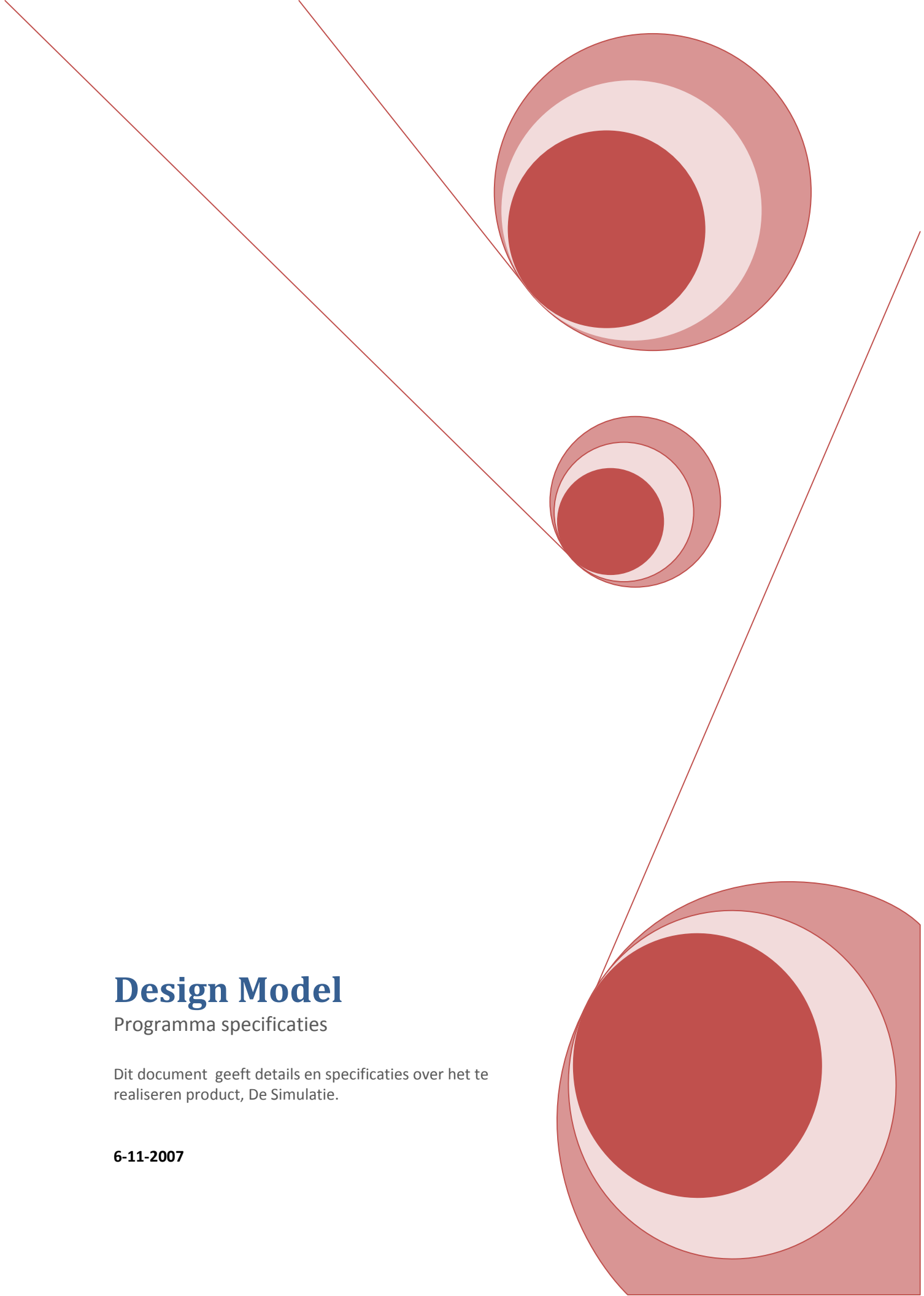


Design Model

Programma specificaties

Dit document geeft details en specificaties over het te realiseren product, De Simulatie.

6-11-2007

Inhoudsopgave

Inleiding

Een korte inleiding in het design model

Dit document is het vervolg op de opgeleverde analyse modellen. De verschillende onderdelen worden verder uitgediept. Denk hierbij aan o.a. het klassendiagram, het sequentiediagram en technische gegevens. Deze technische gegevens zijn bedoeld ter ondersteuning van de implementatie.

