

Faculteit Ingenieurswetenschappen en Architectuur  
  
Schakelprogramma tot Master of Science in de   
Industriële Wetenschappen: Informatica

Vakoverschrijdend project (E767014)

Projectgroep 3

VOP Verkeerscentrum:

Software-Analyse

De Bock Jelle

Floré Brent

Stofferis Jeroen

Vandemoortele Simon

Vervenne Jan



Academiejaar 2015-2016



Faculteit Ingenieurswetenschappen en Architectuur  
  
Schakelprogramma tot Master of Science in de   
Industriële Wetenschappen: Informatica

Vakoverschrijdend project (E767014)

Projectgroep 3

VOP Verkeerscentrum:

Softwareanalyse

De Bock Jelle

Floré Brent

Stofferis Jeroen

Vandemoortele Simon

Vervenne Jan

Academiejaar 2015-2016

Inhoud

[Inleiding 4](#_Toc448588163)

[Analyse 5](#_Toc448588164)

[1 User Stories 6](#_Toc448588165)

[2 Use cases diagram 7](#_Toc448588166)

[3 Use cases 8](#_Toc448588167)

[3.1 Use Case 1: Meld abnormaliteiten 8](#_Toc448588168)

[3.2 Use Case 2: Bekijk reistijden 9](#_Toc448588169)

[3.3 Use Case 3: Controleer data 10](#_Toc448588170)

[3.4 Use Case 4: Bekijk actuele verkeerssituatie 11](#_Toc448588171)

[3.5 Use Case 5: Vergelijk verkeersinformatie 12](#_Toc448588172)

[3.6 Use Case 6: Data ophalen en hergebruiken 13](#_Toc448588173)

[Ontwerp 14](#_Toc448588174)

[4 Inleiding 15](#_Toc448588175)

[5 Providerapplicatie 15](#_Toc448588176)

[5.1 State diagram 15](#_Toc448588177)

[5.2 Design Class Diagram (DCD) 15](#_Toc448588178)

[5.2.1 Detailoverzicht scrapers 16](#_Toc448588179)

[5.2.2 Detailoverzicht domein 16](#_Toc448588180)

[5.2.3 Detailoverzicht repository/controller 16](#_Toc448588181)

[5.2.4 Legende 16](#_Toc448588182)

[5.3 Algoritmes 17](#_Toc448588183)

[5.3.1 API call maken 17](#_Toc448588184)

[6 Webapplicatie 19](#_Toc448588185)

[6.1 Algoritmes 19](#_Toc448588186)

[6.1.1 Home pagina bezoeken 19](#_Toc448588187)

[7 Databank 20](#_Toc448588188)

[7.1 Entity-relationship diagram (ERD) 20](#_Toc448588189)

[8 Scope definitie 22](#_Toc448588190)

[9 Unit testen 22](#_Toc448588191)

[10 Integration testen 22](#_Toc448588192)

[11 Usability testen 22](#_Toc448588193)

[11.1 Inleiding 24](#_Toc448588194)

[11.2 Installatie 24](#_Toc448588195)

[11.2.1 Server 24](#_Toc448588196)

[11.2.2 Desktop 34](#_Toc448588197)

[11.2.3 Deployment 35](#_Toc448588198)

[12 Aandachtspunten sprints 37](#_Toc448588199)

[12.1 Sprint 2 37](#_Toc448588200)

# Inleiding

Deze softwareanalyse is er gekomen naar aanleiding van het vakoverschrijdend eindproject. Hierbij is het de bedoeling om in teamverband een reëel project uit te werken. Het project dat ons toegewezen werd is in opdracht van stad Gent.

Dit document is het resultaat van de software analyse. Het is specifiek gericht tot de vereisten van de te ontwerpen software en de daarbij horende functionaliteiten. Het bevat onder meer het ontwerpdocumenten van het ontwikkelingsproces.

Het doel van deze analyse is om de vereisten van de klant te gaan vertalen is deelproblemen en deze zo beknopt mogelijk te formuleren. Het herwerken van deze bundel van vereisten is essentieel om een goeie oplossing te bekomen.

Hoofdstuk één behandelt de user stories. Deze pagina bevat een korte beschrijving van de vereisten, gecombineerd met hun gewicht en prioriteit. De twee volgende hoofdstukken, use case diagram en use cases, beschrijven dan weer concrete handelingen die de gebruikers uitvoeren op het systeem. Hoofdstuk vier en vijf bevatten respectievelijk het klassediagram en het databaseschema. Het document eindigt met enkele aandachtspunten voor de volgende sprint.

Analyse

# User Stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prioriteit | Gewicht | Opgenomen als issue | User | Story |
| 1 | 10 | x | Gebruiker | De reistijden van trajecten afkomstig van verschillende providers kunnen vergelijken. |
| 2 | 3 | x | Gebruiker | De historiek van de reistijden kunnen vergelijken. |
| 3 | 4 | x | Operator | De actuele verkeersituatie raadplegen op een kaart. |
| 4 | 4 | x | Operator | De ideale reistijden van trajecten vergelijken met de huidige reistijden. |
| 5 | 5 |  | ontwikkelaar | Ophalen van verschillende gegevens via een API. |
| 6.1 | 6 |  | Operator | Raadplegen van een dashboard om trajecten op verschillende momenten, verschillende trajecten,… te vergelijken. |
| 6.2 | 3 |  | Operator | Verwittiging ontvangen bij ernstige gebeurtenissen. |
| 6.3 | 2 |  | Operator | Instellen na hoeveel minuten een boodschap verstuurd wordt als er vertragingen zijn. |

# Use cases diagram



# Use cases

## Use Case 1: Meld abnormaliteiten

|  |  |
| --- | --- |
| **Use Case naam** | Meld abnormaliteiten |
| **Use case nummer** | 1 |
| **Toepassing** | Back-end |
| **Beschrijving** | Een abnormaliteit wordt automatisch gemeld via de beschikbare kanalen. |
| **Primaire Actoren** | het systeem |
| **Stakeholders** | De gebruikers van de communicatiekanalen en operator. |
| **Precondition** | nvt |
| **Trigger** | Een bepaalde hoeveelheid vertraging werd overschreden op een welbepaald traject. |
| **Basisverloop** | 1. Het systeem detecteert een abnormale vertraging. |
|  | 2. Het systeem selecteert de beschikbare kanalen die verwittigd moeten worden. |
|  | 3. Het systeem stuurt een melding naar de beschikbare kanalen. |
|  | 4. Het systeem rapporteert zijn tussenkomst. |
| **Alternatief verloop** | nvt |
| **Domeinregels** | 1. De drempel waarop een vertraging abnormaal wordt kan door een operator ingesteld worden. |
|  | 2. Mogelijke kanalen zijn |
|  | - Twitter |
|  | - Opvallende plaats op de website |
|  | - Native apps |

