular 2 aan de andere kant heeft Angular 7 al op de planning staan. Angular3 wordt overgeslagen en in maart 2017 staat de release van Angular 4 op de planning. Het updaten vanaf Angular 2 zou een stuk [18] [19]

7.3. Uitbreidbaarheid

Zowel Angular 2 als Vue.js hebben een breed scala van third-party plugins, waardoor alle basis functies ondersteund worden. Wel lijkt het erop dat er op dit moment meer plugins voor Agnular 2 zijn. Daarnaast vereist Vue.js geen TypeScript en is het maar de vraag of de plugins ook ondersteuning voor TypeScript bieden.

7.4. TypeScript

Het ontwikkelen van een Angular 2 applicatie vergt meer kennis dan een Vue.js applicatie. Wel zorgt de vaste Angular structuur ervoor dat een nieuwe ontwikkelaar, met kennis van Angular 2, zich gelijk thuis voelt in een bestaand project. Het gebruik van TypeScript vereist ook nieuwe kennis, ook voor een JavaScript programmeur. Voor het Data Analysis Portal is besloten van TypeScript gebruik te maken, voornamelijk omdat op deze manier fouten eerder gevonden kunnen worden tijdens compilatie. Daarnaast is JavaScript niet een gewenste taal om in te werken bij PFM, dit is C#. De overstap naar TypeScript is gemakkelijker dan de overstap naar JavaScript.

8. Conclusie

In dit onderzoek is naar vijf frameworks gekeken: React, Angular 2, Ember, Vue.js en Aurelia. Aan de hand van de totale hoeveelheid Git watchers kan afgeleid worden dat React de grootste community heeft. Ook heeft React de meeste Git forks, wat inhoud dat er vele mensen ontwikkelen of ontwikkeld hebben voor React. React is ook het oudste framework en zodra daar rekening mee wordt gehouden blijkt dat Angular2 gevolgd door Vue.js de populairste zijn.

De performance van de frameworks ligt dicht op elkaar, maar er zijn twee uitblinkers. Ember is veruit de langzaamste en heeft het meeste ram gebruik. Vue.js aan de andere kant is het snelst en heeft het minste ram gebruik.

Voor het Data Analysis Portal zijn een aantal eisen opgesteld. Drie van de framewoks voldoen aan de eisen, dit zijn: Angular 2, Vue.js en Ember. React is geen compleet framework, het kan beter als een library gezien worden. React zorgt puur voor de view van een webapplicatie, om deze reden ondersteund het geen 2-way databinding. Aurelia vaalt op het gebied van compatibiliteit. Dit framework ondersteund geen keyed methode, wat betreft DOM-beheer en dit is wel wenselijk.

Ember scoort niet goed in de meeste categorieën. De performance is het minst en het heeft de minst grootte community. Om deze twee reden is besloten het onderzoek niet voort te zetten voor dit framework.

Welk framework het best te onderhouden is, is alleen onderzocht voor Angular 2 en Vue.js. Kijkende naar de roadmap lijkt het erop dat Angular 2 een actievere ontwikkeling heeft dan Vue.js. Dit kan betekenen dat Vue.js nu wel de beste performance heeft, maar dit volgend jaar heel anders kan zijn. Daarnaast is het gebruik van TypeScript voor ons erg belangrijk. Angular 2 is in TypeScript geschreven en heeft dus goede ondersteuning, ook de plugins. Vue.js is dit niet, maar heeft wel ondersteuning voor TypeScript. Deze TypeScript ondersteuning is alleen niet dekkend voor de gehele API. Ook is het niet zeker dat de plugins van Vue.js deze ondersteuning ook bieden.

Voor het Data Analysis Portal zal Angular 2 gebruikt worden. Dit is het enigste framework waarvan wij zeker zijn dat het aan al onze requirements voldoet en het een lang leven voor zich heeft.

Verwijzingen

[1] E. Korotya, „5 Best JavaScript Frameworks in 2017,” 19 Januari 2017. [Online]. Available: https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282. [Geopend 13 Februari 2017].

[2] J. P. Miles, „Why Vue 2 beats Angular 2 and React,” 5 November 2016. [Online]. Available: https://medium.com/@codingfriend/why-vue-2-beats-angular-2-and-react-cfb709b92c59. [Geopend 13 Februari 2017].

[3] „React,” [Online]. Available: https://github.com/facebook/react. [Geopend 2017 Februari 13].

[4] „Angular,” [Online]. Available: https://github.com/angular/angular. [Geopend 13 Februari 2017].

[5] „Ember,” [Online]. Available: https://github.com/emberjs/ember.js. [Geopend 13 Februari 2017].

[6] „Vue,” [Online]. Available: https://github.com/vuejs/vue. [Geopend 13 Februari 2017].

[7] „Aurelia,” [Online]. Available: https://github.com/aurelia/framework. [Geopend 13 Februari 2017].

[8] i. hrisVillanueva, „Lazy Loading - React,” [Online]. Available: https://webpack.js.org/guides/lazy-load-react/. [Geopend 13 Februari 2017].

[9] duizendnegen, „ember-cli-lazy-load,” Juni 2016. [Online]. Available: https://www.npmjs.com/package/ember-cli-lazy-load. [Geopend 13 Februari 2017].

[10] „Lazy Loading Routes,” [Online]. Available: https://router.vuejs.org/en/advanced/lazy-loading.html. [Geopend 13 Februarie 2017].

[11] „Declaring Build Resources,” [Online]. Available: http://aurelia.io/hub.html#/doc/article/aurelia/framework/latest/bundling-webpack/6. [Geopend 13 Februari 2017].

[12] D. D. Toom, „Angular 2 lazy loading with Webpack,” 1 October 2016. [Online]. Available: https://medium.com/@daviddentoom/angular-2-lazy-loading-with-webpack-d25fe71c29c1. [Geopend 13 Februari 2017].

[13] S. Krause, „JS web frameworks benchmark – Round 5,” 25 Januari 2017. [Online]. Available: http://www.stefankrause.net/wp/?m=201701. [Geopend 14 Februari 2017].

[14] S. Krause, „Results for js web frameworks benchmark – round 5,” 25 Januari 2017. [Online]. Available: http://stefankrause.net/js-frameworks-benchmark5/webdriver-ts/table.html. [Geopend 14 Februari 2017].

[15] S. Krause, „JS web frameworks benchmark – keyed vs. non-keyed,” 24 Januari 2017. [Online]. Available: http://www.stefankrause.net/wp/?p=342. [Geopend 14 Februari 2017].

[16] „Comparison with Other Frameworks," [Online]. Available: https://vuejs.org/v2/guide/comparison.html#Angular-2. [Geopend 15 Februari 2017].

[17] „What Is AngularJS?," [Online]. Available: https://docs.angularjs.org/guide/introduction. [Geopend 15 Februari 2017].

[18] Brian, „Angular, React, and Vue: What's Coming in 2017?," 20 Januari 2017. [Online]. Available: https://angular.jsnews.io/angular-react-and-vue-whats-coming-in-2017-js-javascript-angular2-reactjs-vuejs/. [Geopend 15 Februari 2017].

[19] I. Minar, „Versioning and Releasing Angular," 7 October 2016. [Online]. Available: https://angularjs.blogspot.nl/2016/10/versioning-and-releasing-angular.html. [Geopend 15 Februari 2017].

[20] „TypeScript Support," [Online]. Available: https://vuejs.org/v2/guide/typescript.html. [Geopend 13 Februari 2017].

Akk.NET Backend Server

Onderzoeksrapport

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Lead	Tim van Egmond
Opdrachtgever	CityTraffic
Plaats	Alphen aan den Rijn
Datum	10 maart 2017
Versie	1.1

Inhoudsopgaven

1. Inleiding

Dit onderzoek is uitgevoerd om een beeld te geven hoe Akka.NET gebruikt kan worden in het Data Analysis Portal project. Van de lezer wordt verwacht dat hij/zij enige technische kennis heeft.

1.1. Wat is Akka.NET

Akka.NET is een framework dat zich richt op schaalbare, fouttolerante real-time data verwerking. Dit wordt berijkt door gebruik te maken van het Actor framework. Akk.NET is een .NET port van de Java / scala variant Akka. Akka is al vele malen succesvol toegepast in zware productieomgevingen, zoals de webservers van Zalando en Twitter.

“We believe that writing correct, concurrent, fault-tolerant and scalable applications is too hard. Most of the time, that's because we are using the wrong tools and the wrong level of abstraction. Akka is here to change that.” – Akka.NET

2. Opdracht

2.1. Probleemstelling

De performance van het data-analysesysteem is van groot belang. De applicatie zal een grote hoeveelheid data te verwerken krijgen, wat zo snel mogelijk verwerkt moet worden. Door deze reden moet er een oplossing gezocht worden welke dit probleem op kan lossen.

2.2. Doelstelling

Het doel van dit onderzoek is uitzoeken of het actor framework en specifieker Akka.NET gebruikt kan worden om bovenstaand probleem op te lossen / te verminderen.

2.3. Eisen

Voor de backend gelden een aantal eisen, waar Akka.NET aan moet voldoen.

1. Akka.NET moet sneller zijn dan de traditionele manier.
2. Akka.NET moet goed schaalbaar zijn.
3. Akka.NET moet excepties op kunnen lossen.

2.4. Vragen

Om de doelstelling van dit onderzoek te bereiken en de beste keus te maken, zijn er een aantal vragen opgesteld. Aan de hand van de deelvragen zal de hoofdvraag beantwoord kunnen worden.

Hoofdvraag:

“Hoe zal Akka.NET geïntegreerd worden binnen dit project?”

Deelvragen:

5. Kan Akka.NET gebruikt worden om de performance van het systeem te waarborgen?
6. Hoe kan Akka.NET geïntegreerd worden in de webapplicatie?
7. Hoe behaalt Akka.NET schaalbaarheid en fouttolerantie?

3. Werkwijze

De voorkennis betreft Akka.NET en het actor model is minimaal. Om deze reden zal allereerst een bootcamp gevolgd worden, welke verschillende principes behandeld en demonstreert. Vervolgens zal gekeken worden hoe Akka.NET geïntegreerd kan worden in een webapplicatie. Aan de hand daarvan zal een schatting gemaakt kunnen worden of het de performance van de webapplicatie kan ondersteunen.

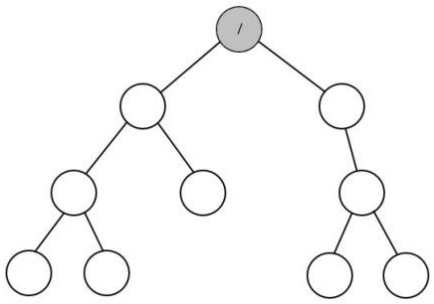
De bootcamp is te vinden in bron: [1]

4. Algemene informatie

Dit hoofdstuk is bedoeld voor mensen die nog niet bekent zijn met Akka.NET en het Actor model. Vanaf dit punt zal naar Akka.NET verwezen worden met gewoon Akka.

4.1. Actor systeem

Een actor is een object welke zowel status als gedrag bezitten. Actoren kunnen door middel van berichten met elkaar communiceren, deze berichten kunnen zowel synchroon als asynchroon zijn. Het actor systeem kan het best gezien worden als een groep mensen. Een persoon is de leider, die afzonderlijke taken verdeeld onder zijn team. Hierdoor ontstaat een hiërarchische structuur, deze structuur is gevisualiseerd in Figuur 1. Het voordeel dat het actor systeem biet over het traditionele multithreading is dat de ontwikkelaar niet na hoeft te denken over thread beheer, het actor systeem neemt deze taak op zich. Hierdoor kunnen actoren in één thread draaien of in honderd verschillende. [2]



Figuur 1: Het actor model gevisualiseerd

4.2. Schaalbaarheid

Schaalbaarheid is een belangrijk aspect van Akka, dit aspect wordt bereikt op twee manieren. In 4.1 is te lezen dat het actor model gezien kan worden als een groep mensen, met elk afzonderlijke taken. Als uitbreiding hierop kan ook ingebeeld worden dat er meerdere mensen zijn met dezelfde taak. Zodra de leider meerdere keren dezelfde taak krijgt kan deze taak verdeeld worden over de groep mensen die deze taak kan uitvoeren. In Akka kan dit ook, door middel van verschillende strategieën kunnen meerdere actoren aangemaakt worden met dezelfde taak. Dit principe wordt routing genoemd. [3]

Een andere manier waarop Akka schaalbaarheid bereikt, is het gebruik van clusters en remoting. Remoting is het gebruik van meerdere servers waarop het actor systeem actoren kan plaatsen en gebruiken. Het actor systeem is niet afhankelijk van één computer, het kunnen er gerust vijf of meer zijn. [4]

4.3. Fouttolerantie

Fouttolerantie is het tweede belangrijke aspect van Akka, dit kan bereikt worden op meerdere manieren. Zoals in 4.1 te lezen is, is er een hiërarchische structuur van actoren. De relatie tussen actoren kan gezien worden als een child parent relatie. Elke parent is verantwoordelijke voor de child actoren, dit wordt gedaan door middel van de supervisor strategy. Zodra een child actor stopt met werken zal de parent beslissen wat er moet gebeuren en kan de child actor opnieuw opgestart worden. Alle berichten die de gefaalde actor gekregen heeft kunnen alsnog verwerkt worden door een andere actor, hierdoor zal er geen werk verloren gaan. [5]

Een ander onderdeel wat een actor systeem zelf herstellend maakt, is het gebruik van clusters. Clusters is een abstractie laag bovenop remoting, behandeld in 4.2. Een cluster is een peer-to-peer netwerk van Akka applicaties, dit kunnen verschillende applicaties zijn, maar hoeft niet. Door het gebruik van clusters kan het systeem zichzelf herstellen, zodra een actor systeem uitvalt. Wanneer een actor systeem uitvalt kan het werk van dit systeem overgenomen worden door een ander systeem. [6, 7]

5. Mogelijke scenario's gebruik Akka

Nadat de bootcamp [1] uitgevoerd is nagedacht over de verschillende manieren hoe Akka geïntegreerd kan worden in de webserver. Tijdens dit proces is ook gekken naar het artikel in bron [8]. Dit artikel geeft een mogelijke integratie met behulp van een zogenaamde applicatieserver.

5.1. Toelichting app server

De applicatie (app) server is de server, of cluster van servers, waar alle logica van de Akka applicatie op staat. Deze server functioneert als een soort cache dat leidend is voor de database. De webserver en alle andere servers die data van de database nodig hebben, kunnen deze data via de applicatie server ophalen. Het voordeel dat hiermee bereikt wordt, is dat data vele male sneller aangeleverd kan worden. Databases schalen heel slecht mee met een toenemende hoeveelheid gebruikers, vooral bij het toevoegen van data. Een ander voordeel dat bereikt wordt is dat er een duidelijke afsplitsing komt qua verantwoordelijkheden van servers. [8]

Figuur 2: Mogelijke server setup

5.2. Oplossing 1: Een nieuw systeem

De beste lange termijn oplossing is het maken van een nieuw systeem. Dit systeem zal van de grond af opgebouwd worden met Akka, waarmee een betrouwbare App Server gemaakt kan worden. Het voornaamste voordeel dat deze oplossing met zich meeneemt is dat de alle losstaande systemen samengevoegd kunnen worden in één. Deze oplossing zal wel erg veel tijd kosten om te realiseren. Het realiseren van een systeem van deze omvang zal te veel tijd kosten binnen dit project.

Voordeel	Nadeel
Één systeem	Groote investering
Meer future proof	Vereist veel kennis

5.3. Oplossing 2: Integreren in de data analyse portal (zonder app server)

De snelste oplossing is om Akka te integreren in het huidige systeem zonder app server. Akka zou hierin alleen gebruikt kunnen worden voor de taken die ook met multithreading opgelost kunnen worden. Dit komt voornamelijk doordat Akka alsnog afhankelijk zal zijn van de tijd dat het kost om queries en http requests, naar een api of database uit te voeren. In het geval dat een gebruiker enkel data uit de database op wil halen is het niet verstandig Akka te gebruiken. Akka kan de snelheid van de database niet verbeteren. Wel kan je met akka ervoor zorgen dat er meerdere database vragen tegelijk uitgevoerd worden. In het geval dat er maar één database vraag is, is het beter gebruik te maken van de cache van het entity framework.