Management samenvatting

Een samenvatting van het gehele document

Om gebruik te kunnen maken van de “RailCab” simulator is één computer vereist. Doordat er gekozen is voor een one-tier systeem zal alles lokaal bewerkt en tijdelijk opgeslagen worden. Het is dan ook niet nodig om een internet of netwerk verbinding te hebben. Voor het realiseren van de simulatie is er gekozen om het systeem te ontwikkelen op basis van de programmeertaal JAVA. Om latere aanpassingen eenvoudig te kunnen implementeren is er gekozen om drie lagen te gebruiken. Zo is er een data laag, business laag en een user interface laag gerealiseerd die onderling met elkaar communiceren.

In het hoofdstuk klassendiagram vind u een uitgebreide specificatie van de klassen. Deze klassen zijn als geheel systeem weergeven evenals een gelaagde uitwerking. Hierbij vind u ook de klassendetails ter ondersteuning aan de diagrammen.

In het hoofdstuk sequentiediagrammen vind u een uitgebreide specificatie van het gedrag van het systeem. Dit gedrag word uitgebeeld in enkele situaties die zich voor zullen doen binnen de simulatie.

Systeem ontwerp

Hierin word beschreven welke opbouw er gebruikt gaat worden in het railcab systeem.

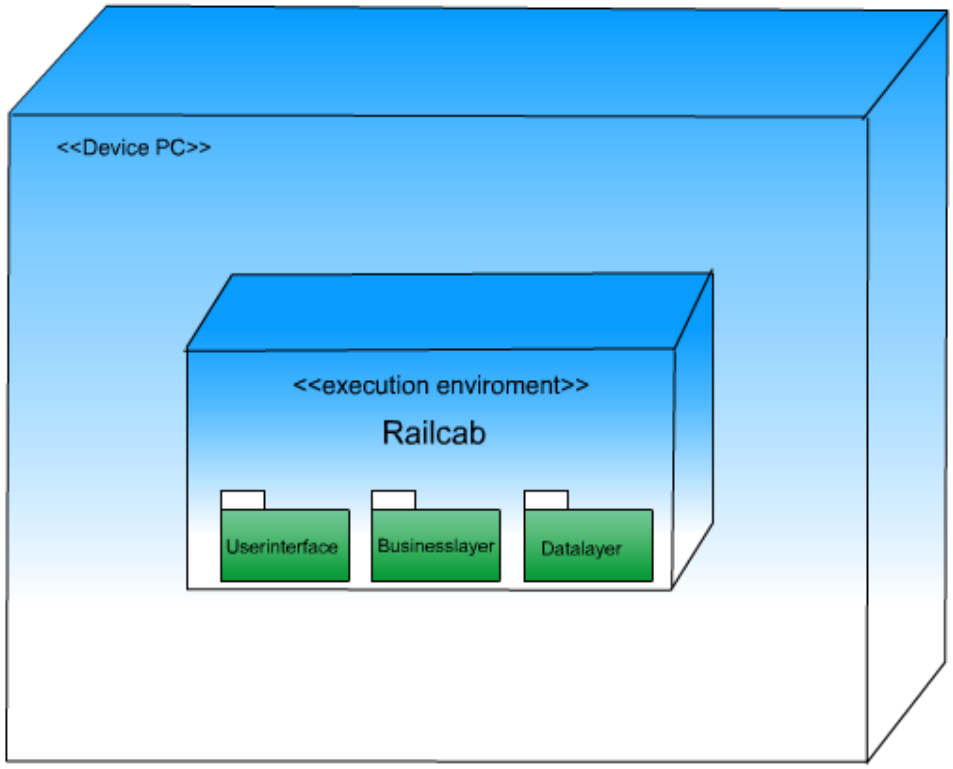
Aantal tiers

Voor dit systeem wordt gebruik gemaakt van een one-tier architectuur. Het voordeel van een one-tier systeem is de eenvoud. Het is bij dit project niet nodig om verschillende systemen te laten communiceren omdat het om een simulatie gaat en niet om de implementatie in de realiteit.

Aantal machines

Een computer die de simulatie draait heeft geen andere computer nodig om de simulatie te draaien. Er kunnen dus geen connectie problemen optreden bij het overdragen van gegevens naar andere computers omdat dit niet nodig is.

Vereenvoudigd implementatie diagram



Gekozen technologiën

In dit hoofdstuk zal worden uitgelegd welke technologieën er worden gebruikt en welke juist niet.

Voor het implementeren van de simulatie van het railcab systeem is gekozen voor de programmeertaal Java. Java is gekozen om twee redenen.

Ten eerste is het platform onafhankelijk en dus op iedere pc te gebruiken.
Ten tweede omdat de expertise van Itopia bij Java ligt. Om projectrisico's te voorkomen zal de keuze bij Java blijven.