## Use Case 2: Bekijk reistijden

|  |  |
| --- | --- |
| **Use Case naam** | Bekijk reistijden |
| **Use case nummer** | 2 |
| **Toepassing** | front end |
| **Beschrijving** | De actor bekijkt de reistijden die voor hem relevant zijn. |
| **Primaire Actoren** | Gebruiker, operator |
| **Precondition** | nvt |
| **Postcondition** | De actor heeft een overzicht van de reistijden. |
| **Trigger** | nvt |
| **Basisverloop** | 1. De actor wenst een reistijd te bekijken. |
|  | 2. Het systeem retourneert een overzicht van de beschikbare trajecten. |
|  | 3. De actor kiest het gewenste traject. |
|  | 4. Het systeem retourneert het overzicht van het geselecteerde traject. |
|  | 5. De actor bekijkt de gewenste reistijden. |
| **Alternatief verloop** | 5A. De actor wenst uitgebreide informatie van het traject. |
|  | 6. Het systeem retourneert de beschikbare opties. |
|  | 7. De actor kiest de gewenste opties. |
|  | 8. Het systeem retourneert de uitgebreide informatie van het geselecteerde traject. |
|  | 9. De actor bekijkt de uitgebreide informatie. |
| **Domeinregels** | 1. Beschikbare uitgebreide opties zijn: |
|  | - Historische reistijden, te selecteren voor een bepaalde periode |
|  | - Filter reistijden op basis van provider |
|  | 2. De reistijden worden gepresenteerd in het volgende formaat: [hh:mm:ss] |

## Use Case 3: Controleer data

|  |  |
| --- | --- |
| **Use Case naam** | Controleer data |
| **Use case nummer** | 3 |
| **Toepassing** | back-end |
| **Beschrijving** | Periodiek de data in de database controleren en corrigerende updates doorvoeren. |
| **Primaire Actoren** | systeem |
| **Preconditie** | nvt |
| **Postconditie** | De optimale reistijden zijn represenatief voor de huidige verkeerssituatie op macroniveau. |
| **Trigger** | Elke week op middernacht. |
| **Basisverloop** | 1. Het systeem verzamelt de reistijden van alle providers voor de trajecten. |
|  | 2. Het systeem herberekend de optimale reistijden voor de trajecten op basis van de reistijden. |
|  | 3. Het systeem corrigeert de optimale reistijden indien nodig. |
| **Alternatief verloop** | nvt |
| **Domeinregels** | 1. Voor de herberekening van de optimale reistijd moet met volgende factoren rekening gehouden worden: |
|  | \* Voorzien de providers al dan niet een optimale reistijd voor de trajecten. |
|  | \* Enkel de realtime reistijden op een rustiger tijdstip (21u tot 6u) zijn bruikbaar voor de analyse. |
|  | 2. De optimale reistijd wordt berekend aan de hand van de gemiddelde van de bruikbare data |

## Use Case 4: Bekijk actuele verkeerssituatie

|  |  |
| --- | --- |
| **Use Case naam** | Bekijk actuele verkeerssituatie |
| **Use case nummer** | 4 |
| **Toepassing** | front end |
| **Beschrijving** | De actor bekijkt de actuele verkeerssituatie |
| **Primaire Actoren** | Gebruiker, Operator |
| **Stakeholders** | / |
| **Precondition** | / |
| **Postcondition** | De actor beschikt over de actuele verkeerssituatie. |
| **Basisverloop** | 1. De actor wenst de actuele verkeersituatie te bekijken. |
|  | 2. Het systeem haalt de reistijden en gebeurtenissen van de routes op. |
|  | 3. Het systeem geeft de routes met bijhorende reistijden en eventuele gebeurtenissen weer. |
| **Alternatief verloop** | 2A. Het systeem kan geen routes met bijhorende data ophalen. |
|  | 2A1. Het systeem geeft een foutboodschap weer. |
|  | 2A2. Einde use case zonder bereiken postconditie. |
| **Domeinregels** | 1. Verkeersituatie wordt weergegeven op een kaart. |

## Use Case 5: Vergelijk verkeersinformatie

|  |  |
| --- | --- |
| **Use Case naam** | Vergelijk verkeersinformatie |
| **Use case nummer** | 5 |
| **Toepassing** | front end |
| **Beschrijving** | De actor vergelijkt actuele of historische verkeerssituaties |
| **Primaire Actoren** | Operator |
| **Stakeholders** | / |
| **Precondition** | / |
| **Postcondition** | De actor kan verschillende verkeerssituaties met elkaar vergelijken. |
| **Basisverloop** | 1. De actor wenst verkeerssituaties te vergelijken. |
|  | 2. Het systeem geeft de mogelijke vergelijkingen weer. |
|  | 3. De actor selecteert een type vergelijking. |
|  | 4. Het systeem valideert en haalt een overzicht op van de mogelijke verkeersituaties. |
|  | 5. De actor selecteert een aantal verkeerssituaties om te vergelijken. |
|  | 6. Het systeem valideert en haalt de detailgegevens op van de relevante verkeerssituaties |
|  | 7. Het systeem geeft de detailgegevens weer. |
| **Alternatief verloop** | 4A. Het systeem kan geen relevante verkeerssituaties ophalen. |
|  | 4A1. Het systeem geeft een foutboodschap weer. |
|  | 4A2. Ga naar 3 |
|  | 6A. Het systeem kan geen detailgegevens ophalen. |
|  | 6A1. Het systeem geeft een foutboodschap weer. |
|  | 6A2. Ga naar 5 |
| **Domeinregels** | **Mogelijke types vergelijking:** |
|  | 1. Gegevens van verschillende momenten(uren, dagen, speciale perioden) |
|  | 2. Gegevens van verschillende trajecten |
|  | 3. Verkeerssituatie met bezettingsgraad bepaalde parking |