Voordeel	Nadeel
Makkelijk te integreren	
Mogelijke performance winst bij bepaalde workloads	

5.4. Oplossing 3: Integreren in de data analyse portal (met app server)

De data analyse portal zou ook gemaakt kunnen worden in het huidige systeem, met een app server. Dit is een onhandige oplossing, omdat alle huidige systemen omgebouwd moeten worden, doordat de datastroom door de app server moet gaan. Daarnaast zal dit niet de voornaamste pluspunt verhelpen, alle systemen samenvoegen in één.

Voordeel	Nadeel
Een leidende app server	Veel werk
	Mogelijk weinig performance winst

6. Advies

Mijn advies is om voor oplossing 2 te kiezen. Aan deze oplossing zitten de minste risico's gebonden, er is geen aanpassing aan de huidige systemen voor nodig. Wel kan op deze manier het algoritme geoptimaliseerd worden, door op een gemakkelijke manier meerdere taken gelijktijdig te laten verlopen. Er kan op een veilige manier met multithreading gewerkt worden.

Op lange termijn zal ik wel kijken naar vervangingen voor het huidige systeem. Het lijkt mij niet wenselijk te werken met veel verschillende servers die afhankelijk zijn van elkaar. Zeker met de toenemende vraag naar realtime data en de slechte schaalbaarheid van databases, lijkt oplossing 1 een goede toekomstige investering.

Oplossing 3 lijkt mij niet verstandig het geeft veel werk voor mogelijk weinig winst. Ook los je het probleem niet op dat er nu veel servers in een keten van elkaar afhankelijk zijn.

Het is niet bekend of één van deze oplossingen sneller zal zijn dan reguliere multithreading. Dit is ook onmogelijk voor mij van tevoren betrouwbaar te voorspellen en ik kan dus ook niet garanderen dat Akka kan voldoen aan de eis dat het sneller is. Mijn verwachting dat Akka wel sneller is in het geval dat er veel gebruikers zijn, omdat Akka bepaald hoeveel threads er zijn. Stel dat je voor elke gebruiker regulier 5 threads aanmaakt, dan wordt het al snel teveel voor een operating system.

7. Conclusie

Akka kan gebruikt worden om de performance en fouttolerantie van het systeem te waarborgen. Het framework kan dit bereiken door runtime, op een veilige manier, meer of minder threads aan te maken, waar de verschillende actoren in leven. Als er iets fout gaat zal het framework deze fout op proberen te lossen en het proces opnieuw op starten. Er zijn meerdere manieren om Akka te integreren in de webapplicatie, dit kan zowel met als zonder applicatieserver. De beste lange termijn oplossing is om een nieuw systeem te ontwerpen rondom Akka. Dit is niet reëel en ver buiten de scope van de originele opdracht. Wat wel kan is het gebruik van Akka, voor onderdelen waar werk tegelijktijdig uitgevoerd kan worden.

Voor dit project is ervoor gekozen Akka te gebruiken als multithreading framework. Hoewel er niet van tevoren met zekerheid gezegd kan worden dat het de performance van het systeem verbeterd, zal dit project ook als test voor Akka functioneren. Voor het project is het dus van belang dat er een goede performance test gemaakt zal worden, wat aantoont of Akka de performance van het systeem verbeterd.

Verwijzingen

[1] A. Stannard, „Akka.NET Bootcamp," 18 Januari 2017. [Online]. Available: https://github.com/petabridge/akka-bootcamp. [Geopend 2 Maart 2017].

[2] „Actor Systems," [Online]. Available: http://getakka.net/docs/concepts/actorsystem. [Geopend 8 Maart 2017].

[3] „Routers," [Online]. Available: http://getakka.net/docs/working-with-actors/Routers. [Geopend 8 Maart 2017].

[4] „Akka.Remote Overview," [Online]. Available: http://getakka.net/docs/remoting/. [Geopend 8 Maart 2017].

[5] „Fault Tolerance," [Online]. Available: http://getakka.net/docs/Fault%20tolerance. [Geopend 8 Maart 2017].

[6] „Akka.Cluster Overview," [Online]. Available: http://getakka.net/docs/clustering/cluster-overview. [Geopend 8 Maart 2017].

[7] R. Roestenburg, R. Bakker en R. Williams, „Akka in Action: Why use clustering?," September 2016. [Online]. Available: http://freecontent.manning.com/akka-in-action-why-use-clustering/. [Geopend 8 Maart 2017].

[8] A. Stannard, „The Inevitable Rise of the Stateful Web Application," 17 Augustus 2015. [Online]. Available: https://petabridge.com/blog/stateful-web-applications/. [Geopend 8 Maart 2017].

Requirements Discipline

Data Analysis Portal

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Lead	Tim van Egmond
Opdrachtgever	Citytraffic
Plaats	Alphen aan den Rijn
Datum	19 Mei 2017
Versie	0.9

Table of Contents

1. Requirements

1.1. Processes

ID	Process
p.1	Repairing of footfall and scanner data

1.2. Business Rules

ID	Rule	Priority	Origin	Notes
Rule.1	Only by user approval can the system save corrected data to the database	Must	Bart	
Rule.2	Footfall and wifi data is aggregated per 30 minutes	Must	Gerrit	This is subject to change
Rule.3	New data gets delivered to Advantage II once every day	Must	Gerrit	This is subject to change
Rule.4	Data is deviated when it is more than 20 % of the average.	Must	Chris, Gerrit	
Rule.5	The average is calculated from the last 175 days	Must	Chris, Bart	
Rule.6	The average can only use data from the same days of the week (ex. Only mondays)	Must	Chris, Bart	
Rule.7	The footfall error types are no data, partly zero and deviation	Must	Chris, Bart	

1.3. Business Requirements

ID	Requirement	Priority	Derived from	Origin	Notes
BR.1	The automatic repair algorithm is faster than the repair algorithm in Advantage II	Must	p.1	Bart	No exact numbers of the old situation are known
BR.2	The error detection algorithm is faster than the algorithm in Advantage II	Must	p.1	Bart	No exact numbers of the old situation are known
BR.3	The backend of the application uses the existing Advantage II API	Should	p.1	Bart	The API currently does not deliver all data. The entity framework will be used.
BR.4	The application uses a single page application as user interface	Must	p.1	Bart, Chris	
BR.5	The user must be logged in at all time to use the application	Must	p.1	-	
BR.6	The single page application uses lazy loading	Should	p.1	Chris	
BR.7	The single page application uses server side rendering	Should	p.1	Chris	

BR.8	The user interface can support multiple languages	Could	p.1	Bart	
BR.9	The error detection algorithm runs periodically on the webserver	Should	p.1	Chris	
BR.10	The error detection algorithm is the same as the detection algorithm used in advantage II	Should	p.1	Chris	The algorithm uses the same errors. It is not possible to reuse the code.
BR.11	The error detection algorithm checks if the data exists (no data error)	Must	p.1	Chris	
BR.12	The error detection algorithm checks if the data is 0 on timestamps (partly 0 error)	Must	p.1	Chris	
BR.13	The error detection algorithm checks if the data is deviated on timestamps (deviation error)	Must	p.1	Chris	

1.4. User Requirements

ID	Use case	Requirement	Priority	Derived from	Origin
UR.1		The user can log in and out	Should	BR.5	-
UR.2		The user can switch between languages	Could	BR.8	-
UR.3		The user can toggle the navigation menu	Must	BR.4	Bart
UR.4	UC.1	The user can see an overview of all errors	Must	BR.4	Koen, Adrie
UR.5	UC.1	The user can see an overview of all today errors	Must	BR.4	Koen, Adrie
UR.6	UC.2	The user can repair count data of footfall devices	Must	BR.4	Koen, Adrie
UR.7	UC.2	The user can repair count data of wifi devices	Should	BR.4	Koen, Adrie
UR.8	UC.2	The user can repair KPI data	Could	BR.4	Koen, Adrie
UR.9	UC.3	The user can filter errors by error type	Should	BR.4	Koen, Adrie
UR.10	UC.2	The user can select a location that needs repairing	Must	BR.4	Koen, Adrie
UR.11	UC.2	The user can select a date that needs repairing	Must	BR.4	Koen, Adrie
UR.12		The user can filter locations by customer	should	BR.4	Koen, Adrie
UR.13		the user can see all errors on the repair page	should	BR.4	Koen, Adrie
UR.14		The user can see historic data on the repair page	could	BR.4	Koen, Adrie
UR.15		The user can ignore errors that do not need a repair	Should	BR.4	Koen, Adrie

1.5. Functional Software Requirements

ID	Use case	Requirement	Priority	Derived from	Origin	Notes
FSR.1		The client can toggle the navigation menu	Must	BR.4	Bart	
FSR.2		The client pushes the page content to the side of the navigation, when in desktop mode	Must	BR.4	Bart	
FSR.3		The client overlaps the content with the navigation menu when in tabled or mobile mode	Must	BR.4	Bart	
FSR.4		The server can log massages to a log file	Should		Chris, Gerrit	
FSR.5		The server can schedule tasks that need to be periodically executed	Should	BR.9	Bart, Chris	
FSR.6		The server can cancel tasks that are scheduled	Should	BR.9	Chris	
FSR.7		The error detection has a start and end date between it checks for errors	Must	BR.10	Chris	
FSR.8		The error detection inserts all errors in one batch	Must	BR.9	Chris, Tim	
FSR.9		The error detection ignores errors that have already been detected in a previous run	Should	BR.9	Chris	

1.6. Non Functional Software Requirements

ID	Requirement	Priority	Derived from	Origin
NFSR.1	The user interface is responsive. (mobile, tablet and desktop mode)	Should	BR.4	Bart
NFSR.2	The user interface is written in English	Must	BR.8	Bart
NFSR.3	The user interface is written in Dutch	Could	BR.8	Bart
NFSR.4	The ui uses the material design language	Must	BR.4	Bart
NFSR.5	The ui has a toolbar that is always on top	Must	BR.4	Bart
NFSR.6	The ui has a navigation menu on the left of the page	Must	BR.4	Bart
NFSR.7	The ui has a footer that is under the page content	Must	BR.4	Bart
NFSR.8	The ui focusses on the amount of open (unresolved) errors	Must	BR.4	Bart, Koen, adrie
NFSR.9	The user interface uses the following colors: Toolbar: DD033, Footer: 303E47, Primary: *3F51B5, Accent: E2E4E4 Warn: F44336, Background: white*	Must	BR.4	Bart

NFSR.10	The error detection runs for all active locations and devices	Must	BR.9	Chris, Bart
NFSR.11	The error detection only checks for errors between the opening and closing time of a location	Should	BR.9	Chris
NFSR.12	A detected error can be open, resolved and ignored	Must	BR.9	Chris
NFSR.13	The admin dashboard shows the open today and total of all footfall errors and footfall error types	Must	BR.4	koen, Adrie
NFSR.14	The admin dashboard shows the open today and total of all wifi errors and footfall error types	Should	BR.4	koen, Adrie
NFSR.15	The admin dashboard shows the open today and total of all kpi errors and footfall error types	Could	BR.4	koen, Adrie
NFSR.16	The (today) error page uses deferent tabs for footfall, wifi and kpi errors	Must	BR.4	koen, Adrie
NFSR.17	The (today) error page shows a list of all errors with the timestamp of the error, location name, device name and error type	Must	BR.4	koen, Adrie
NFSR.18	The today error page only shows errors from today and yesterday	Must	BR.4	koen, Adrie
NFSR.19	The server uses different types of models (database, domain)	Must		Chris
NFSR.20	The server uses DTO models	Should		Chris
NSFR.21	The location page shows the location name and the amount of errors (taday and total)	Must	BR.4	Koen, Adrie
NSFR.22	The location detail page shows a date with the amount of open and detected errors	Must	BR.4	Koen, Adrie

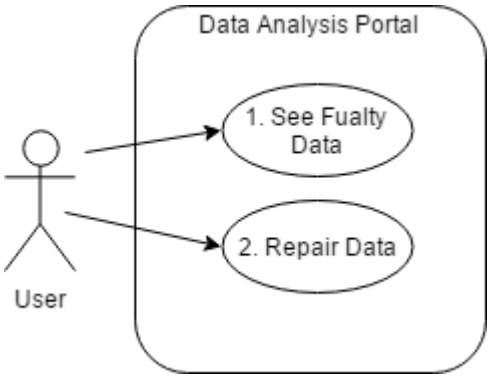
1.7. Technical Constraints

ID	Constraint	Origin	Date added	Notes
TC.1	The web application uses Webpack for script bundling	Chris, Tim	8-2-2017	
TC.2	The backend is made using ASP.NET Core	Chris	6-2-2017	
TC.3	The frontend of the application is made using Angular 2	Chris, Tim	17-2-2017	
TC.4	The frontend of the application uses angular material for all design	Bart, Tim	15-3-2017	

TC.5	The frontend of the application uses angular flex layout for mobile friendliness	Bart, Tim	15-3-2017	
TC.6	Not all components of Akka.NET are compatible with the .NET Core library at this moment	Tim	10-3-2017	
TC.7	Angular Material is currently not fully compatible with Angular Universal. Expected compatibility in version 4.1.x.	Tim	17-3-2017	Server side rendering does not work. Angular 4 is now available
TC.8	The Advantage API currently does not deliver all required data. For now, the entity framework will be used.	Chris, Tim	10-3-2017	

2. Use cases

2.1. Diagram



2.2. Use case Descriptions

1. See Faulty Data

Name	See faulty Data
ID	UC.1
Description	Show a list of all faulty data
Primary Actor(s)	User
Secondary Actor(s)	-
Precondition(s)	1. The user is logged in
Main Flow	1. The user clicks on the “Errors” navigation item in the sidenav 2. The client navigates to the error page 3. The client requests all errors from the rest api 4. the rest api gets all errors from the Advantage II database 5. The rest api sends all errors to the client 6. The client shows the list of errors to the user [Extend: UC.3]
Postcondition(s)	1. the application show a list of all detected errors
Alternative Flow	1a. Today errors 1. The user clicks on “Today Errors”

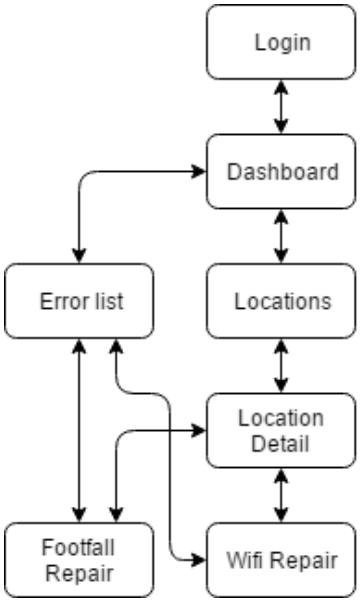
2. Repair Data

Name	Repair Data
ID	UC.2
Description	Repair data from devices
Primary Actor(s)	User
Secondary Actor(s)	-
Precondition(s)	1. The user is logged in
Main Flow	1. The user clicks on the location navigation item 2. The user clicks on a location 3. For the desired date, the user clicks on footfall repair 4. The system shows all devices and 5. The user selects all devices and time slots that need repairing 6. The user edits fields with faulty data 7. The user clicks on save 8. The system commits all changes to the database
Postcondition(s)	1. Count data is repaired.
Alternative Flow	3a. Wifi 1. The user clicks on wifi repair 3b. KPI 2. The user clicks on wifi KPI repair 6a. Auto repair data 1. The user clicks on auto repair data 2. The system makes a suggestion