Om de data op te slaan is gekozen voor interne opslag. Een database is overwogen, maar niet gekozen om ook twee redenen.

Ten eerste zal een lokale database het idee van platform "onafhankelijkheid" tegen gaan. Voor een lokale database zullen programma's op de betreffende pc geïnstalleerd moeten worden. Dit moet worden voorkomen.

Ten tweede zal een externe database de eenvoudigheid van het programma tegen gaan. Om gebruik te maken van een externe database is een internet of netwerk verbinding nodig op de pc die de applicatie uitvoert.

Systeem lagen

Voor de Railcab simulatie worden drie systeem lagen gebruikt. Deze drie lagen staan hier onder in het onderstaande tabel.

Laag	Beschrijving
User interface	Railcab Simulatie
Business	Handelt alle reserveringen af en zorgt dat de shuttles op tijd rijden.
Data	Bewaart alle benodigde data, zoals reserveringen en statistieken

Dit is een 1-tier lagen systeem. Alles gebeurt binnen het programma. Er zullen dus geen externe transacties plaats vinden. De drie lagen zijn logisch opgedeeld. De User Interface laag beeld het algoritme uit. Dit algoritme is geheel uitgewerkt in de Systeem laag die nauw samenwerkt met de Data laag waarin de reserveringen op worden geslagen. De User Interface werkt op zijn beurt weer samen met de Data laag om statistieken te kunnen tonen die in deze laag zijn opgeslagen.

Processen en threads

Hier wordt beschreven hoe de processen en threads afhankelijk van elkaar zijn.

Er zullen drie threads gevormd worden. Één voor het RailCab systeem, één voor het ZelfReizen systeem en één de Simulatie. Het RailCab systeem kan los van de andere draaien. De simulatie heeft het RailCab systeem nodig en het ZelfReizen heeft beide, het RailCab systeem en de simulatie, nodig om te functioneren.

Packages

Er zullen drie packages komen. Een met de logische inhoud van het RailCab systeem en de statistieken. En een package met de GUI die de simulatie bevat en het ZelfReizen systeem.

User interface	Business	Data
-StatisticUI -ReservationUI -StressTestUI -OverviewUI	-StressTest -Statistics -Shuttle -ShuntArea -Central -Perron -Station -Terminal -StationTerminal -PerronTerminal -Passenger	-Database -Reservation -SimDate

Systeem overeenstemming

In deze paragraaf zullen er verschillende systeemovereenstemmingen aan bod komen.
Het kan mogelijk zijn dat meerdere processen gebruik maken van een data bron en dan wel dezelfde data. Wanneer dat het geval is kan het voorkomen dat er data gemanipuleerd wordt bij het ene proces en het andere proces wat denkt dat de data juist is, ontvangt daarmee de gemanipuleerde onjuiste data. Om te voorkomen dat processen dezelfde data benaderen en manipuleren, zijn deze overeenstemmingen opgesteld.

Aangezien wij een simulatie maken wat bestaat uit een proces namelijk de simulatie,zijn er geen systeem overeenstemmingen opgesteld voor ons systeem. Wanneer er gebruik gemaakt zal worden van een n-tier systeem dat dezelfde data kan benaderen, dienen er echter de volgende overeenstemmingen plaats te vinden;

1. **Een persoon kan nooit een perron toegezegd krijgen door het invoeren van een code als deze op exact het zelfde moment worden ingevoerd door een tweede persoon.**

   1.1

   Voordat er een station wordt toegewezen, wordt er nogmaals gekeken of de code al eerder is ingevoerd. Zo kan het dus niet voorkomen dat de code 2 maal gebruik wordt.

Beveiliging

In deze paragraaf komen verschillende aspecten wat betreft beveiliging naar voren. Deze aspecten worden besproken en tevens worden er verschillende oplossingen aangedragen.

Aangezien wij een simulatie maken die bestaat uit één proces zijn er geen beveiligingsmaatregelen opgesteld voor ons systeem. Mocht er een daadwerkelijke implementatie van het systeem plaatsvinden, dient er met de volgende aspecten rekening gehouden te worden

• Privacy
De gebruikers van het systeem moeten niet de mogelijkheid hebben, informatie zoals statistieken te bemachtigen. Deze informatie moet dus verborgen blijven.

Oplossingen:
Om privacy te garanderen wordt de Database van de buitenwereld beschermd.
Dit wordt gedaan door firewalls en uitgebreide Autorisatie(zie ontkenning).
Tevens worden belangrijke gegevens, mochten deze zich voortdoen op het systeem, over een beveiligde verbinding (encryptie) worden verstuurd.

• Ontkenning:
De bron kan ten alle tijden niet ontkennen dat hij / zij de bron is.