## Use Case 6: Data ophalen en hergebruiken

|  |  |
| --- | --- |
| **Use Case naam** | Data ophalen en hergebruiken |
| **Use case nummer** | 6 |
| **Toepassing** | front end |
| **Beschrijving** | De actor maakt een API call om de verzamelde data te krijgen en te hergebruiken |
| **Primaire Actoren** | Gebruiker |
| **Stakeholders** | / |
| **Precondition** | / |
| **Postcondition** | De actor beschikt over de verzamelde verkeersdata. |
| **Basisverloop** | 1. De actor wenst zelf een nieuwe toepassing te maken adhv de verkeersdata. |
|  | 2. De actor maakt een API call naar de gepaste URL naar gelang de gewenste data. |
|  | 3. De actor krijgt de data terug in JSON formaat. |
| **Alternatief verloop** | 3A. Er is geen data beschikbaar die voldoet aan de eisen van de actor, dus wordt een correcte response teruggestuurd. |
|  | 3B. Er wordt een foutboodschap teruggestuurd. |
| **Domeinregels** | 1. Verkeersituatie wordt weergegeven op een kaart. |

Ontwerp

# Inleiding

Dit onderdeel van het dossier is voornamelijk gericht op de structuur van de applicaties zelf. Dit is nuttig voor programmeurs om zich snel in te kunnen werken of voor onderhoud op middellange termijn te kunnen vergemakkelijken. Het ecosysteem dat instaat voor het opvolgen van het verkeer in en rond Gent valt uit elkaar in enkele verschillende applicaties en ondersteunende componenten.

Allereerst is er de providerapplicatie. Deze applicatie staat in voor het periodiek opvragen van de verkeerssituatie en het wegschrijven van deze informatie naar een databank. De applicatie zelf staat in voor het wachten tot de volgende reeks gegeven in verband met het verkeer moeten worden opgehaald. Deze planning wordt dus niet vanuit het besturingsysteem opgelegd zoals met CRON bevoorbeeld.

Naast de providerapplicatie is er ook een webapplicatie. Deze kan gebruikt worden door mensen van het verkeerscentrum of inwoners van Gent om informatie op te halen in verband met de huidige verkeersituatie. Hiernaast zijn er voor mensen van het verkeercentrum nog meer mogelijkheden zoals het wijzigen van de op te volgen trajecten, geschiedenis van de verkeerssituatie vergelijken en meer.

Natuurlijk is er ook een databank aanwezig achter de schermen waarin alle informatie wordt bijgehouden.