3. Filter Errors

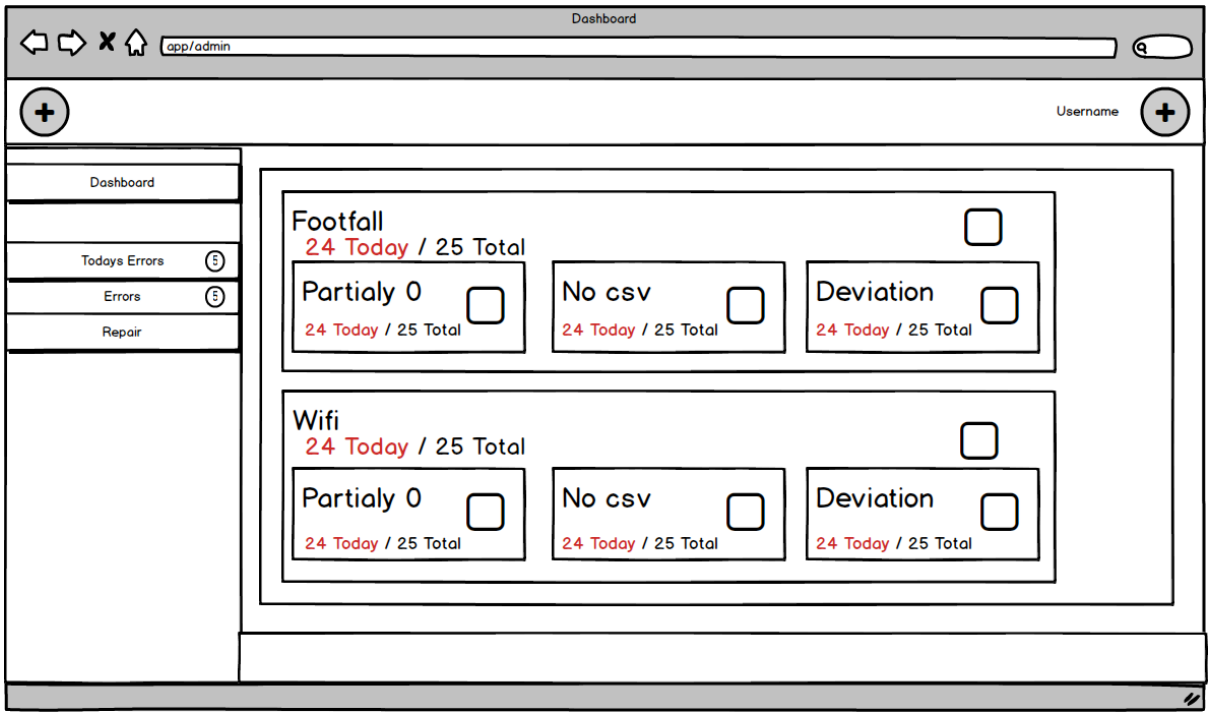
Name	Filter errors
ID	UC.3
Description	Filter the list of faulty data to the desired error type
Primary Actor(s)	User
Secondary Actor(s)	-
Precondition(s)	1. UC.1
Main Flow	1. The user clicks on the desired error types to show 2. The client clears the current list of errors from the table 3. The client adds every error to the table that has the desired error type 4. The table redraws itself with the new list of errors
Postcondition(s)	1. The shows only de desired errors
Alternative Flow	

3. Navigation map

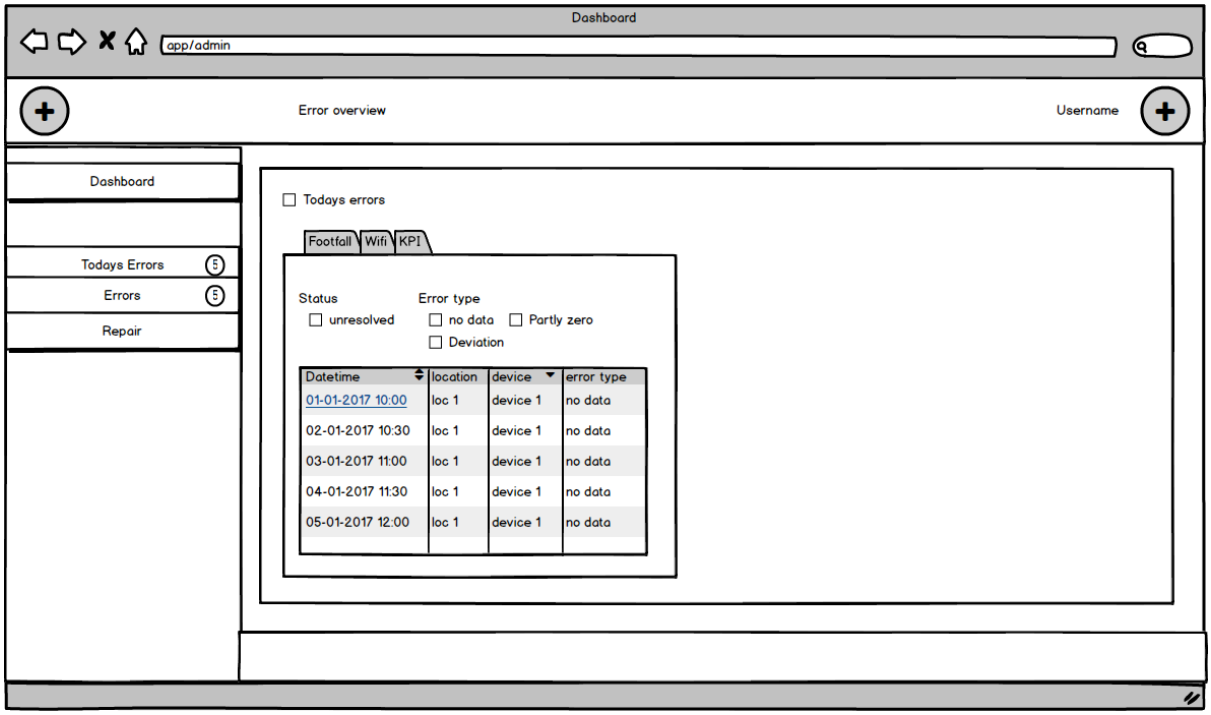


4. Mockups

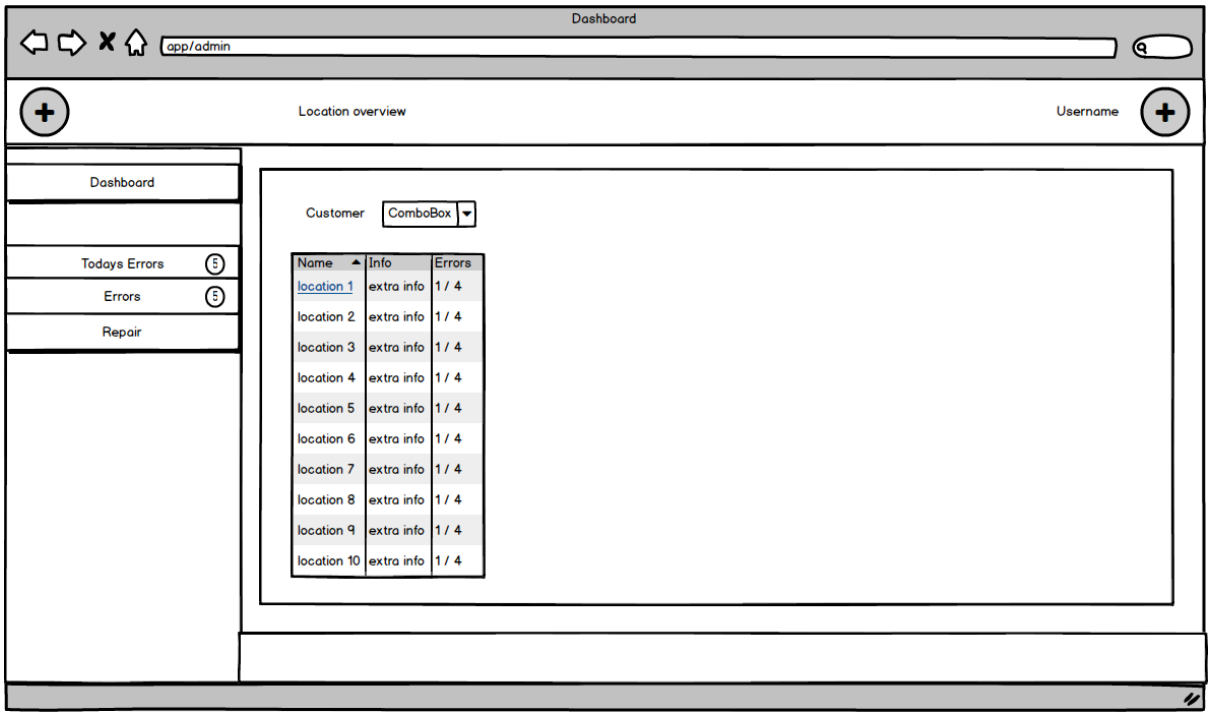
4.1. Dashboard



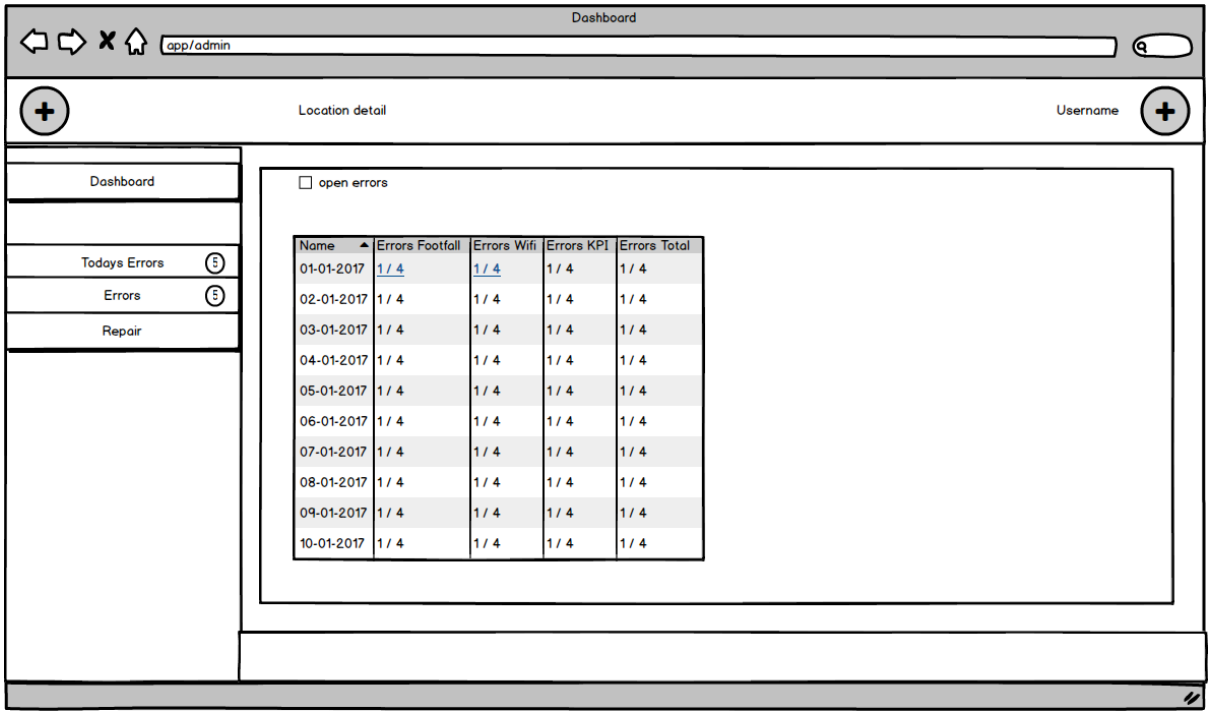
4.2. Errors Overview



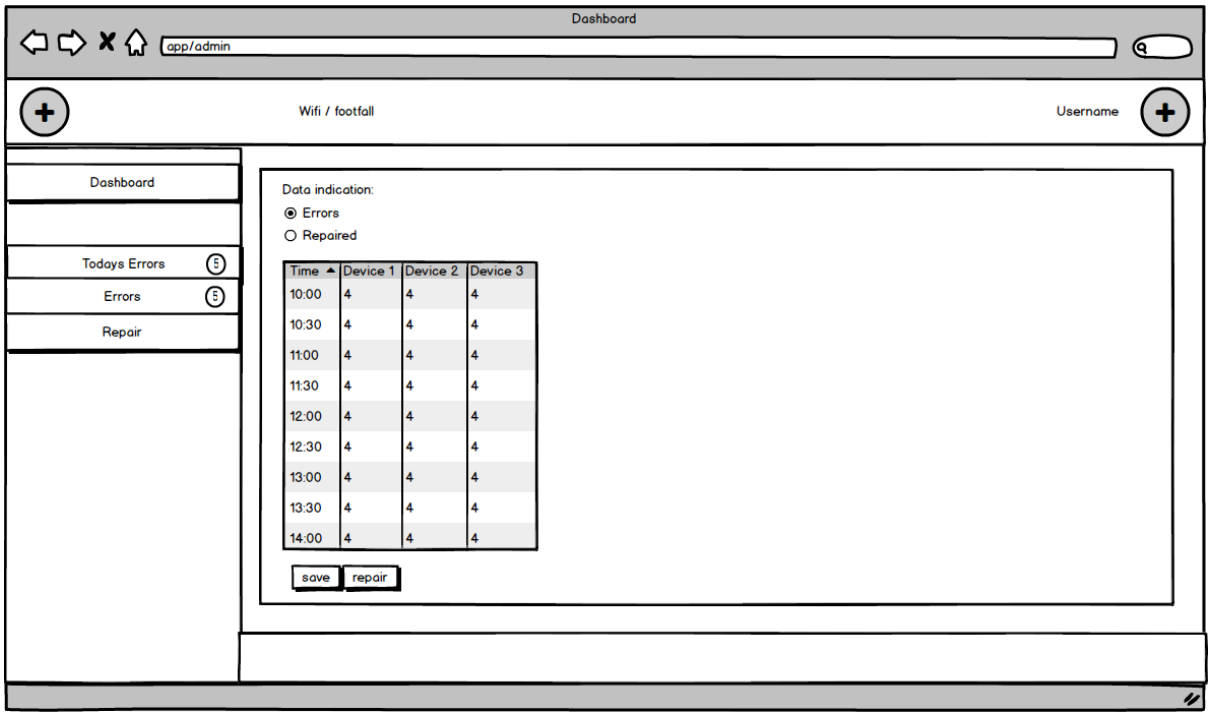
4.3. Location Overview



4.4. Location Detail



4.5. Repair Wifi/Footfall



Design Discipline

Data Analysis Portal

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Lead	Tim van Egmond
Opdrachtgever	Citytraffic
Plaats	Alphen aan den Rijn
Datum	19 Mei 2017
Versie	0.9

Table of Contents

1. Used Libraries/Frameworks

The Angular App has the following dependencies:

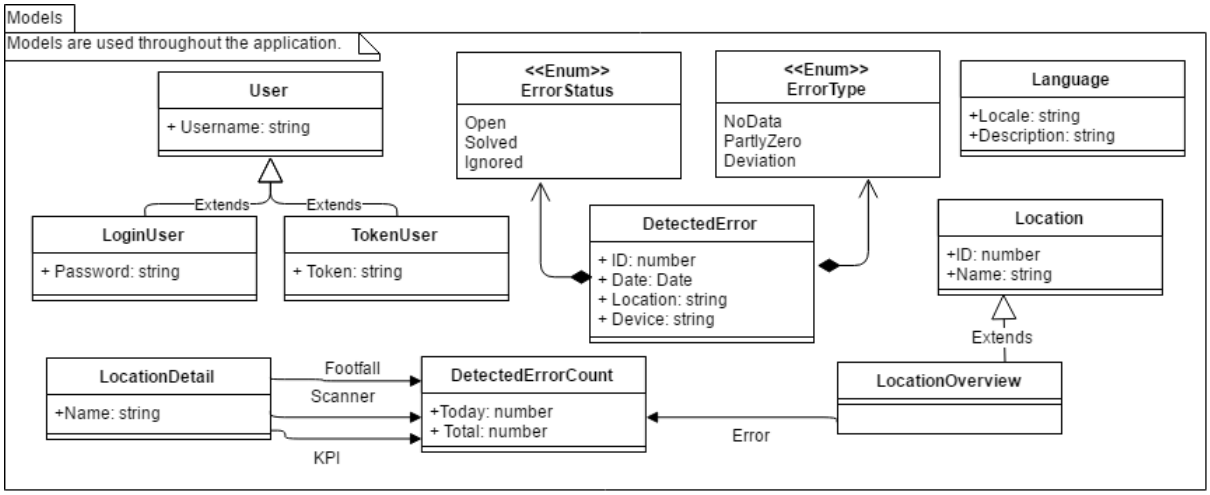
- Angular
- Angular Flex
- Angular Material
- Angular2-moment
- Material design icons
- Ngx-translate
- Ngx-datatable
- Screenfull

The ASP.NET Core webserver has the following dependencies:

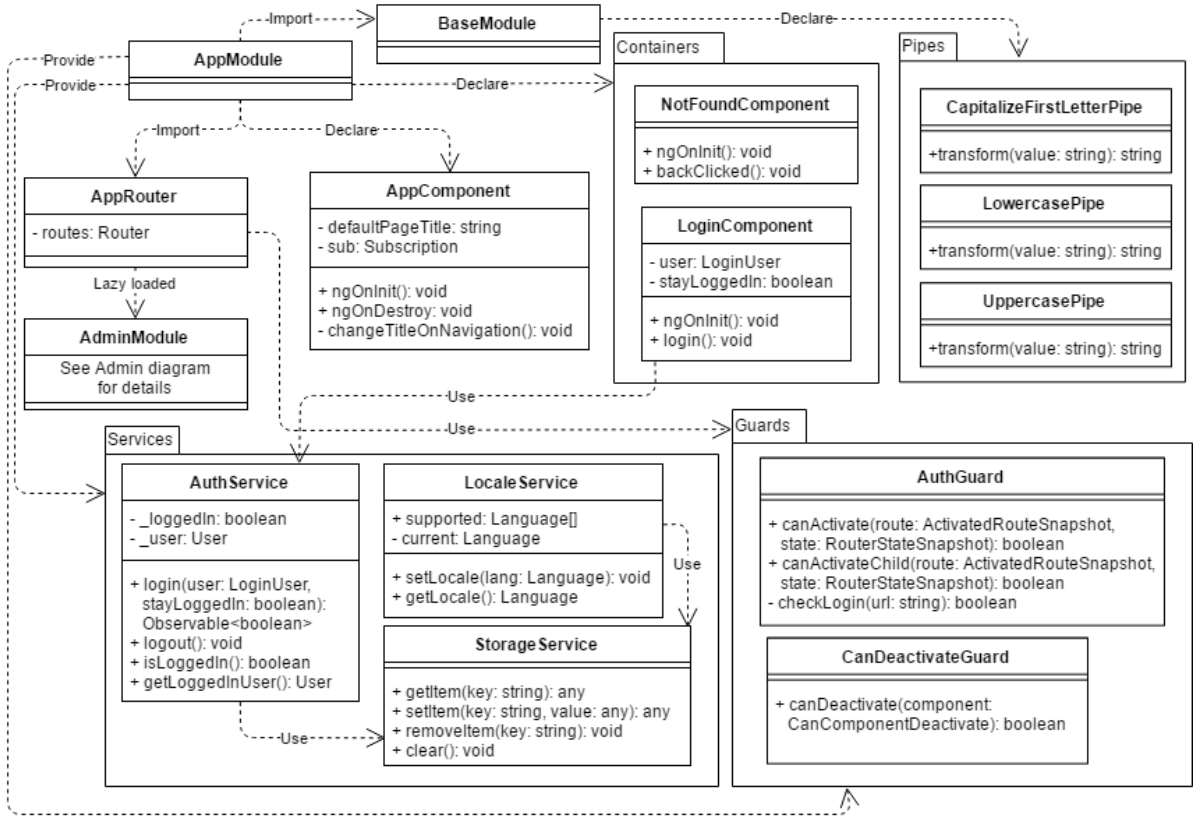
- Akka.NET
- Akka.NET dependency injection
- Akka.NET Serilogger
- Autofac
- Automapper
- Entity Framework Core
- Serilogger