Oplossingen:
Dit wordt opgelost zoals besproken in het kopje Autorisatie, door het ip adres van de bron op te slaan in samenstelling met het gebruikersnaam/wachtwoord systeem wordt dit gegarandeerd.

• Data integriteit:
De data moet ten allen tijden juist zijn en niet gemanipuleerd zijn.

Oplossingen:
Aangezien wij in dit systeem over het algemeen niet met gevoelige informatie in aanraking komen, wordt er geen gebruik gemaakt van oplossingen die de data integriteit kunnen waarborgen.

• Autorisatie
Het moet ten alle tijden mogelijk zijn de bron te achterhalen.

Oplossingen:
Zoals besproken bij privacy wordt er gebruik gemaakt van autorisatie indien gevoelige informatie wordt benaderd. Deze autorisatie geschiedt door middel van een gebruikersnaam en een wachtwoord. Tijdens de autorisatie wordt het ip adres van de bron opgeslagen zodat de bron achterhaald kan worden.

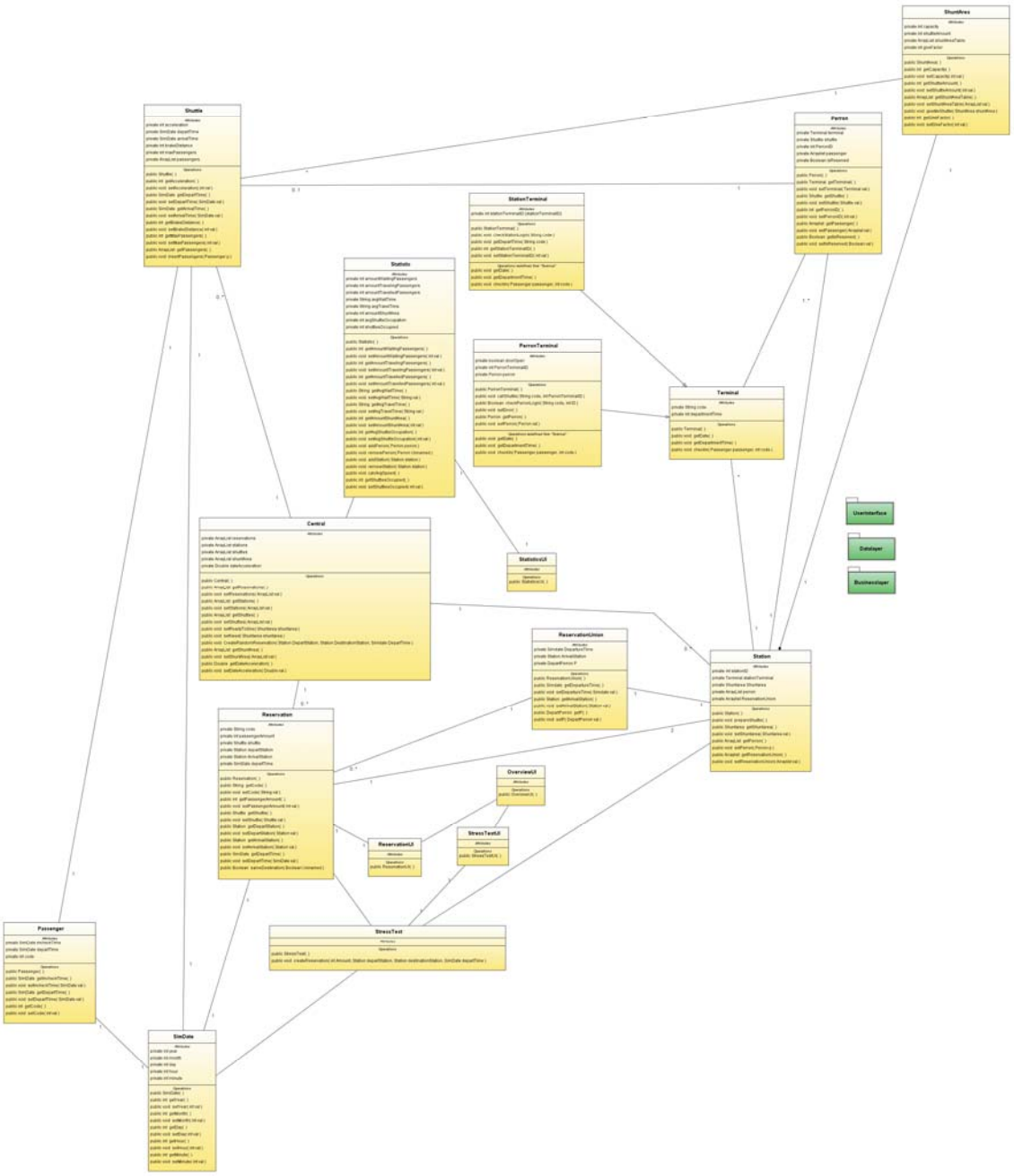
Sub-systeem ontwerp

In dit hoofdstuk word dieper ingegaan op de layers en de klassen.