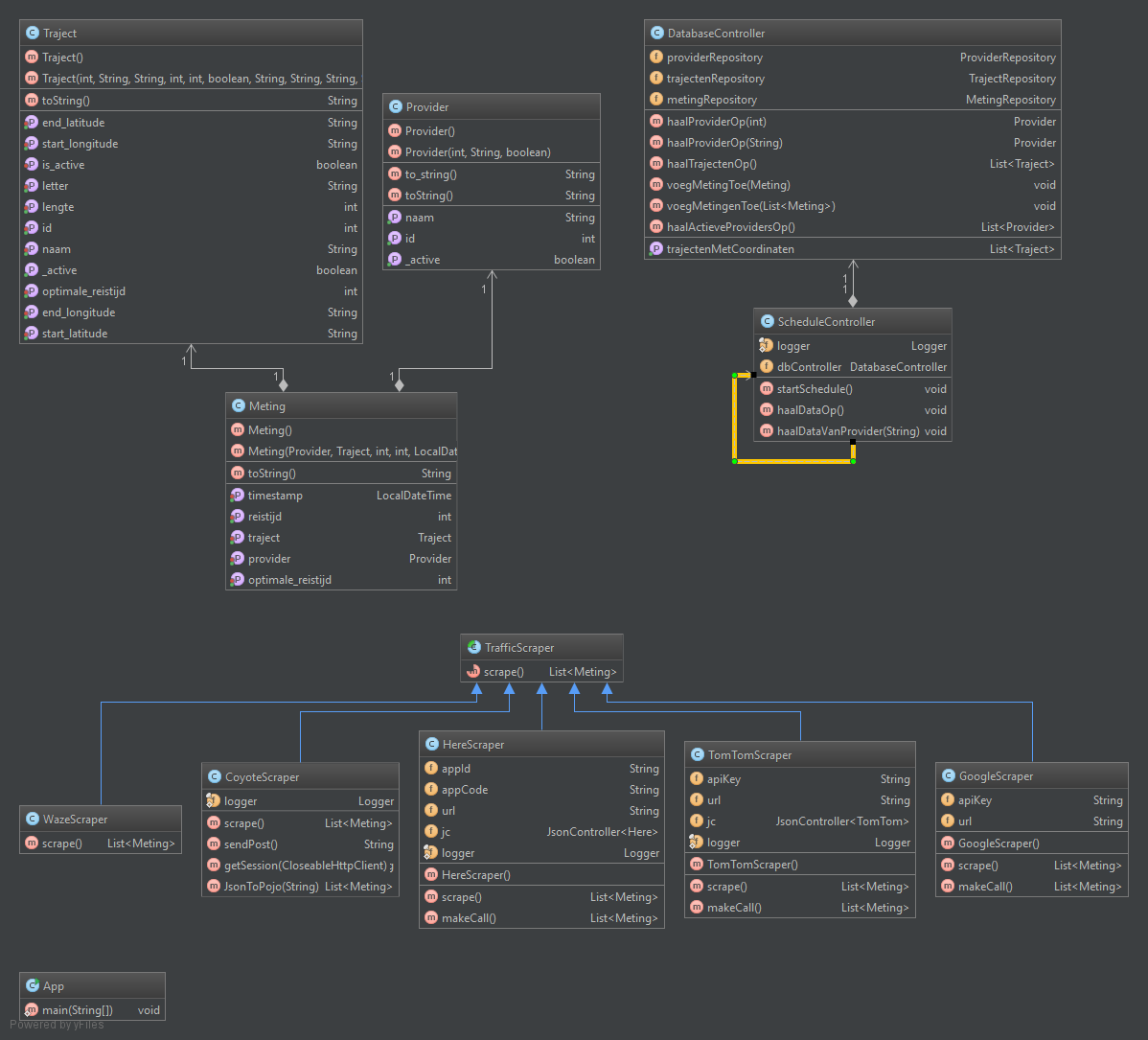
# Providerapplicatie

## State diagram



## Design Class Diagram (DCD)

Te fixen $$$$$ opsplitsen van het DCD



### Detailoverzicht scrapers

### Detailoverzicht domein

### Detailoverzicht repository/controller

### Legende

|  |  |
| --- | --- |
| Icon | Description |
| classTypeJavaClass | Class |
| classTypeAbstract | Abstract Java class |
| classTypeAnnot | Annotation |
| classTypeEnum | Enumeration |
| classTypeException | Exception |
| img | Final Java class |
| classTypeInterface | Interface |
| classTypeMain | Java class that contains declaration of the main() method. |
| classTypeTestCase | Test case |
| classTypeJavaOutOfSourceRoot | Java class located out of the source root. Refer to the section [Configuring Content Roots](https://www.jetbrains.com/help/idea/2016.1/configuring-content-roots.html) for details. |
| excludeFromCompilation.png | Java class [excluded from compilation](https://www.jetbrains.com/help/idea/2016.1/configuring-compiler-settings.html). |
| method | Method |
| method_abstract | Abstract method |
| field | Field |
| variable | Variable |
| property | Property |
| property | Property with getter |
| property | Property with setter |
| property_yellow | Parameter |
| xml_element | Element |
| folder | Directory |
| moduleFolder | Module |
| groupOfModules | Group of modules |
| iconPackage | Package |
| rootSource | Source root |
| rootTest | Test root |
| rootExcluded | Excluded root |
| rootResourceIJ | Resources |
| rootTestResourceIJ | Test resources |
| rootGeneratedSourceIJ | Generated source roots |
| rootGeneratedTestSourceIJ | Generated test source roots |
| Visibility modifiers | |
| locked | Read-only class, e.g. from a jar of an external library. |
| private.png | private |
| protected.png | protected |
| package_protected.png | package protected |
| staticMark | static |
| public.png | public |

## Algoritmes

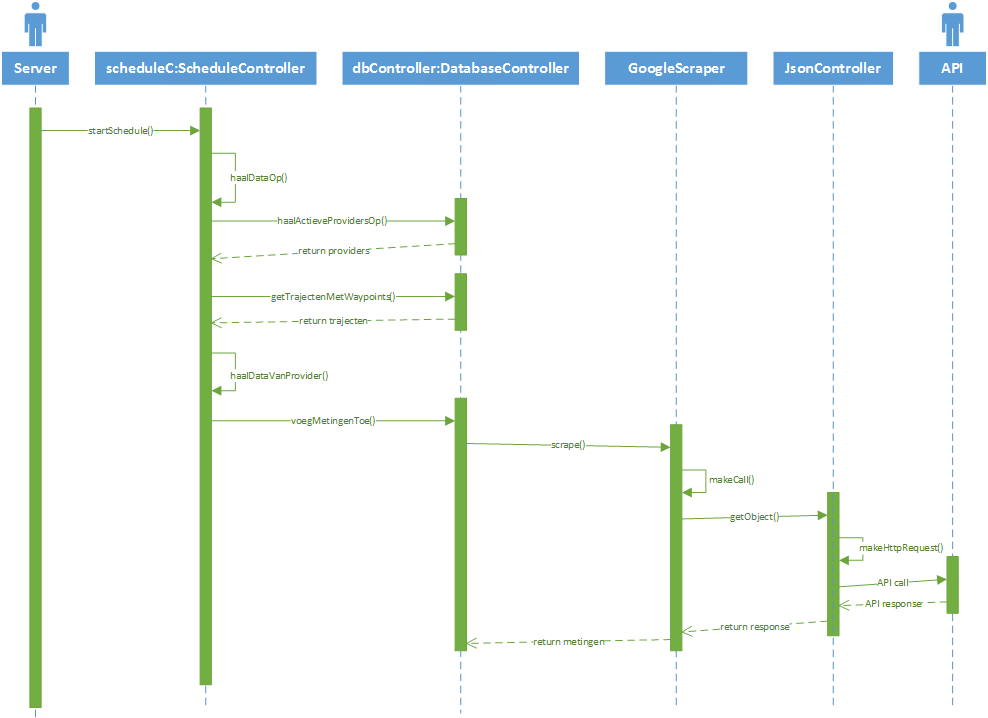
### API call maken

Een API call maken, en de database vullen met metingen van de verschillende routes, wordt gestart door de ScheduleController. Deze haalt eerst de benodigde data op om de API call te kunnen maken namelijk alle beschikbare providers naar waar een API call gemaakt kan worden, en alle trajecten waarvoor een API call gemaakt moet worden.

Eenmaal al de benodigde data verzameld is, kunnen metingen gemaakt worden en toegevoegd worden aan de database. Daarvoor wordt de DatabaseController en een Scraper opgeroepen. De Scraper maakt een call, waarbij een request wordt gestuurd naar de JsonController gestuurd wordt. De JsonController maakt op zijn beurt dan een API call naar de API en vangt zijn response op. Deze response wordt dan terug gestuurd naar de Scraper die de oproep deed.

De Scraper controleert dan op zijn beurt de response. Als de response correcte data bevat, dan wordt de meting toegevoegd in de database. Als de response een error of geen data bevat, dan wordt een lege meting toegevoegd in de database.

Voor elke provider wordt de DatabaseController elke 5 minuten opgeroepen en wordt de bovenstaande procedure doorlopen.



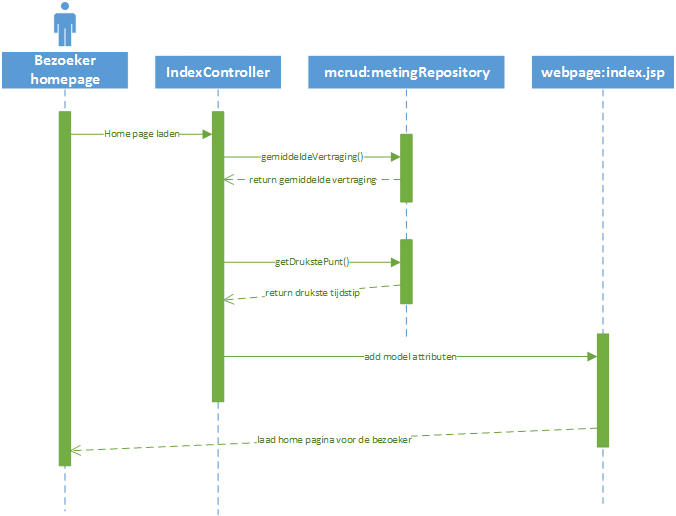
# Webapplicatie

## Algoritmes

### Home pagina bezoeken

Als een bezoeker de home pagina wil bezoeken, dan wordt deze geladen door de IndexController.

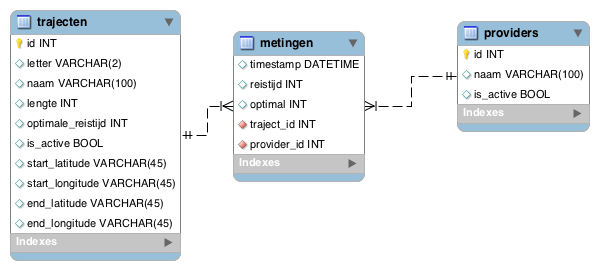
De IndexController vraagt eerst via de metingRepository de gemiddelde vertraging en het drukste traject met het drukste punt op, zodat deze als attributen kunnen meegegeven worden aan de index javaserver page. Deze index.jsp wordt dan met deze attributen geladen door de IndexController, zodat de homepagina geladen kan worden voor de bezoeker.