2. Class Diagram

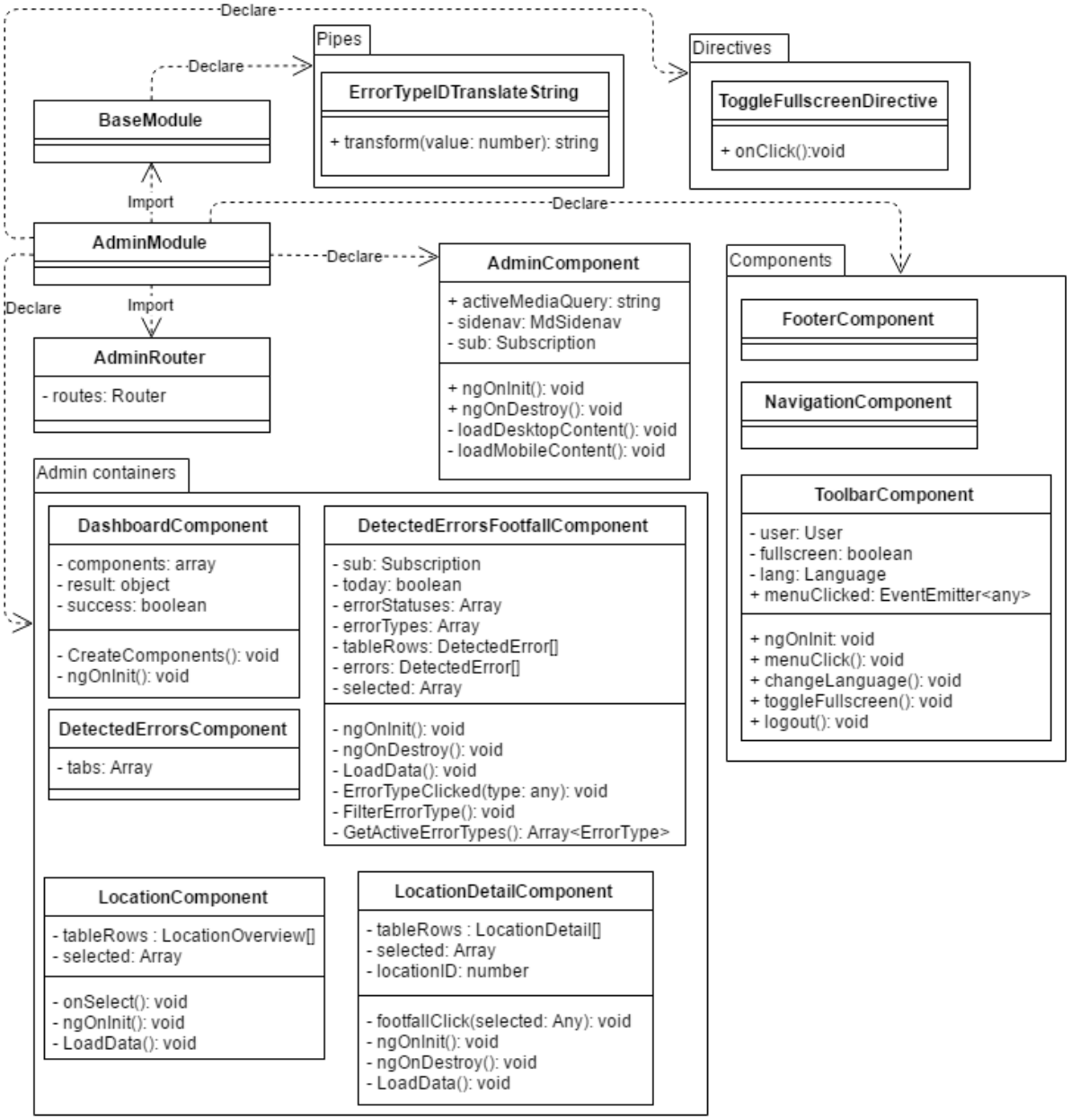
2.1. Angular Models



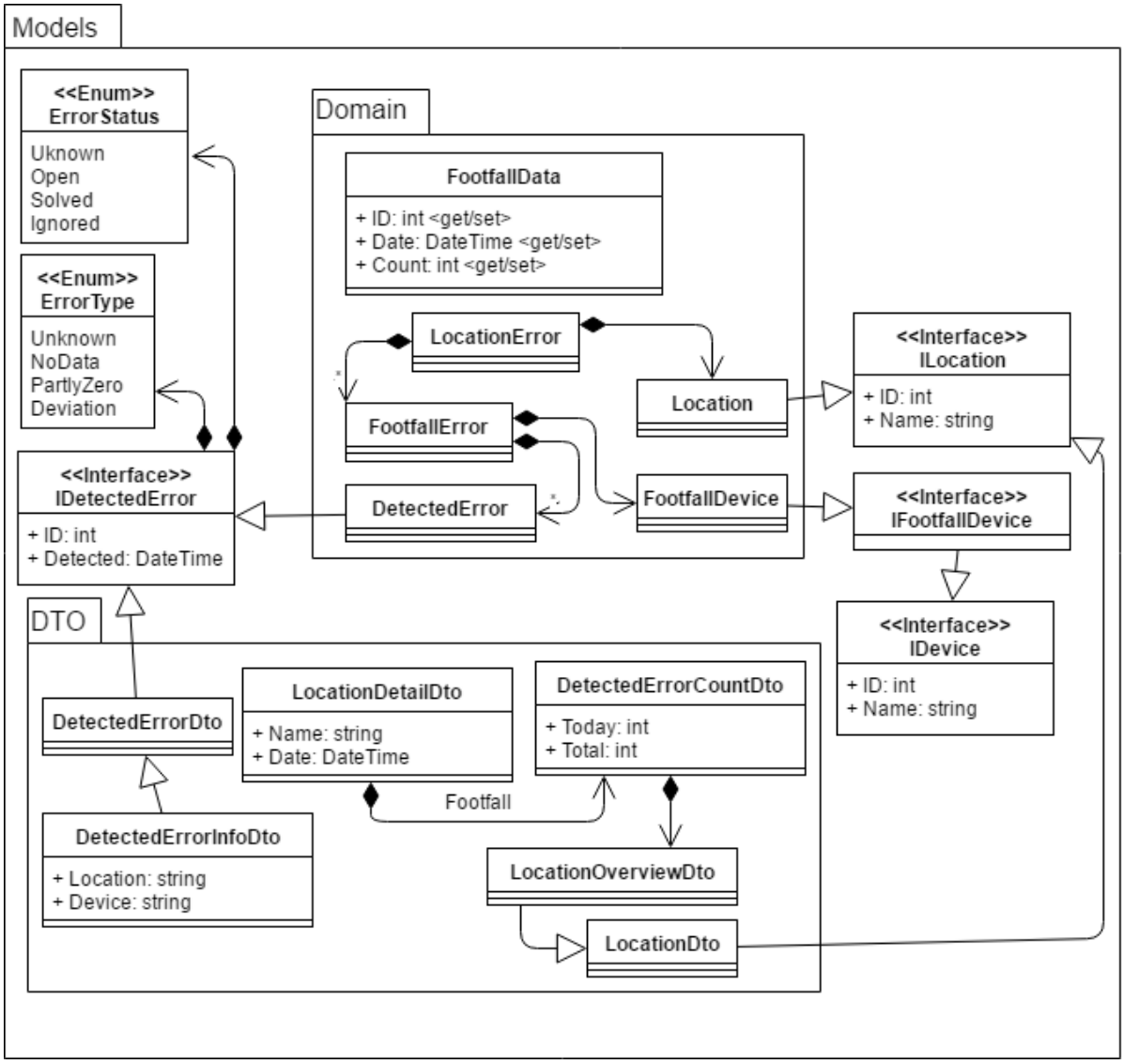
2.2. Angular Base App



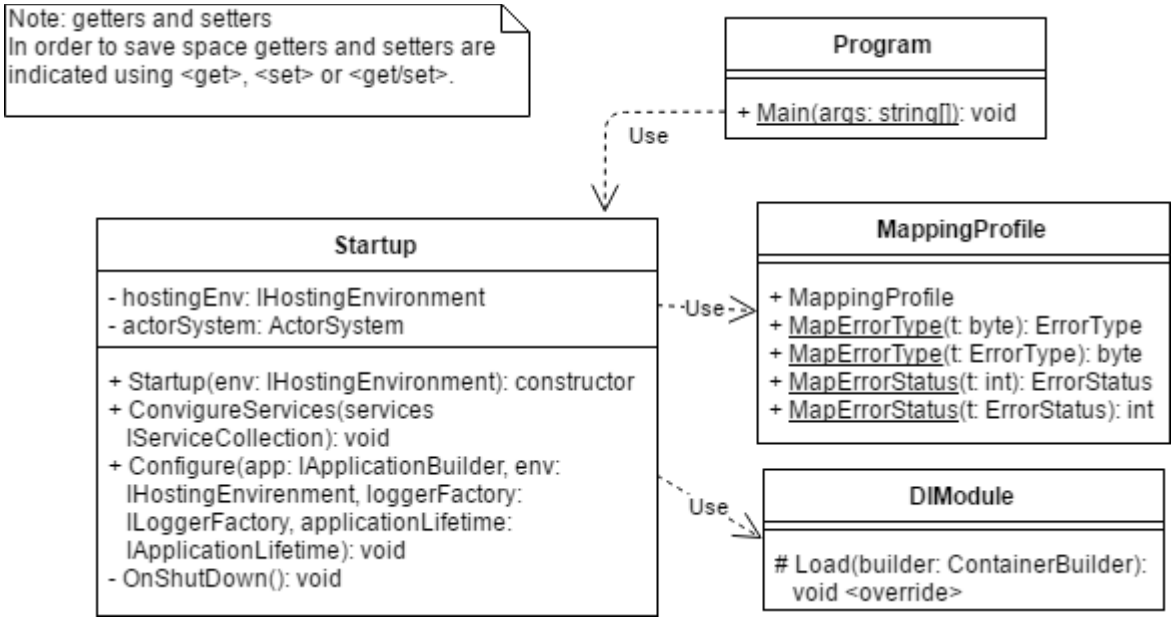
2.3. Angular Admin App



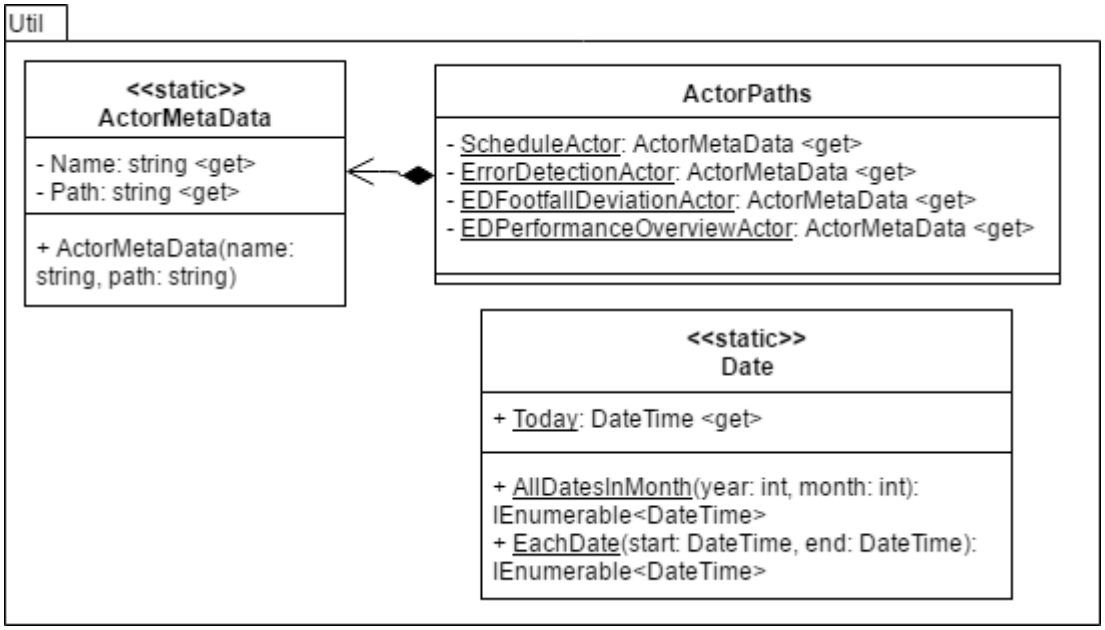
2.4. Server Models



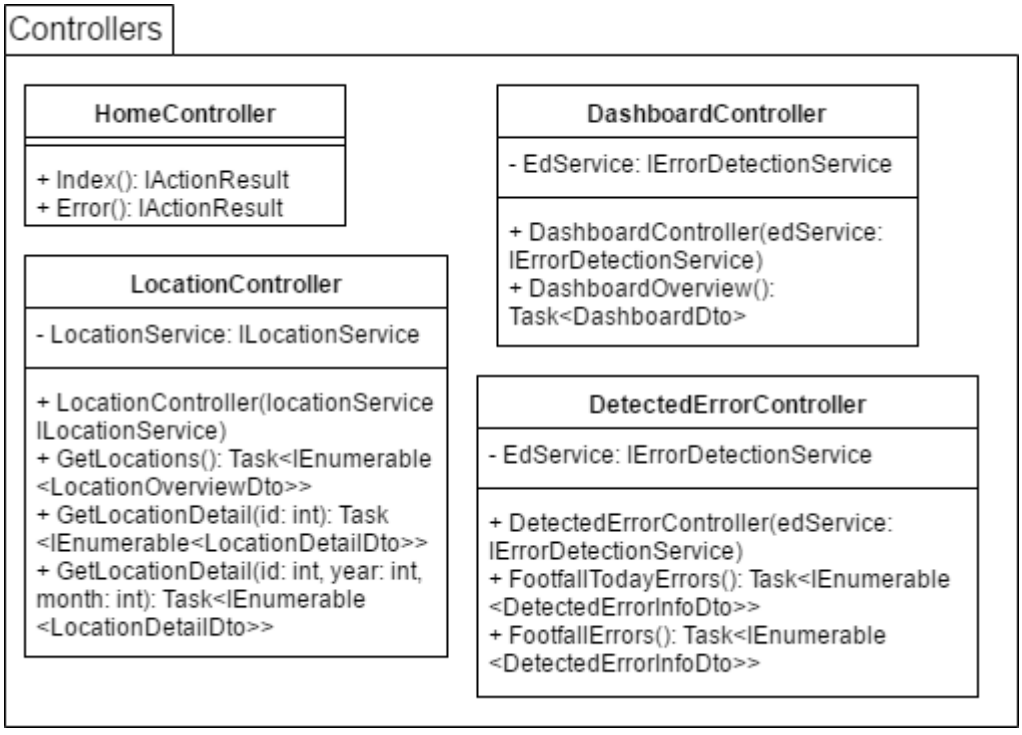
2.5. Server Startup



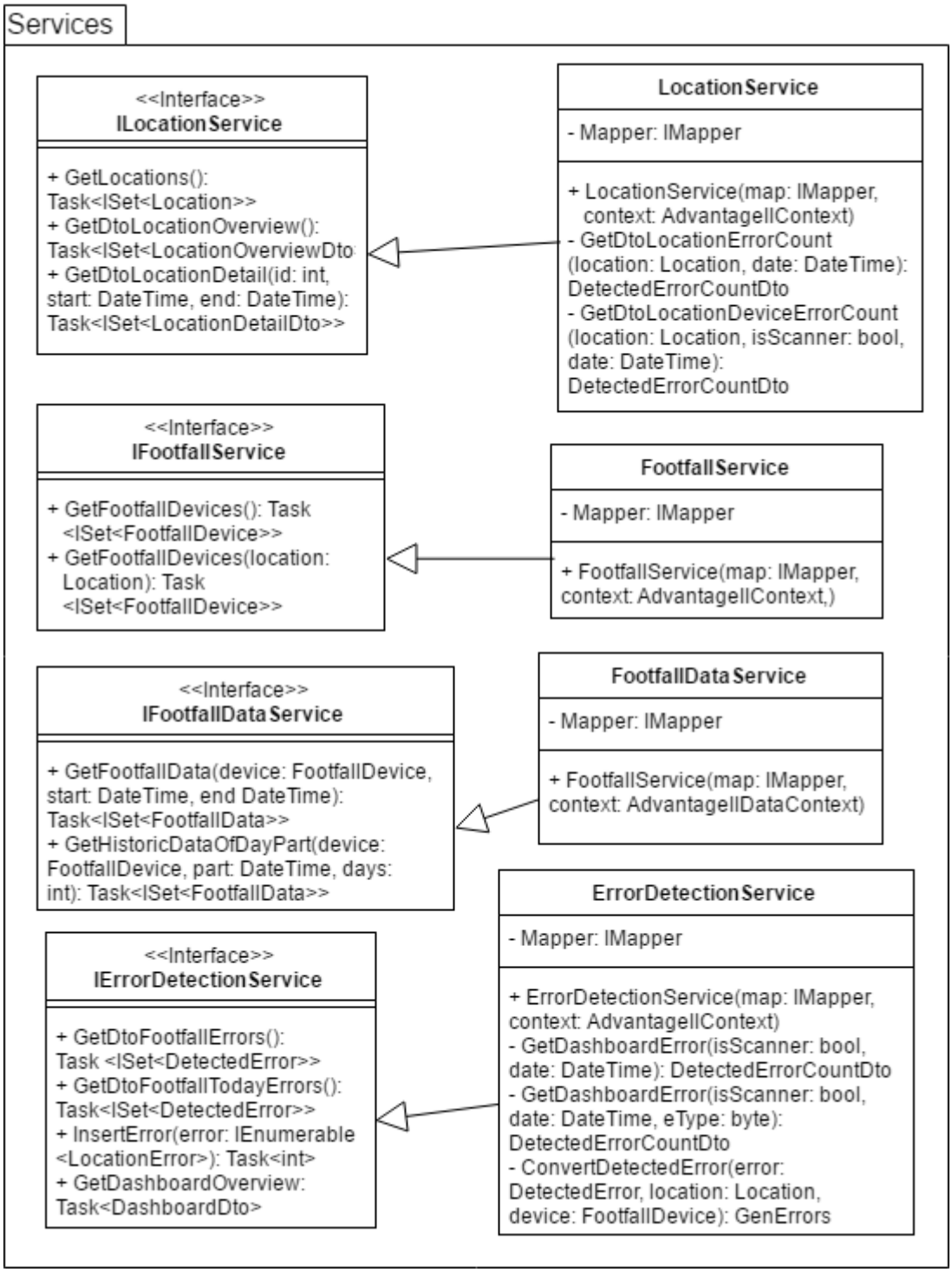
2.6. Server Utilities



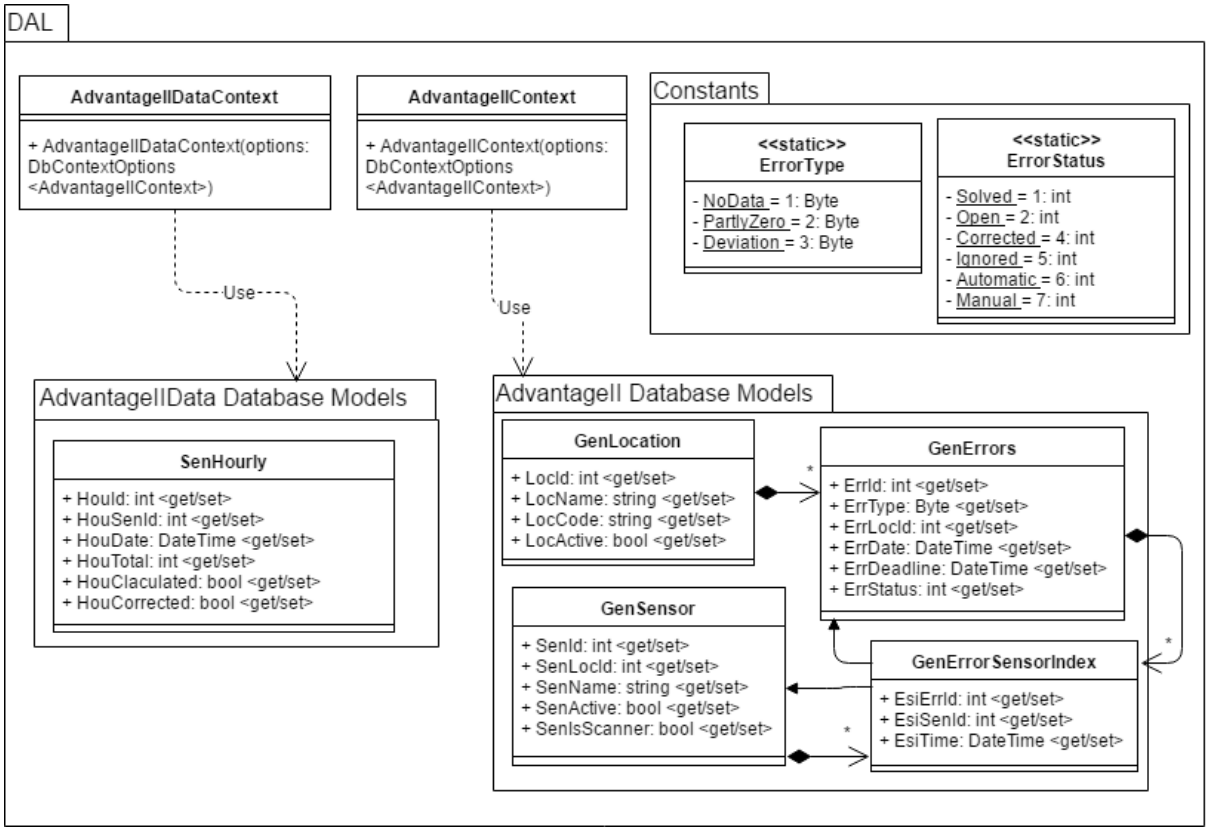
2.7. Server Controllers



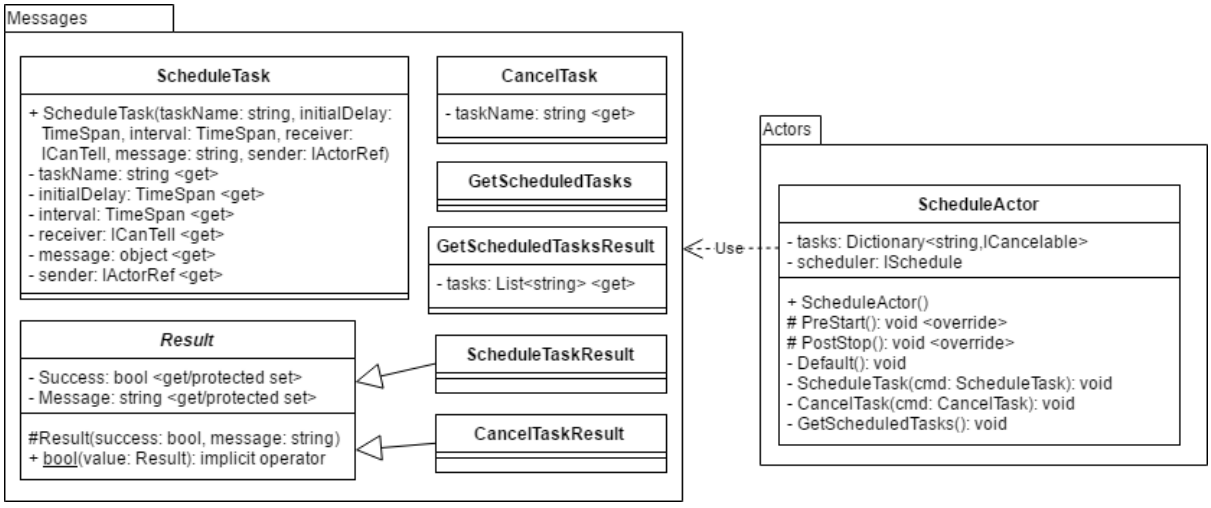
2.8. Server Services



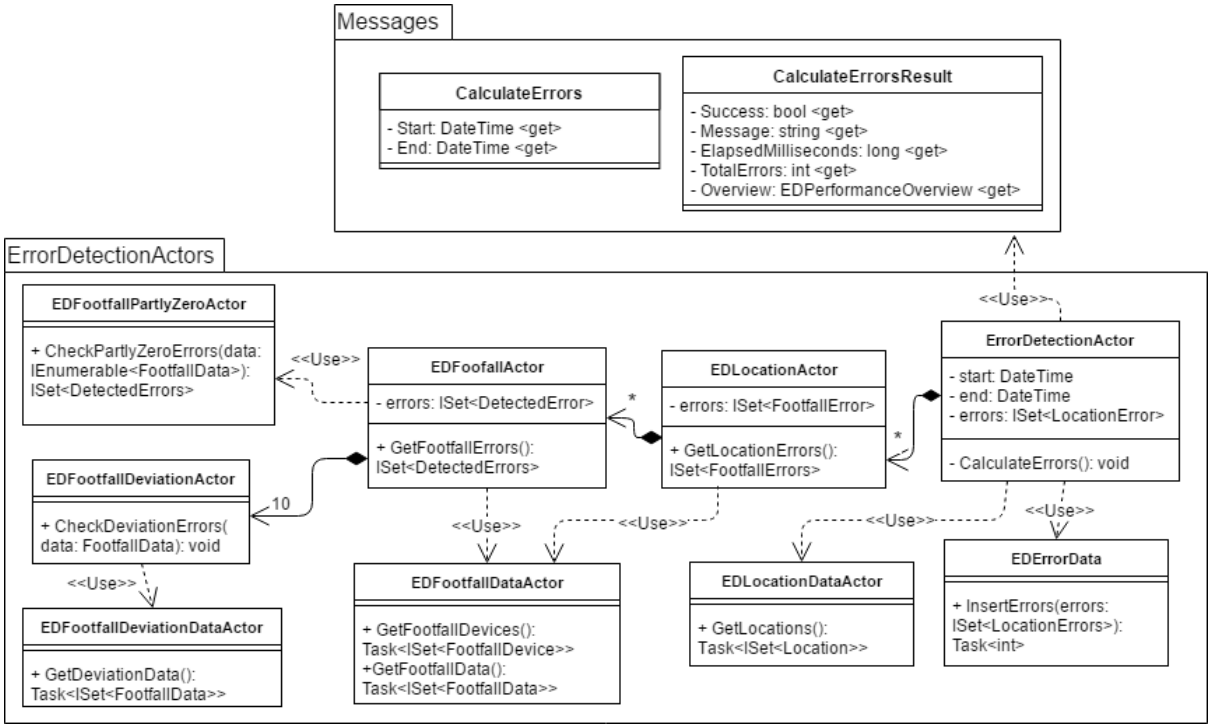
2.9. Server Data Access Layer



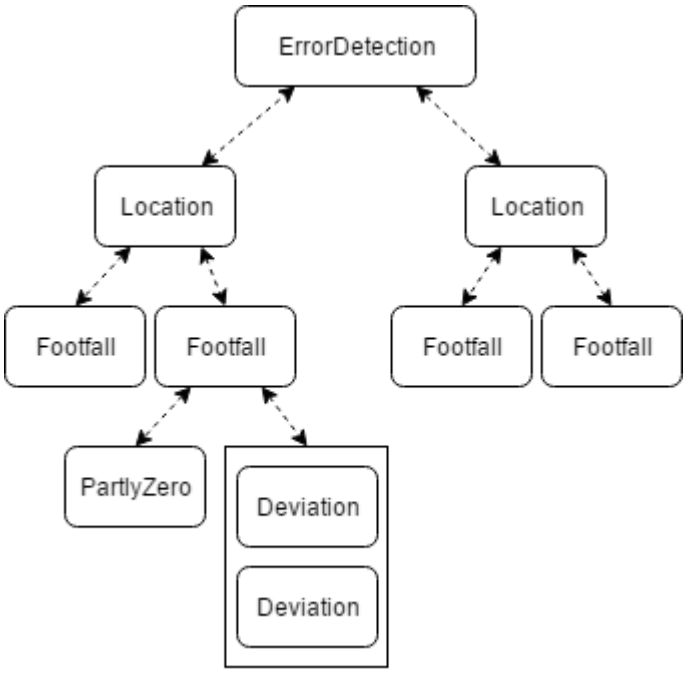
2.10. Server Schedule Actor



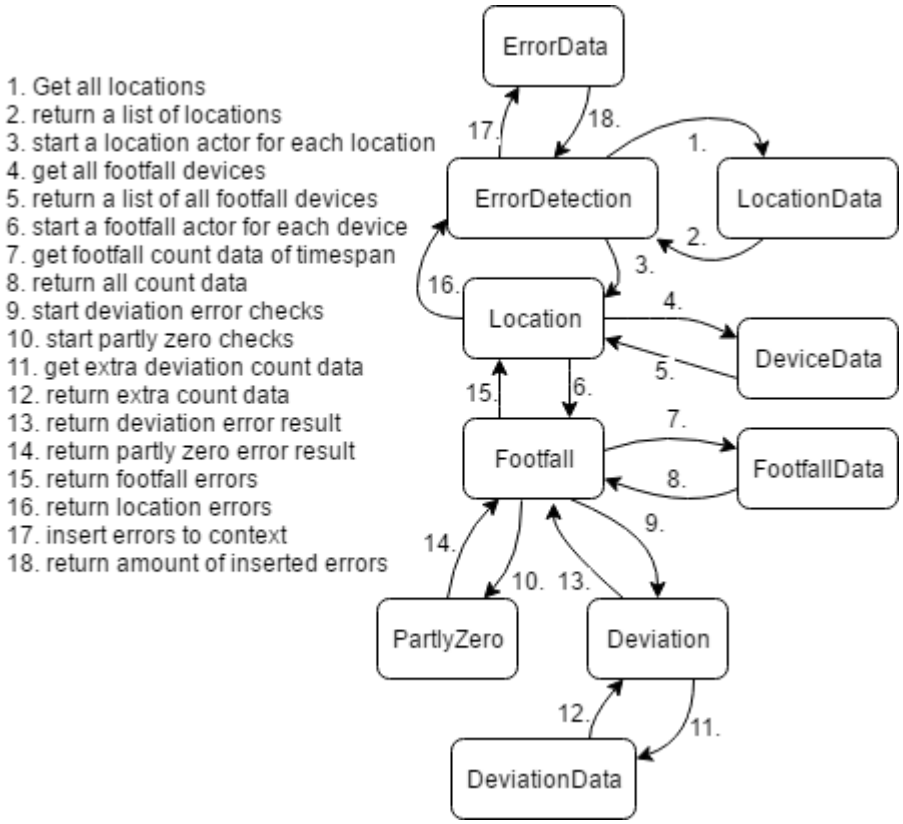
2.11. Server Error Detection



2.11.1. Actor Scaling

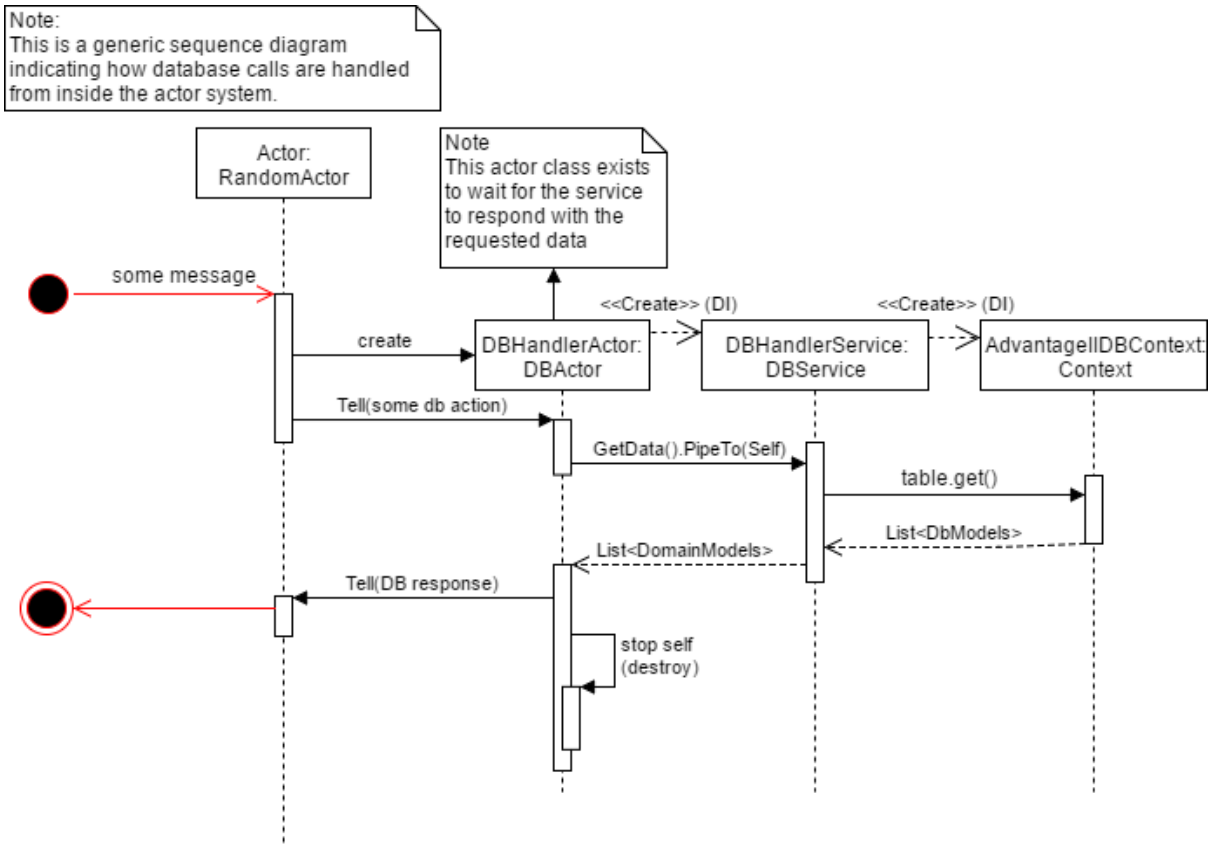


2.11.2. Message flow



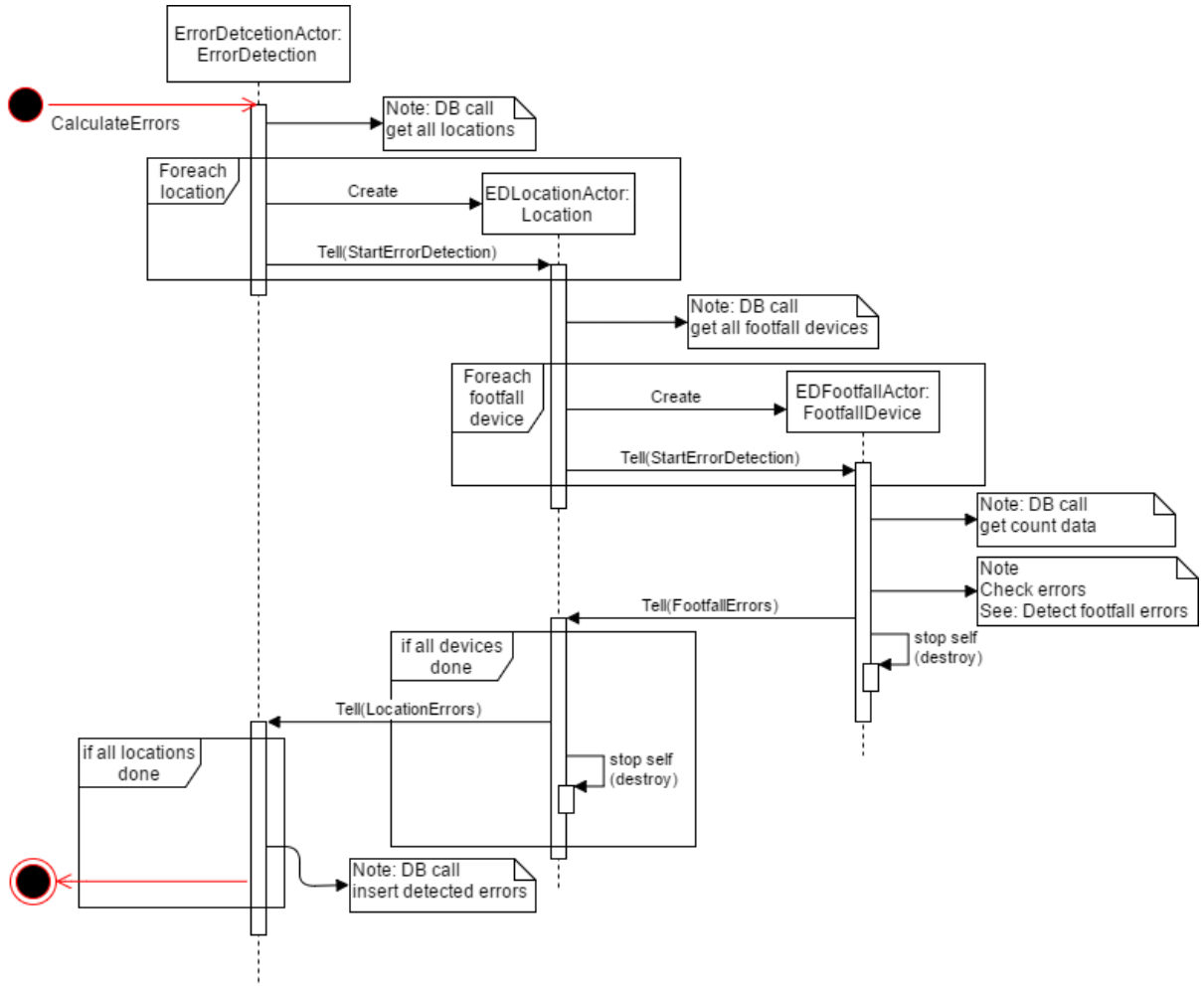
3. Sequence Diagram

3.1. Database Calls from actor

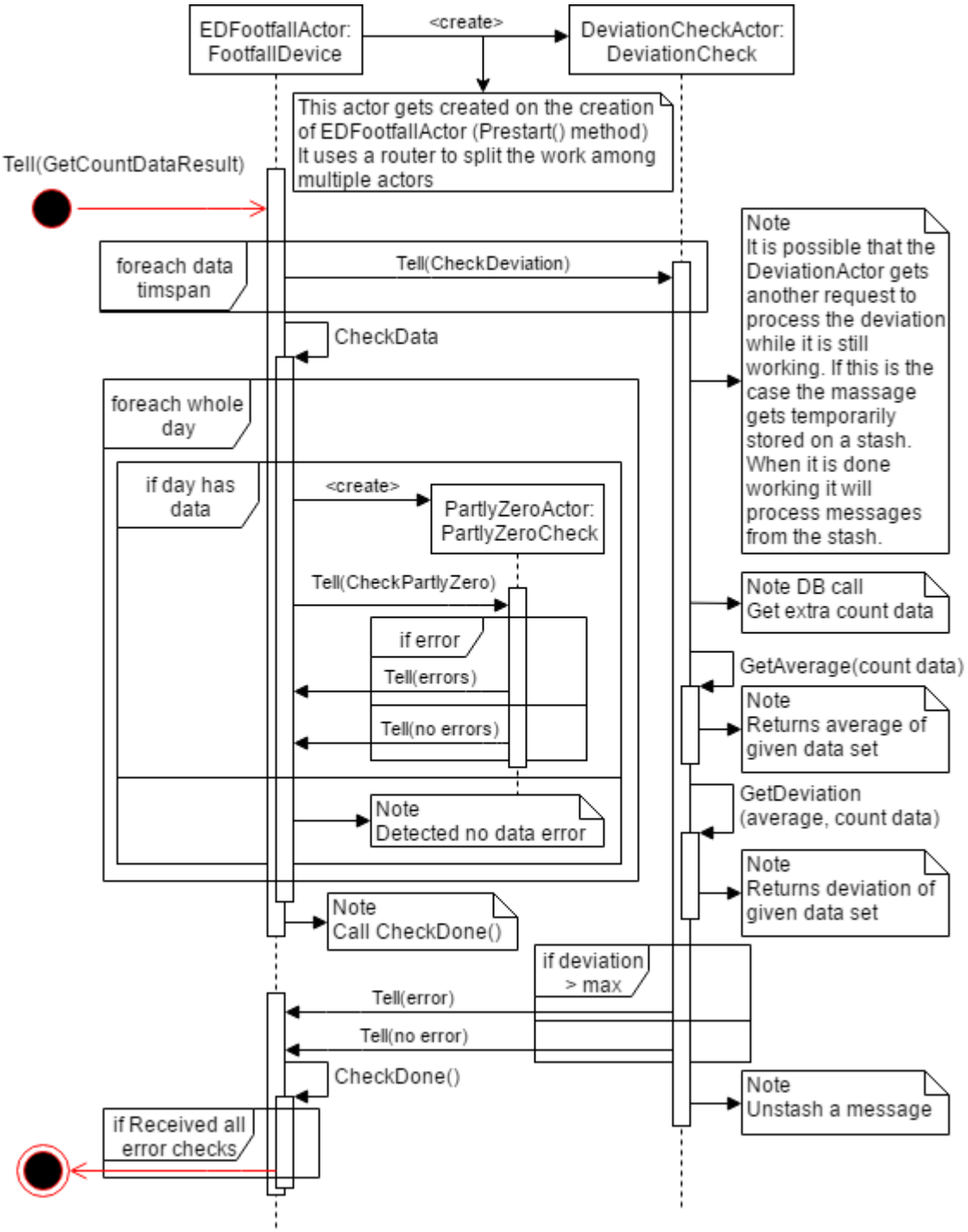


3.2. Error Detection

3.2.1. Startup

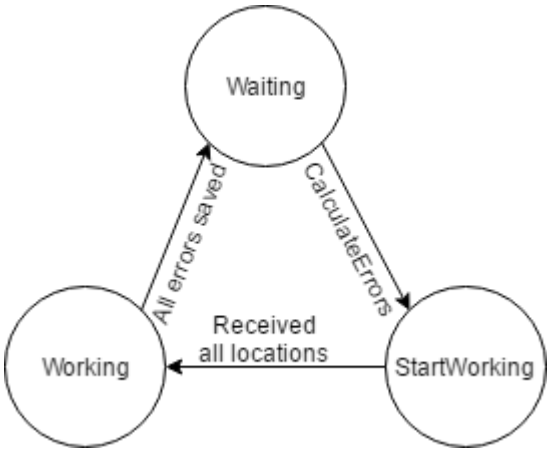


3.2.2. Detect Footfall Errors

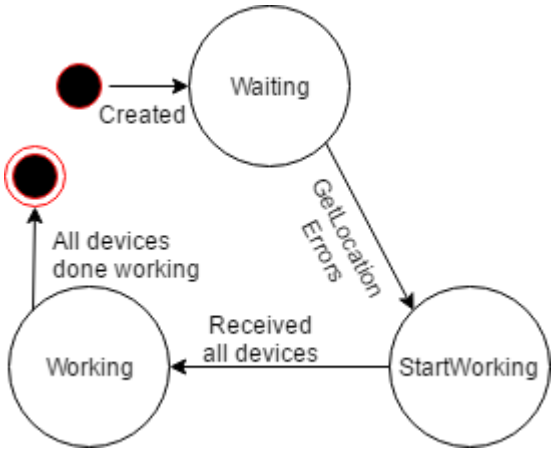


4. State Diagram

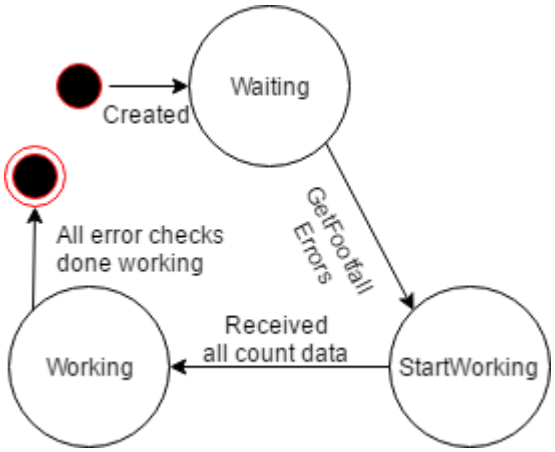
4.1. Error Detection Actor



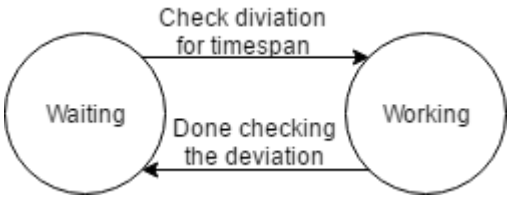
4.2. Error Detection Location Actor



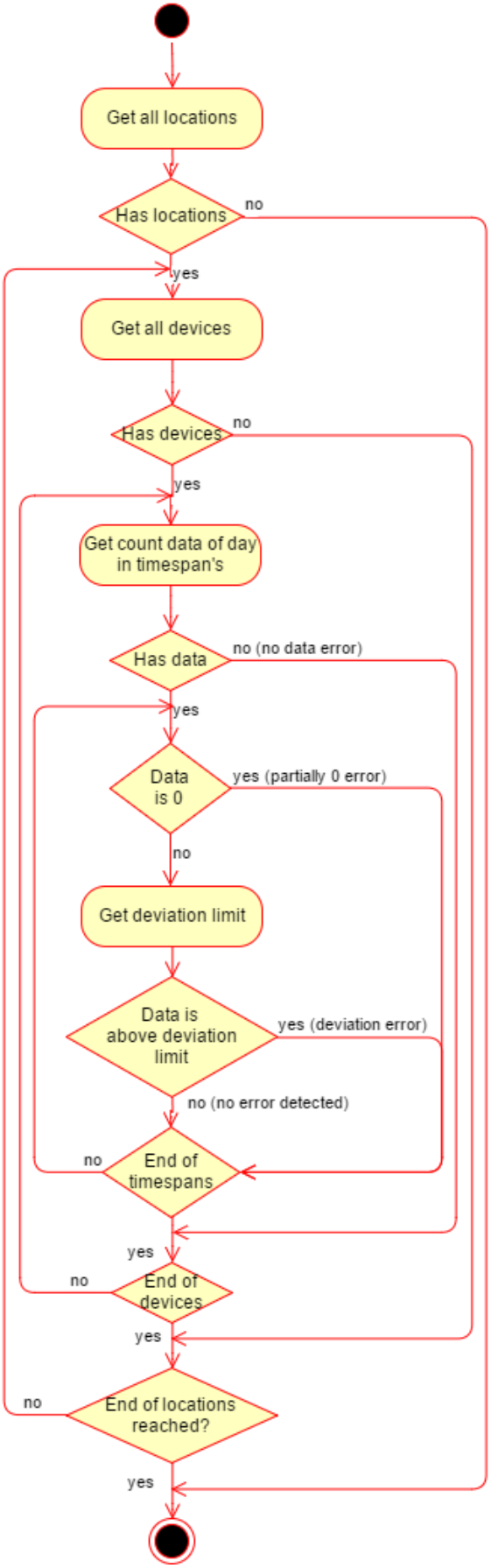
4.3. Error Detection Footfall Actor



4.4. Error Detection Footfall Deviation Actor



5. Error Detection Flowchart



Testrapport

Data Analysis Portal

Auteur	Tim van Egmond
Project	Data Analysis Portal
Project Lead	Tim van Egmond
Opdrachtgever	CityTraffic
Plaats	Alphen aan den Rijn
Datum	15 mei 2017
Versie	0.6

Inhoudsopgaven

1. Acceptatietest

Bij elke bijeenkomst met de stakeholders is een acceptatietest. Elk van de stakeholders krijgt hier de kans zelf door de applicatie heen te klikken en hierop feedback te geven. In dit hoofdstuk is voor elke bijeenkomst de feedback genoteerd en de gemaakte afspraken, betreft wijzigingen van het opgeleverde product.

1.1. Layout

Aanwezigen:

- Bart
- Koen

Afspraken:

- De kleuren van de applicatie omzette, de kleur codes zullen aangeleverd worden.
- Wijzig het logo naar een wit logo met een hogere resolutie.

1.2. Globale navigatie

Aanwezigen:

- Bart
- Koen
- Chris

Afspraken

- Op het dashboard zowel de errors van huidige dag als de totale aantal laten zien.
- Een apart item in de navigatie voor errors van de huidige dag. Hier kan dezelfde pagina voor gebruikt worden, enkel een extra filter dat de datum van de error controleert. Het is niet de bedoeling dat de gebruiker deze filter kan zien.
- [Needs Detail] Er moet een manier komen dat de geschiedenis van een locatie gezien kan worden. Dit zal de data analist ondersteunen in het bepalen van een correcte reparatie.
- [Needs Detail] De locaties moeten gefilterd kunnen worden per klant.
- [Needs Detail] Er moet een manier komen om te zien of een footfall device in of uit tellingen maakt.

1.3. Error detectie pagina

Aanwezigen:

- Bart
- Koen
- Chris

Afspraken:

- Een knop onder aan de tabel om gedetecteerde errors te negeren
- [Needs detail] Een knop om een error te repareren

1.4. Locatie pagina’s

Aanwezigen:

- Bart
- Koen
- Chris

Afspraken:

- Filter de locatie per locatie naam
- Filter de locatie per klant
- De locatie pagina's hebben een afscheiding tussen verschillende error types
- De reparatie pagina heeft tabs net als error pagina voor (footfall, scanner en kpi)

2. Unit tests

2.1. Webserver

Voor de webserver is het van belang dat de data dat verstuurd wordt richting de databases goed aankomt en intern verwerkt wordt. Om deze reden is de focus van de unittesten gericht op de services waar de webapplicatie gebruik van maakt. Naast de services worden ook de actoren binnen in de error detectie gecontroleerd. De unittesten zijn gemaakt met behulp van Visual Studio testtools en kunnen m.b.v. Visual Studio uitgevoerd worden.

2.1.1. Test setup

De webserver heeft meerdere dependencies welke elke opgezet moet worden, voordat een unittest uitgevoerd kan worden. Om dit proces zo gebruiksvriendelijk mogelijk te maken is er een Setup class aanwezig. Bij het uitvoeren van een test moet je ervoor zorgen dat van deze class de Initialize methode is aangeroepen. Zodra dit het geval is zorgt de class dat alles binnen de webserver getest kan worden.

De webserver maakt gebruik van twee databases Flucon en FluconData. Een groot deel van de webapplicatie is hiervan afhankelijk. Er is ervoor gekozen deze databases te mocken in sqlite, om zo een consistent test dataset te hebben, welke niet te groot is.

2.1.2. Services

Geen van de methodes binnen de services zijn complex. De huidige services bestaan enkel om data uit de database te halen en de models om te zetten van DBModels naar DomainModels.

Voor de services zijn de volgende tests gemaakt:

- LocationService_GetLocations
- FootfallService_GetFootfallDevices
- FootfallDataService_GetFootfallData
- ErrorDetectionService_InsertErrors

2.1.3. Error detection algorithm

Het error detection algoritme is gemaakt m.b.v. Akka.NET, om deze reden kan het niet op een traditionele manier geunittest worden. Vanuit een unit test kan de inhoudelijke data van het algoritme niet bekeken worden. Ook kunnen individuele methodes van een Actor niet aangeroepen worden. Je kan enkel een bericht steuren en bekijken wat het antwoordt is van de actor. De error detectie is zo opgezet dat een actor altijd antwoordt met de gevonden errors. Hierdoor zijn verschillende datasets meegegeven aan de EDFootfallActor, zodra de actor antwoordt geeft kan gekken worden of de verwachte errors gevonden zijn.

De volgende testen zijn uitgevoerd over actoren:

- EDFootfallActor_NoDataError
- EDFootfallActor_PartiallyZeroError
- EDFootfallActor_DeviationError

3. Performancetest Error Detection

3.1. Uitvoering

De performance testen zijn alle uitgevoerd op een test server met soortgelijke specs als op de live omgeving. De testomgeving maakt gebruik van een test database welke een exacte copy is van de live database. Tijdens de uitvoering van het algoritme zal m.b.v. Windows taakbeheer en resource monitor de gebruikte hardware resources geobserveerd worden.