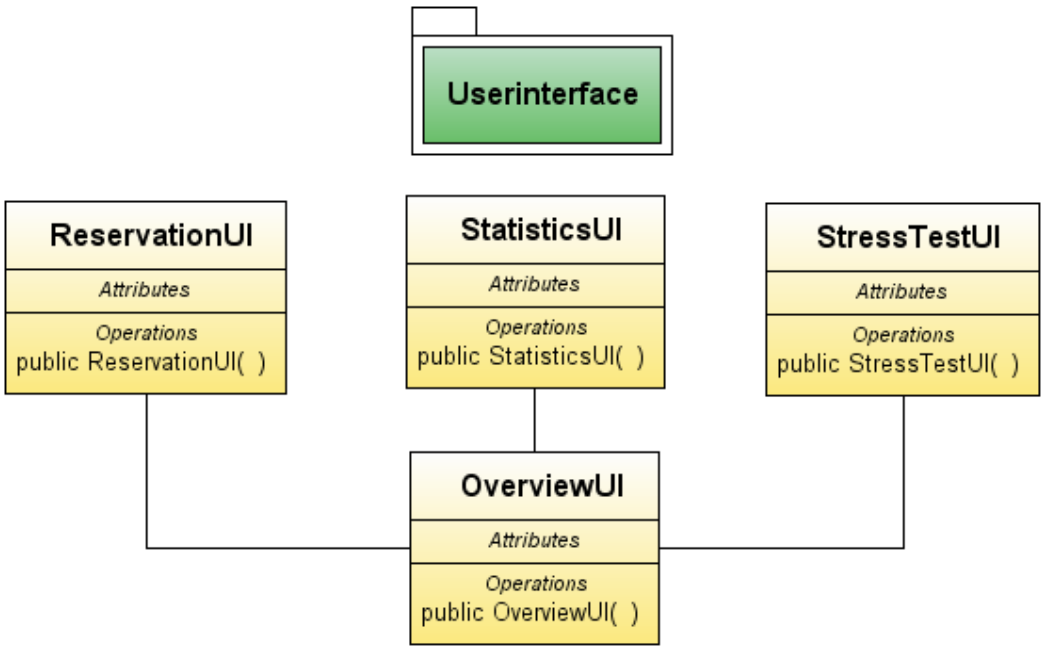
Klassendiagrammen

I*n de klassendiagrammen kan je de relaties, attributen en methodes tussen de verschillende klassen afleiden.*