# Databank

## Entity-relationship diagram (ERD)

ERD diagram updaten $$$$$$



Testen

# Testplan

## Scope definitie

Tijdens de testfase worden meerdere onderdelen van de applicatie getest. De aandacht zal vooral gevestigd worden op de interne functionaliteit, bruikbaarheid en de interactie met gebruikers.

Om een testplan op te stellen is het belangrijk om de scope te definiëren. De scope kan opgesplitst worden in enkele categorieën: gebruiksvriendelijk, functionaliteit en abstractie (voor toekomstige ontwikkelaars).

Onder gebruiksvriendelijkheid zullen alle testen vallen die betrekking hebben met de userinterface van de applicatie. De functionaliteit zal getest worden aan de hand van unit en integration testen. Het testen van de abstractie is een proces dat continue uitgevoerd moet worden tijdens de ontwikkeling van de software. Het uittesten hiervan vereist dat de testers de broncode doornemen en om zoek gaan naar mogelijkheden tot verdere abstractie.

Wanneer een scope gedefinieerd wordt is het net zo belangrijk om te vermelden welke elementen geen onderdeel zijn van de scope. De scope wordt beperkt tot de applicatie zelf. Interactie met onderliggende lagen zoals het besturingssysteem, GlassFish en MariaDB worden niet getest. Er wordt uitgegaan van een werkende omgeving waarop het programma kan draaien. Tot slot wordt het stresstesten van de applicatie eveneens uit de scope gehouden.

## Unit testen

$ aan te vullen

## Integration testen

$ aan te vullen

## Usability testen

In dit onderdeel wordt de GUI (grafische user interface) getest. Het uittesten van de grafische interface gaat hand in hand met de use cases. Iedere use case beschrijft een handeling die de gebruiker zou moeten kunnen uitvoeren met het systeem. Indien de gebruiker dit niet kan (bijvoorbeeld door fouten) of heel wat hinder ondervindt om tot een resultaat te komen, dan kan de test als mislukt beschouwd worden. Het toekennen van veel van deze opdrachten aan externe personen is zeker een meerwaarde mede door hun beperkte of afwezige kennis van de toepassing.

$ aan te vullen: per usecase een test ontwerpen + eventuele extra testen

Use Case 1: Meld abnormaliteiten 7

Use Case 2: Bekijk reistijden 8

Use Case 3: Controleer data 9

use Case 4: Bekijk actuele verkeerssituatie 10

use Case 5: Vergelijk verkeersinformatie 11

use Case 6: Data ophalen en hergebruiken 12

## Toewijzing testen

$tabel aanvullen

|  |  |
| --- | --- |
| Naam van de test | Naam van de tester |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Installatie en onderhoud

## Inleiding

Dit onderdeel van de software-analyse zal gaan over de installatie. Hieronder wordt in detail besproken hoe vanaf de broncode een werkend programma gemaakt kan worden.

## Installatie

Om het project te laten runnen zijn er wat praktische componenten nodig. Er zijn minimaal twee toestellen nodig om dit project te laten werken. Allereerst is er een server nodig. Dit kan zowel een fysieke hardware server, een desktop als een virtuele machine zijn. Om deze server zal Linux draaien. Een tweede component is een Windows toestel. Deze zal eenmalig gebruikt worden om de broncode te compileren. Eenmaal dit gebeurt is kan het gecompileerde project overgeplaatst worden naar de server.

### Server

Vooraleer de installatie van start kan gaan moet een keuze gemaakt worden welk besturingssysteem gebruikt zal worden. De keuze die gemaakt wordt hangt af van persoonlijk voorkeur. Het is aan te raden om een distributie te kiezen die eerder gericht is naar servers en niet naar desktops. In deze installatiehandleiding zullen twee distributies in detail besproken worden: enerzijds Fedora en anderzijds CentOS. De reden hiervoor is dat een Fedoraserver aangeboden werd door de opleidingscoördinatoren van onze richting. Fedora is zeker geschikt als server maar deze heeft echter het nadeel van een snelle Life-Cycle. Dit wil zeggen dat een versie van Fedora sneller beschouwd wordt als ‘oud’ en hierdoor geen updates of beveiligingspatches meer ontvangt. In het algemeen wordt een bepaalde versie ondersteund tot één maand nadat twee nieuwe versies uitgekomen zijn. Er wordt ongeveer ieder half jaar een nieuwe versie gereleased. CentOS daarentegen heeft een veel langere ‘life cycle’. De huidige versie (CentOS 7) werd gereleased in juni 2015 en zal onderhouden worden tot juni 2024.

Stabiliteit is eveneens een belangrijk aspect van dit project. Het is de bedoeling dat deze server (en het project) 24 uur per dag, 7 dagen per week draait. Hoe stabieler het besturingssysteem, hoe groter de kans dat dit zich ook daadwerkelijk in de praktijk voordoet. Zowel CentOS als Fedora staan bekend voor hun stabiliteit, dit is voor heel wat Linux distributies het geval. Door de vele releases van Fedora lijdt de stabiliteit eronder. Als een CentOS een label van 100% stabiel krijgt, dan komt dit overeen met 99.5% stabiliteit bij Fedora. Uiteindelijk zijn beide server geschikt als server en ligt de keuze bij de eindgebruiker.

#### Fedora

De installatie van Fedora is relatief eenvoudig. Net zoals bij Windows kan de installatie gebeuren op een grafische manier. Het besturingssysteem kan gedownload worden via de website getfedora.com. Hier is het belangrijk om de serverversie te selecteren. De keuze voor 32-bit of een 64-bit systeem is afhankelijk van de fysieke server. De algemene regel is om een 32-bit installatie uit te voeren op een systeem met 4GB ram of minder. Indien de CPU van het toestel 64-bit ondersteunt, wordt best gekozen voor een 64-bit installatie.

Wanneer het toestel opgestart wordt met de installatie-cd van Fedora wordt gevraagd aan de gebruiker of deze de software wil installeren of we testen. Hier wordt geklikt op ‘install’.



De eerste stap is een taal selecteren. Indien er internettoegang is, zal Fedora automatisch Nederlands selecteren. Klik op verdergaan.