De tijden die gemeten worden komen direct uit de actoren. Elke actor bezit een C# StopWatch object, zodra de actor start wordt de stopwatch aangezet en zodra de actor klaar is wordt de stopwatch stopgezet. De resultaten van de stopwatch worden verstuurd naar een speciale performance overview actor. Deze actor is verantwoordelijk om alle tijden bij te houden en zodra de error detectie klaar is de resultaten te printen in de log.

Om het opstarten van de error detectie gemakkelijker te maken, is er een speciale devolopment pagina beschikbaar in de client. Op deze pagina kan de error detectie gestart worden en zodra deze klaar is, zullen de resultaten op de pagina getoond worden.

De server waar de test op uitgevoerd wordt heeft de volgende specs:

OS	Windows Server 2012 R2 64 Bit
Processor	AMD Opteron 6128 (4 Cores @ 2.0 GHz)
RAM	16 GB
Moederbord	Intel 440BX

Scenario's

De performance testen zijn uitgevoerd met verschillende scenario's, oplopend in het aantal locaties. Elk van deze scenario's zal vijf keer uitgevoerd worden. Van deze resultaten zal vervolgens een gemiddelde afgeleid worden, om eventuele beschikbare hardware resource fluctuaties zo goed mogelijk eruit te kunnen filteren.

	Locaties	Footfall	Footfall met data	Time spans to check	Errors
1.	10	272	178	6516	3819
2.	25	702	354	11620	9418
3.	50	1346	564	17710	16524
4.	100	2049	700	22078	20804
5.	200	3131	1208	40480	33691
6.	300	3689	1548	54042	38834
7.	314	3745	1558	54474	38948

3.2. Resultaten

In onderstaande tabel zijn de gemiddelde resultaten te zien voor elk van de scenario's.

Scenario 1.	Count	Total	Avg	Min	max
Error detection	1	17239			
Locations	10	50580	5058	270	10105
Footfall	272	1600473	5884	166	9865
Get Locations	1	3			
Get Footfall	10	354	35	3	178
Get Footfall data	272	115507	424	36	2148
Get diviation data	6516	6295011	966	19	3331

Scenario 2.	Count	Total	Avg	Min	max
Error detection	1	29418			
Locations	25	205939	8237	17139	19482
Footfall	702	6510214	9273	640	19401
Get Locations	1	2			
Get Footfall	25	1044	41	4	507
Get Footfall data	702	300595	428	106	4638
Get diviation data	11620	22328272	1921	18	5323

Scenario 3.	Count	Total	Avg	Min	max
Error detection	1	51778			
Locations	50	870898	17417	1367	37095
Footfall	1346	20797939	15451	1135	37008
Get Locations	1	121			
Get Footfall	50	1821	36	4	503
Get Footfall data	1346	837452	622	200	5247
Get diviation data	17710	75930559	4287	12	9779

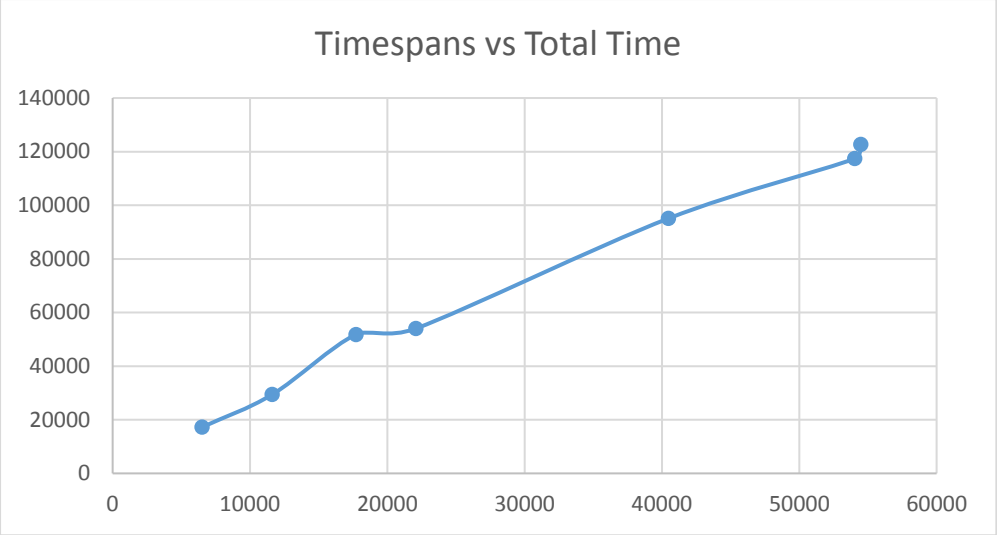
Scenario 4.	Count	Total	Avg	Min	max
Error detection	1	54018			
Locations	100	1425349	14253	1185	36940
Footfall	2049	26441704	12904	820	35703
Get Locations	1	8			
Get Footfall	100	49694	496	5	1171
Get Footfall data	2049	1805801	881	115	9224
Get diviation data	22078	100393582	4547	50	11381

Scenario 5.	Count	Total	Avg	Min	max
Error detection	1	95097			
Locations	200	6185638	30928	142	69405
Footfall	3131	82680816	26407	2416	68818
Get Locations	1	16			
Get Footfall	200	11429	57	11	100
Get Footfall data	3131	6257876	1998	433	3728
Get diviation data	40480	289349841	7147	20	14835

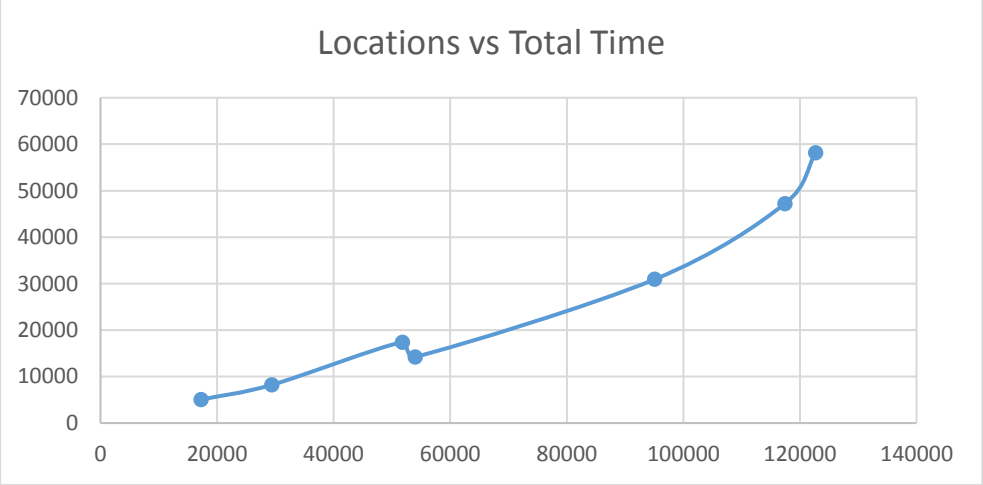
Scenario 6.	**Count**	**Total**	**Avg**	**Min**	**max**
Error detection	1	117427			
Locations	300	14165427	47218	192	87273
Footfall	3689	130729856	35437	6450	85367
Get Locations	1	14			
Get Footfall	300	25881	86	8	168
Get Footfall data	3689	14168861	3840	2928	4991
Get diviation data	54042	448371693	8296	26	22208