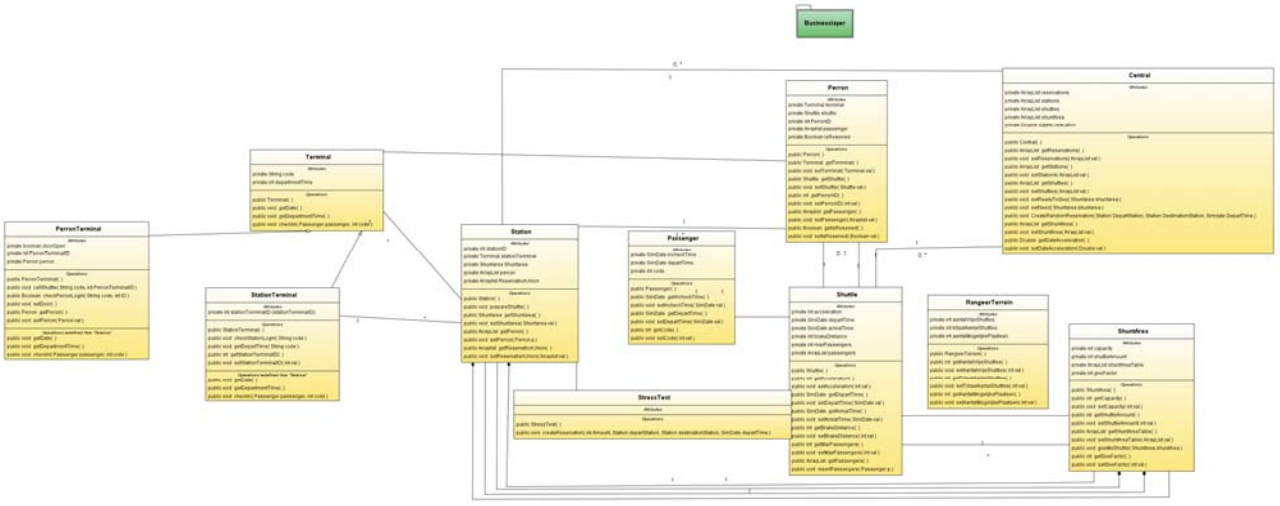
Niet-gelaagde klassendiagram



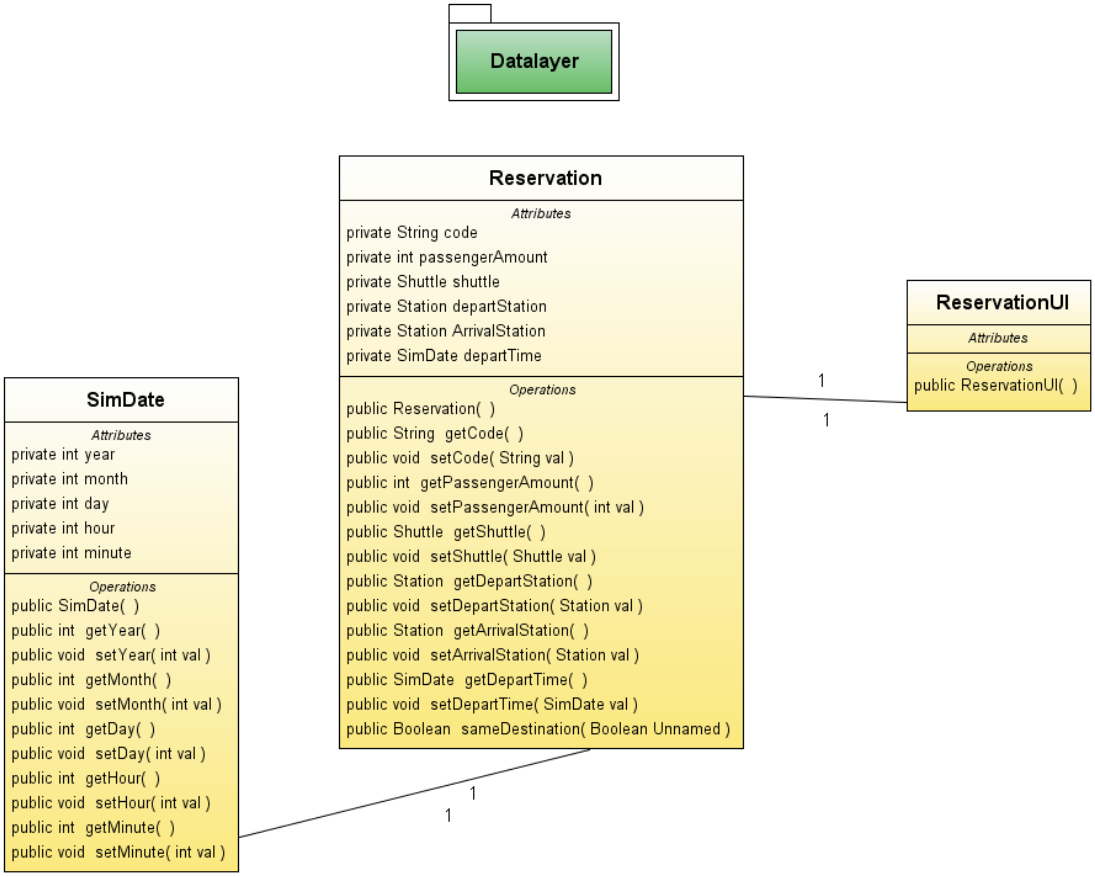
Userinterface layer



Business layer



Data layer



Klassen details

Hierin word uitgelegd wat de verantwoordelijkheden van de klassen zijn.

User interface layer:

- StatisticUI:

Deze klasse is veranwoordelijk voor het tekenen van de statistieken. StatisticUI haalt zijn gegevens uit klasse Statistics.

ReservationUI:

-Het maken van een reservering gebeurt in deze klasse. Deze klasse tekent dus de verschillende invoerschermen die je ziet bij het maken van een reservering.

-StressTestUI:

Deze klasse tekent de benodigde formulieren van de StressTest.

-OverviewUI:

De klasse tekent het hoofdscherm. Hierin zie je alle shuttles naar hun bestemming gaan.

Data layer:

- Database:

Hier word alle data opgeslagen die te maken heeft met de statistieken die getekend moeten worden.

Denk hierbij o.a. de gemiddelde shuttle bezetting e.d.