Vervolgens wordt een overzicht gegeven van de te installeren onderdelen. Hier is het belangrijk om onder softwareselectie en installatiebestemming wijzigingen aan te brengen.



Onder softwareselectie moet het vinkje gemarkeerd worden naast MariaDB (MySQL) database. Deze database is nodig om de gegevens op te slaan die het project genereert. Indien geopteerd wordt voor een ander type database (zoals MySQL), zal deze op eigen houtje geïnstalleerd moeten worden. Vervolgens klikt men linksboven op klaar.



Nu men terug op het overzichtsscherm zit, moet de installatiebestemming geselecteerd worden. Dit wordt bekomen door op het hardeschijficoontje te drukken en vervolgens de harde schijf te selecteren. In onderstaande screenshot staan een voorbeeld van dit menu met de selectie van de harde schijf. Vervolgens klikt men op klaar.



In het overzichtsmenu wordt nu onderaan geklikt op begin met installatie om de installatie te starten. Tijdens de installatie wordt de mogelijkheid geboden om het wachtwoord van de root in te stellen. Een nieuwe gebruiker aanmaken is niet meteen nodig.



Wanneer het installatieproces voltooid is, is het belangrijk om de cd te verwijderen vooraleer opnieuw op te starten. Klik vervolgens op herstarten.



#### CentOS

Het installeren van CentOS is gelijklopend met de installatie van Fedora, dit omdat beide besturingssystemen ontwikkeld zijn door dezelfde ontwikkelaar namelijk Red Hat. Op het moment van schrijven is de huidige versie CentOS 7. Deze is te downloaden via centos.org. Hier moet de keuze gemaakt worden tussen de minimal ISO, de dvd ISO en de full ISO. De dvd ISO is het meest geschikt voor dit project maar indien de beheerder van dit toestel verkiest om zelf de packages te installeren die vereist zijn, is de minimal ISO een oplossing. De controle indien het toestel 32- of 64-bit is, is niet aan de orde om er tijdens de installatie een controle uitgevoerd wordt om welk type toestel het gaat. Afhankelijk van de processor wordt de juiste versie geïnstalleerd.

Wanneer de software opgestart wordt gevraagd indien de gebruiker de software wil installeren of in demo wil runnen. Installatie is de optie die gekozen moet worden.



Het vervolg van de installatie gebeurt op een grafische manier. Eerst moet een taal geselecteerd worden waarin het installatieproces zal verlopen. Na de selectie van de taal klikt met rechtsonder op doorgaan.



Het volgende scherm biedt een overzicht aan van alle mogelijke installatieopties. Onder het submenu van lokalisatie staan heel wat algemene instellingen omtrent toetsenbord, taal, datum & tijd. Deze instellingen worden vaak via het netwerk al reeds ingesteld. De kans is groot dat de juiste opties al geselecteerd zijn.

Onder het submenu software moet wel wat gewijzigd worden. Klik hier op software selectie.



Standaard wordt een minimale installatie voorgesteld, deze kan gewijzigd worden naar een infrastructure server met de add-ons optie MariaDB Database Server in de rechter kolom. MariaDB is nodig voor het opzetten van een database om de gegevens te bewaren die het project genereert. Indien er gebruik gemaakt wordt van een externe database (bijvoorbeeld op een ander toestel) of een ander type database, moet deze optie niet geselecteerd worden. Vervolgens klikt met linksboven op klaar.



Terug bij het overzichtsmenu gekomen staat een waarschuwing bij bestemming van de installatie onder het submenu systeem. Wanneer met hier op klikt hoeft enkel de harde schijf geselecteerd te worden. Voor het aanmaken en configureren van partities, opstartprocedures,… zal de installer zelf zorgen. Vervolgens klikt met op klaar.



Terug bij het overzicht aangekomen wordt onderaan geklikt op begin installatie. Tijdens de installatie is het handig om het root wachtwoord in te stellen. Dit is het wachtwoord voor de administrator zoals deze in Windows genoemd wordt. Een extra gebruiker aanmaken is momenteel niet nodig.



Wanneer de installatie ten einde is, is het belangrijk om eerst de cd uit te werpen vooraleer te klikken op herstarten, dit zal voorkomen dat de server opstart vanaf de cd en die de demo van CentOS zal laden.



#### GlassFish

Het eerste stuk software dat geïnstalleerd moet worden is GlassFish. GlassFish is nodig om het programma te draaien. Het zorgt ervoor dat applicaties platformonafhankelijk kunnen werken.

Vooraleer GlassFish geïnstalleerd kan worden, is het noodzakelijk om de Java EE SDK te installeren.

Java dien als onderliggende laag en is eveneens de programmeertaal waarin de applicatie geschreven is.

Onderstaande commando’s voeren volgende zaken uit:

* programma’s wget en unzip installeren
* gebruiker toevoegen met username glassfish
* het wachtwoord van de glassfish user wijzigen
* java jdk 8.45 downloaden
* java jdk 8.45 installeren

**yum install wget unzip nano**

**adduser --comment ‘Glassfish User’ --home-dir /home/glassfish glassfish**

**passwd glassfish**

**wget --no-cookies --no-check-certificate**

**--header "Cookie: oraclelicense=accept-securebackup-cookie"**

**"http://download.oracle.com/otn-pub/java/jdk/8u45-b14/jdk-8u45-linux-x64.rpm" -O jdk-8u45-linux-x64.rpm**

**yum install jdk-8u45-linux-x64.rpm**

Nu de Java JDK geïnstalleerd is, kan GlassFish toegevoegd worden aan het systeem. Onderstaande commando’s voeren de volgende zaken uit:

* inloggen met de user glassfish
* glassfish 4.1 downloaden en uitpakken
* het gedownloade bestand wissen

**su glassfish**

**wget --no-cookies --no-check-certificate**

**--header "Cookie: oraclelicense=accept-securebackup-cookie"**

**"http://download.oracle.com/glassfish/4.1.1/release/glassfish-4.1.1.zip" -O glassfish-4.1.1.zip**

**unzip glassfish-4.1.1.zip**

**rm –f glassfish-4.1.1.zip**

**exit**

Nu GlassFish aanwezig is op de server resteert nog één stap: het toevoegen van een service om GlassFish te starten. Het aanmaken van deze service komt neer op het aanmaken van een bestand met configuratieparameters. Onderstaand commando maakt dit bestand aan en opent het.

**nano /etc/systemd/system/glassfish.service**

Nu het bestand geopend is, moet onderstaande configuratie toegevoegd worden.