Scenario 7.	**Count**	**Total**	**Avg**	**Min**	**max**
Error detection	1	13599			
Locations	314	18262245	58160	151	108409
Footfall	3745	168159570	44902	10549	108019
Get Locations	1	21			
Get Footfall	314	32306	102	8	6711
Get Footfall data	3745	14007131	3740	2142	25811
Get diviation data	54474	553561102	10161	12	25809

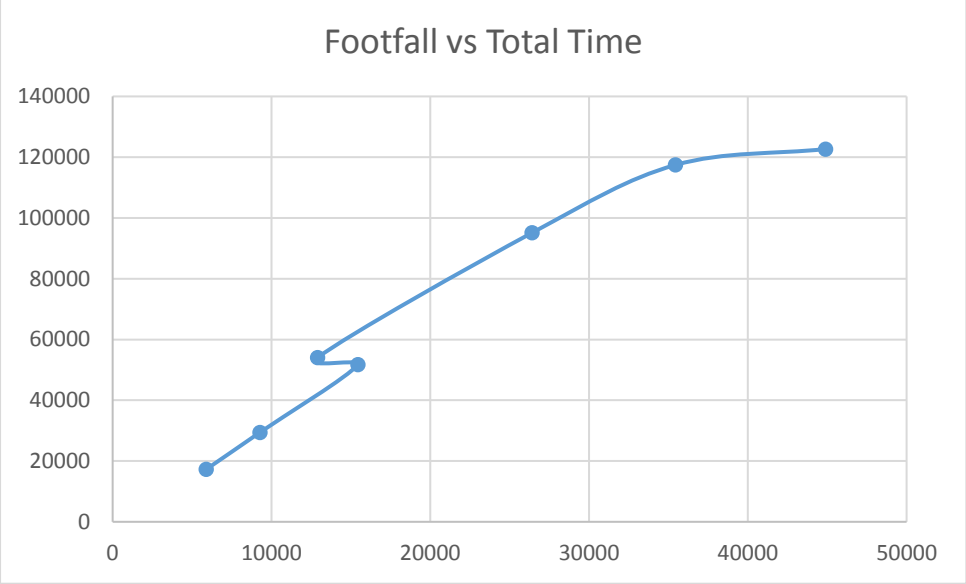
Het belangrijkste verschil tussen de verschillende scenario's is de hoeveelheid data dat gecheckt moet worden voor errors. Dit is het gedeelte van het systeem dat de meeste tijd in beslag neemt. Hieronder zijn drie grafieken weergeven elk met de gemiddelde tijd van zowel de hoeveelheid timespans, locaties of footfall devices tegenover de totale tijd.



Figuur 3: Tijdspanne (data dat gecheckt moet worden) tegenover de totale tijd



Figuur 4: Locatie tegenover de totale tijd



Figuur 5: Footfall devices tegenover de totale tijd

Voor elke van de grafieken is een lineaire stijging te zien, wel ziet elke grafiek er anders uit. Dit wordt veroorzaakt door het verschil in data. Niet elke locatie heeft dezelfde hoeveelheid footfall devices en niet elke footfall device heeft dezelfde hoeveelheid timespan tellingen die voor errors gecheckt moeten worden. Uit deze lineaire stijging kan geconcludeerd worden dat het hier om een O(N) algoritme gaat.

3.2.1. Hardware resources

Tijdens het runnen van de error detectie algoritme is zowel de processor utilisatie als het ram gebruik geobserveerd. Deze observatie is uitgevoerd tijdens het uitvoeren van scenario 7. De 0 sec markering is de status van de webserver voordat het error detectie algoritme is opgestart, de idle state. De webapplicatie maakt gebruik van een in memory cache, waar o.a. het entity framework gebruik van maakt. Hierdoor zal het ram gebruik snel omhooggaan en actief blijven, het wordt mogelijk niet opgeruimd.

	CPU Usage (%)	Threads	RAM (MB)
0 sec	0	18	77
10 sec	20	44	700
30 sec	11	43	1536
60 sec	33	41	1851
90 sec	38	43	1946
120 sec	27	44	981
150 sec	5	43	974
180 sec	0	25	974
210 sec	0	18	974

Zoals in bovenstaande tabel is te zien gebruikt de webserver vanuit zichzelf niet veel resources, wel loopt het gebruik van resources snel op nadat de error detectie gestart is. Op peak momenten zijn er 44 threads actief, hoewel het niet zeker is hoeveel van deze threads actief bezig zijn met het detecteren van errors kan wel vrij zeker afgeleid worden dat het er in ieder geval 20 zijn. De CPU heeft op de testserver 4 cores, het is mogelijk dat Akka meer threads aanmaakt in het geval dat het aantal cores omhoog gaat.

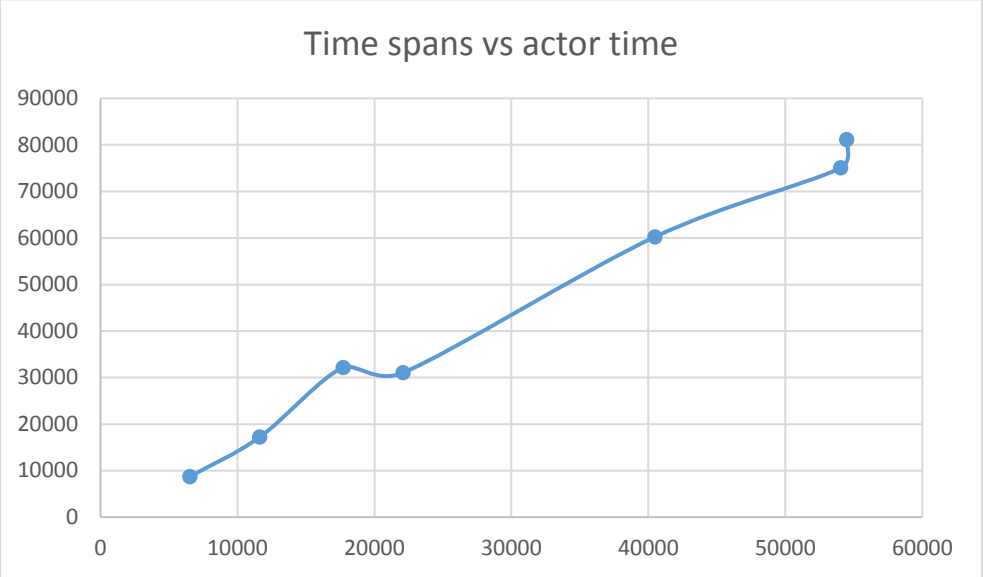
De CPU usage is over het algemeen laag, rond de 30%, maar deze resultaten geven niet het gehele beeld. Het was duidelijk te zien dat op piek momenten de CPU wel 90 tot 100 procent bereikte dit heeft vermoedelijk te maken met de wachttijd van de database. Het kan voorkomen dat de database op bepaalde momenten veel deviation data query's behandeld en retourneert. In dit geval zal de deviation error checks veel actoren gelijktijdig aan het werk hebben, wat pieken veroorzaakt.

Tijdens de error detectie kan je goed zien dat het geheugen gebruik oploopt, met ongeveer 2 gigabyte. Aan de hand van de status nadat de error detectie klaar is (210 sec) is af te lijden dat bijna 1 gigabyte daarvan in gebruik is door de in memory cache. Als de actoren zelf van het error detectie algoritme 1 gigabyte in gebruik hebben, voor alle interne communicatie is dit geen probleem. De server heeft meer dan genoeg RAM en RAM is makkelijk uitbreidbaar.

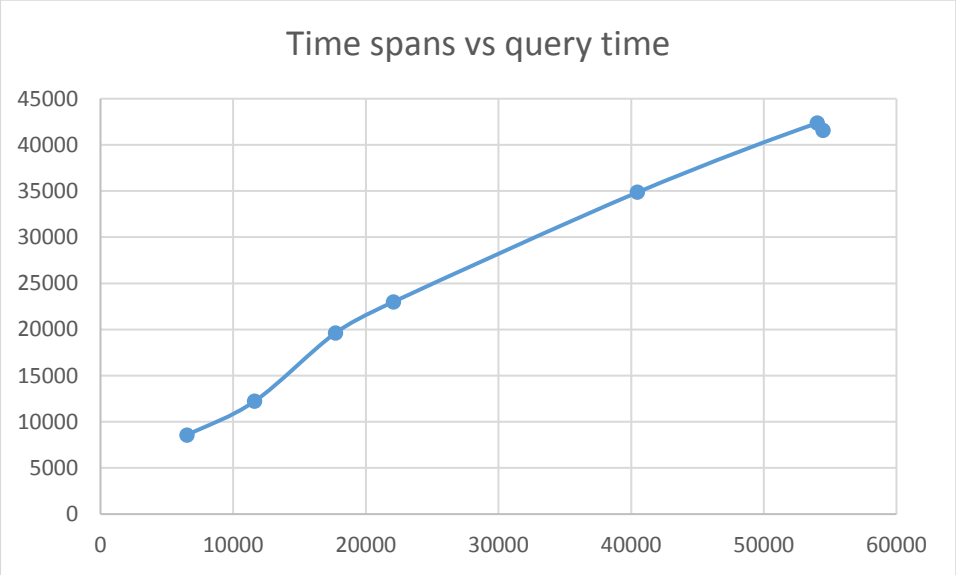
3.2.2. Tijd verdeling

Als gekeken word naar de tijdverdeling tussen de verschillende werkzaamheden (onderstaande grafieke), van het algoritme, is te zien dat zowel de actor activiteit als de query tijd gelijk oploopt. De

reden dat de query tijd lineair oploopt is simpel te verklaren. Het is een relationele database, deze staan erom bekent slecht schaalbaar te zijn. Wat wel opvallend is, is de lineair actor stijging, een mogelijke reden hiervoor is te vinden in 3.3.



Figuur 6: Actor tijdsduur tegenover de hoeveelheid data



Figuur 7: Query tijdsduur tegenover de hoeveelheid data

3.3. Verwachtingen

Op dit moment heeft het algoritme een lineaire stijging, toch verwacht ik dat dit beter kan. Mijn verwachting is dat de actoren beter moeten kunnen schalen, waardoor de tijd dat de actoren bezig zijn (werken) minder zal worden. Met Akka kunnen applicatie gemaakt worden, welke goed van alle CPU cores gebruik kunnen maken, mijn verwachting is dan ook dat de huidige processor niet genoeg cores heeft om dit effect te zien. Huidige servers kunnen gemakkelijk meer dan 20 cores hebben in een CPU. Het zou ook kunnen dat Akka meer threads nodig heeft (huidig is 20), met meer threads kunnen meer actoren tegelijk actief zijn op de processor.

3.4. Conclusie

Het algoritme heeft een duidelijke lineaire stijging. Dus aan de hand van deze resultaten kan gezegd worden dat het algoritme O(N) tijd in gebruik neemt. Het feit dat het algoritme er minder dan drie minuten over doet bij deze hoeveelheid data is al erg goed en is beter dan verwacht. De CPU usage is wel zorgwekkend, niet omdat het gemiddeld 30% is, maar juist omdat er pieken in zitten. Wel is het belangrijk dat er meer tests uitgevoerd worden. Deze tests moeten draaien op een betere server met meer CPU cores. Ook is het belangrijk dat de dataset veel uitgebreider wordt.

DE HAAGSE HOGESCHOOL

Faculteit IT & Design

Delft
Den Haag
Zoetermeer

Evaluatieformulier afstuderen

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)

Student: Tim van Egmond
Periode: februari – juni 2017
Bedrijf c.q. instelling: City Traffic
Bedrijfsmentor: Christiaan Floor
Plaats: Alphen aan den Rijn
Datum: 24-05-2017

1. Heeft de student zich zelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?

Ja, hij heeft zich het domein/de business die relevant was voor het project snel eigen gemaakt.

2. Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega's, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?

Communicatief is Tim erg goed. Wanneer je met hem in overleg bent merk je dat hij altijd goed oplet. En zijn onderbouwing voor bepaalde keuzes is altijd sterk. Ook zijn documentatie is altijd goed op orde en heeft dat gedurende de hele periode ook bijgehouden.

3. Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?

- **Qua verantwoordelijkheid** **goed** / voldoende / matig / onvoldoende
- **Qua zelfstandigheid** **goed** / voldoende / matig / onvoldoende
- **Qua planmatig werken** **goed** / voldoende / matig / onvoldoende
- **Qua creativiteit** **goed** / voldoende / matig / onvoldoende
- **Qua productiviteit** **goed** / voldoende / matig / onvoldoende
- **Qua samenwerken met collega's** **goed** / voldoende / matig / onvoldoende
- **Qua draagvlakontwikkeling** **goed** / voldoende / matig / onvoldoende
- **Qua inspelen op bedrijfscultuur** **goed** / voldoende / matig / onvoldoende
- **Qua rekening houden met de specifieke context van het bedrijf** **goed** / voldoende / matig / onvoldoende
- **Qua het op gang brengen van de nodige veranderingen** **goed** / voldoende / matig / onvoldoende

4. Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?

Ik vind zijn kennis & kunde op een hoog peil en het voldoet zeker aan de verwachting. Natuurlijk is er nog een hoop te leren. Maar in de softwareontwikkelingsbranche is het opdoen van nieuwe kennis part of the job.

5. Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?

Goed, deze producten waren altijd van hoog niveau.

6. Bent u tevreden over het opgeleverde (eind)product?

- **In hoeverre heeft u gekregen wat is afgesproken?**

  Het eerste deel van het project was onderzoek. Daarbij moest eerst het grootste risico getackeld worden, namelijk de performance & schaalbaarheid van het ‘detectie algoritme’ (zie documentatie van Tim voor meer detail).

  Daarna konden we verder en heeft hij een goede basis opgezet voor een detectie & reparatie tool voor onze data. Een basis die zeker verder moet worden uitgebreid, maar het doel was een basis opzetten. Dus we hebben gekregen wat we hadden verwacht.

- **In hoeverre voldoet het (eind)product aan uw verwachtingen?**

  Het voldoet aan de verwachting van ‘een goede basis’.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**

  De bruikbaarheid is zeer groot. Het is nog niet direct inzetbaar als live product, maar het is een goede start. Door Tim zijn goede documentatie & duidelijke opzet van het programma schat ik ook in dat het goed onderhoudbaar. Daarbij geldt het wel dat een juiste overdracht (mits noodzakelijk) cruciaal is om de onderhoudbaarheid te behouden.

- **Wat gebeurt er met het opgeleverde (eind)product?**

  Dit wordt doorontwikkeld tot een live bruikbaar product.

- **Kunt u direct met het opgeleverde product aan de slag?**
  Ja zoals aangegeven wordt het direct doorontwikkeld.

7. Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?

Nee niet in de context van dit document.

8. Bent u bereid een volgende keer weer uw medewerking te verlenen aan het beschikbaar stellen van een afstudeerplaats (graag met toelichting)**?**

Ja, we zijn erg enthousiast over Tim zijn prestatie. Dus we zien graag meer mensen met zijn kwaliteiten bij ons komen voor een opdracht.