- Reservation:
Alle informatie van een reservering die gemaakt is door de gebruiker van de simulatie word in deze klasse opgeslagen.

- SimDate:
Deze klasse bevat de datum en tijd die binnen de simulatie wordt gehanteerd.

Business layer:
- StressTest:
Deze klasse zorgt er voor dat het mogelijk is grote aantallen mensen tegelijkertijd in het systeem in te voeren. Hiermee is het mogelijk om het systeem te load-testen.

- Statistics
Deze klasse voert de berekeningen uit voor de statistieken. Deze data haalt hij uit klasse Database. Later worden deze statistieken getekend in StatisticsUI.

- Shuttle:
Deze klasse bezit alle waardes van een shuttle. Zoals o.a. de accelaration, brakeDistance en amountPassenger.

- ShuntArea:
Dit is het rangeeterrein. Deze klasse regelt het doorgeven van shuttles naar andere ShuntArea's en beheert het aantal gereserveerde en vrije shuttles.

- Central:
Vanuit deze klasse worden alle andere klasses aangestuurd. Deze klasse zal dan ook de Main() methode bevatten.

- Perron:
Deze klasse zorgt voor het in en uitstappen van Passenger.

- Station:
De klasse bevat de perrons. Station kan een bestemming of een vertrekpunt van Passenger en Shuttle zijn.

-Terminal:
Dit is de superklasse van StationTerminal en PerronTerminal en bevat dus een groot aantal van de methoden van die twee klasses.

- StationTerminal:
Deze klasse handeld de code af die ingetikt word door passenger en geeft dan een Perron op en geeft de tijd van vertrek van Shuttle. Zo weet Passenger waar hij heen moet,

- PerronTerminal:
Deze klasse handeld de code af die ingetikt word door de passenger. Hierna opent hij de deuren.

-Passenger:
Deze klasse stelt de passagier voor en beheert de incheck- en depart-Time.