**[Unit]**

**Description = GlassFish Server v4.1.1**

**After = syslog.target network.target**

**[Service]**

**User=glassfish**

**ExecStart = /usr/bin/java -jar /home/glassfish/glassfish4/glassfish/lib/client/appserver-cli.jar start-domain**

**ExecStop = /usr/bin/java -jar /home/glassfish/glassfish4/glassfish/lib/client/appserver-cli.jar stop-domain**

**ExecReload = /usr/bin/java -jar /home/glassfish/glassfish4/glassfish/lib/client/appserver-cli.jar restart-domain**

**Type = forking**

**[Install]**

**WantedBy = multi-user.target**

Vervolgens drukt met de toetsencombinatie CTRL+O in om het bestand op te slaan.

Het is zeer handig om de glassfish service automatisch te laten starten bij het opstarten van het besturingssysteem. Het starten van de service en deze activeren wanneer het systeem opstart kan met onderstaande commando’s.

**systemctl enable glassfish.service**

**systemctl start glassfish.service**

Tot slot is het zeker handig om deze service te kunnen bereiken vanaf een extern toestel. Daarom is het noodzakelijk om bepaalde poorten te openen. Standaard worden volgende poorten door GlassFish gebruikt:

* Administratie: 4848
* HTTP: 8080
* HTTPS: 8181

Poorten openen kan met onderstaande commando’s:

**firewall-cmd --zone=pubic --add-port=4848/tcp --permanent**

**firewall-cmd --zone=pubic --add-port=8080/tcp --permanent**

**firewall-cmd --zone=pubic --add-port=8181/tcp --permanent**

**systemctl restart firewalld**

#### MariaDB

Indien tijdens de initiële installatie van het besturingssysteem van de server MariaDB niet opgenomen werd, wordt in dit onderdeel uit de doeken gedaan hoe dit geïnstalleerd kan worden. Het opzetten van een database is essentieel voor dit project. Zoals eerder aangehaald is het perfect mogelijk om een alternatieve database te gebruiken zoals MySQL of MSSQL. De installatie kan zowel gebeuren op het lokale toestel als op een extern. Sowieso moet een externe server gebruikt worden bij het gebruik van MSSQL omdat dit enkel bestaat voor Windows server. Onderstaande commando’s installeren MariaDB en voeren vervolgens een beveiligde installatie uit.

**yum install mariadb-server**

**myqsl\_secure\_installation**

Vervolgens wordt de database aangemaakt, samen met een MariaDB user waarmee de applicatie zal connecteren. Tot slot wordt het databaseschema geïnstalleerd.

**mysql -u root -p**

**create database vop;**

**grant all privileges on vop.\* to vop@localhost identified by ‘secretpassword’;**

**flush privileges;**

**exit**

**mysql -u root -p password <./DatabankCreate.sql**

Tot slot moet de MariaDB service gestart worden en wordt ingesteld dat deze opstart samen met het systeem.

**systemctl enable mariadb**

**systemctl start mariadb**

#### Vast IP-adres

Het toekennen van een vast IP-adres is belangrijk zodat het steeds mogelijk is om de gebruikersinterface te kunnen bereiken via een hetzelfde IP-adres. Indien er gewerkt wordt met een DHCP server die at random een IP-adres toekent aan de server, dan is de kans groot dat bij het herstarten van dit toestel, er een ander IP-adres toegekend wordt. Voor dit probleem zijn twee mogelijke oplossingen: ofwel stelt men in op de DHCP server dat het toestel met dit specifiek MAC-adres gekoppeld wordt aan een IP-adres, ofwel stelt men lokaal in op de server dat hij zichzelf een vast IP-adres toekent. Beide oplossingen worden hieronder beschreven.

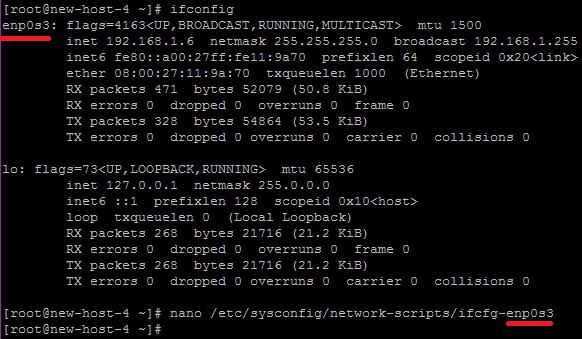
##### Statisch (lokaal)

Linux kent niet zoals Windows een naam toe aan iedere netwerkadapter. Om wijzigingen te kunnen aanbrengen moet deze naam gekend zijn. Standaard wordt aan fysieke toestellen de naam eth0 toegekend. Om de naam te achterhalen moet het eerste commando uitgevoerd worden. Daarna kan het configuratiebestand aangepast worden van die netwerkadapter. Daarbij is het belangrijk om de naam in te vullen die teruggegeven wordt bij het eerste commando.

**ifconfig**

**nano /etc/sysconfig/network-scripts/ifcfg-eth0**

Hieronder staat een voorbeeld van beide commando’s:



Nu het configuratiebestand geopend is, moeten onderstaande regels aangepast of toegevoegd worden:

**BOOTPROTO=none**

**ONBOOT=yes**

**NETWORK=192.168.1.1**

**NETMASK=255.255.255.0**

**IPADDR=192.168.1.10**

**USERCTL=no**

##### Dynamisch

Het instellen van een DHCP-server kan hier niet in detail beschreven worden omdat er oneindig veel types, versies en varianten bestaan. Zowel Linux als Windows kunnen een DHCP-server draaien, routers kunnen instaan als DHCP-server, … Omdat het uitschrijven van de configuratie voor iedere variant onmogelijk is, worden enkel algemene instructies gegeven die overal van toepassing zijn.

Om een toestel te kunnen linken aan een IP-adres in de DHCP-server, heeft deze server het MAC-adres nodig van dit toestel. Het MAC-adres kan gevonden worden via onderstaand commando:

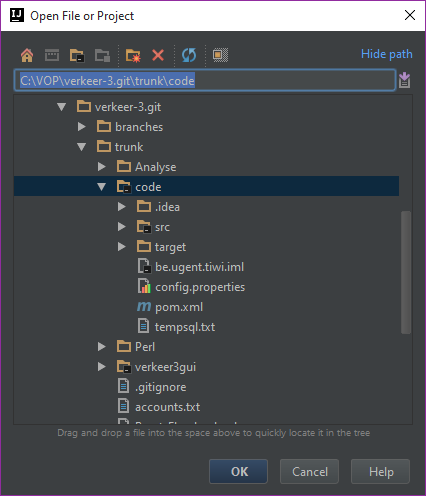
**ifconfig -a**

Bij de netwerkadapter (meestal eth0 genoemd) staat een regel met als naam HWaddr, hiernaast staat het MAC-adres vermeld. Bijvoorbeeld: 00:08:C7:1B:8C:02. Dit moet vervolgens in de DHCP-server toegevoegd worden, samen met het gekoppelde IP-adres.

### Desktop

Op de desktop dient eender welke versie van Windows geïnstalleerd te zijn. De desktop is nodig om de broncode van het project om te zetten in een uitvoerbaar programma, wat op zijn beurt geplaatst kan worden op de server. GlassFish zal dan dienst doen als service die die applicatie uitvoert. Tijdens de productie is gebruik gemaakt van IntelliJ IDEA. Dit is een programmeeromgeving, gelijkaardig met Netbeans en Eclipse. Het is dan ook perfect mogelijk om vertrekkend van de broncode, het volledig project te compileren in de genoemde programmeeromgevingen. IntelliJ kan gedownload worden via de officiële site van de ontwikkelaars: <http://www.jetbrains.com/idea/>.

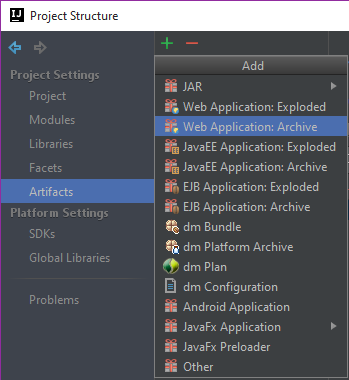
Na de installatie dient het project geïmporteerd te worden. Dit kan via de knoppen file 🡪 open. Vervolgens moet genavigeerd worden naar de projectmap en wordt de map code geselecteerd. Vervolgens klikt men op ok. IntelliJ zal nu de broncode en instellingen laden, dit kan even duren.



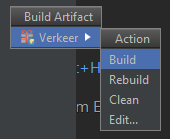
### Deployment

Probleem: enkel mogelijk in de enterprise versie, niet de community edition$$$$$$$$$$$$

Om de applicatie te runnen op de server vereist GlassFish dat er een WAR-file (Java Web Application) aangemaakt wordt. Dit kan bereikt worden via file 🡪 project structure 🡪 artifacts. Vervolgens klikt met op het groene + icoontje en wordt Web Application: Archive geselecteerd. Klik vervolgens op ok.

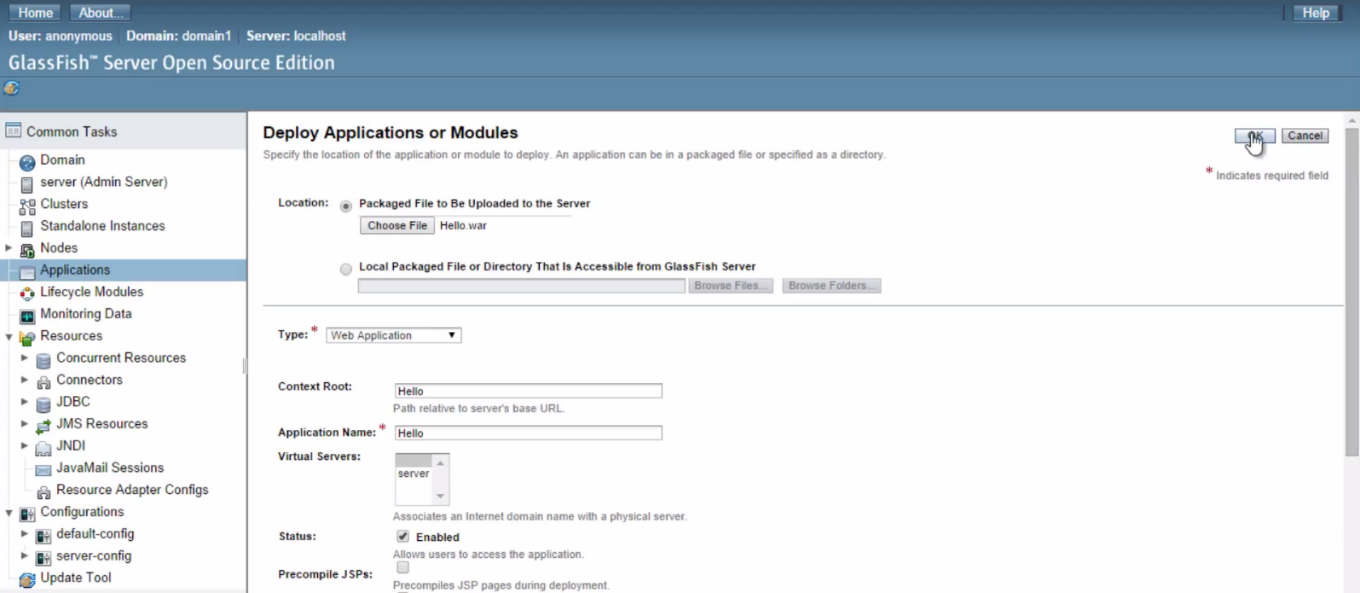


Nu IntelliJ weet wat er in de WAR file geplaatst moet worden, kan de opdracht gegeven worden tot het aanmaken van dit bestand. Dit wordt gedaan via build 🡪 build artifacts 🡪 action 🡪 build



Het WAR bestand is nu aangemaakt onder \code\classes\artifacts\Verkeer

De volgende stap is om dit bestand te deployen op de server. Dit kan bekomen worden door te surfen naar het administratorpaneel van Glassfish. Deze webpagina is actief op poort 4848 van de server bv: 192.168.1.10:4848. Wanneer men inlogt op de webpagina is er aan de linker kant een overzicht met allerlei opties. Om te deployen, druk op applications en vervolgens op choose file. Selecteer nu het WAR bestand dat eerder aangemaakt werd en klik op ok.



Retrospective

# Aandachtspunten sprints

## Sprint 2

* Structuur van domein(en eventueel databank). Elke meting heeft provider => één provider met lijst van metingen is conceptueel logischer.
* Optimalisatie netwerkverkeer: elke meting wordt individueel opgehaald, verwerkt en weggeschreven naar de databank => onnodige overhead, indien mogelijk data in één keer binnenhalen en in groep inserten in databank.
* De gegevens worden momenteel direct gebruikt in de querries. Dit vormt een beveiligingsrisico voor SQL-injection. =>Code herwerken zodat deze gebruik maakt van Prepared Statements