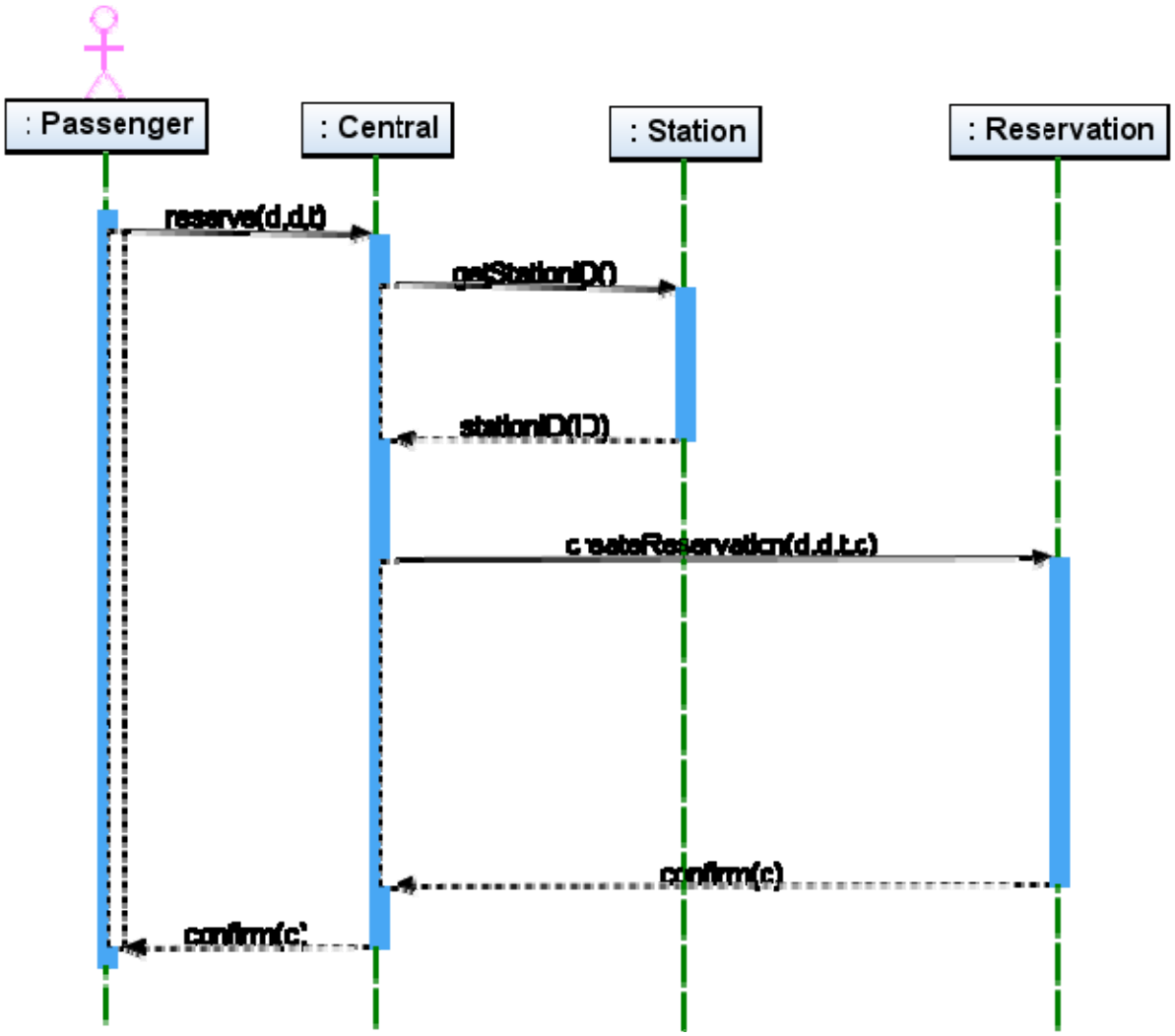
Data laag design

In project Railcab zal niet gebruik gemaakt worden van een aparte server met daarop een database. In plaats daarvan is er een aparte data package gemaakt. In deze package staan : Database, Reservation en SimDate (zie Klasse Details).

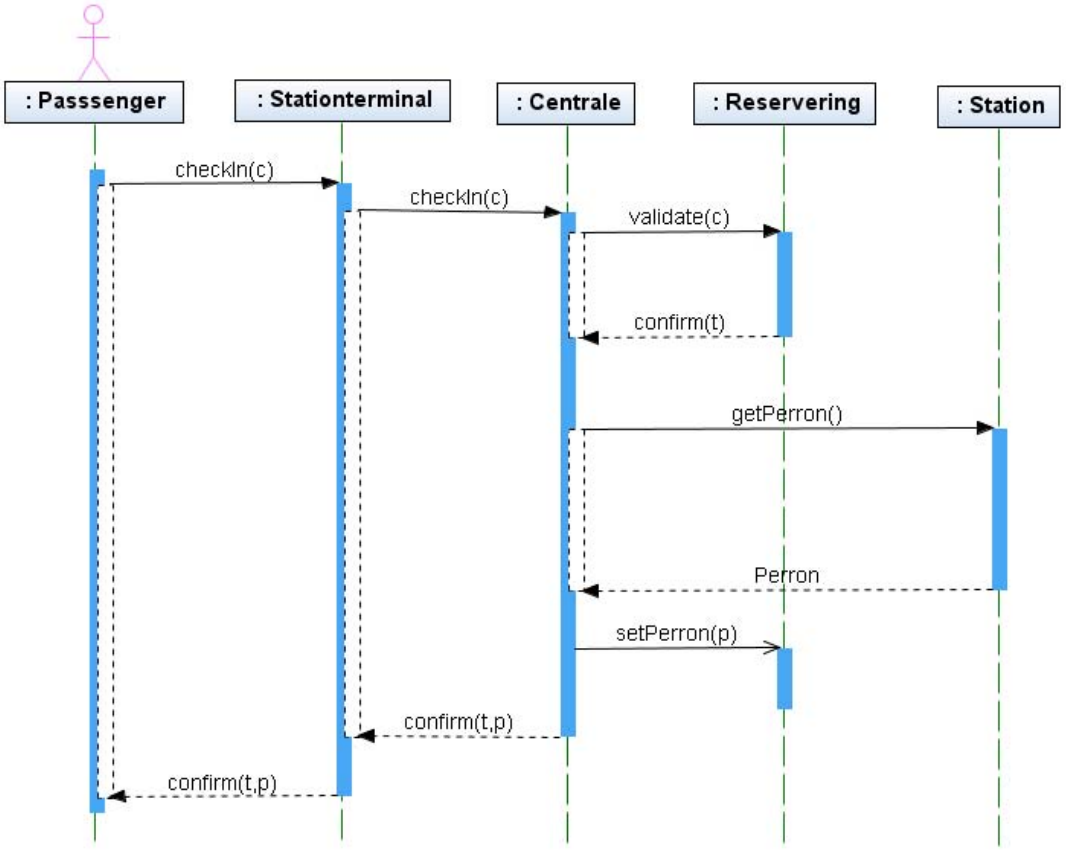
De communicatie tussen deze lagen zal niet geschieden via SQL query's, maar door het aanroepen van methodes.

Sequence diagram

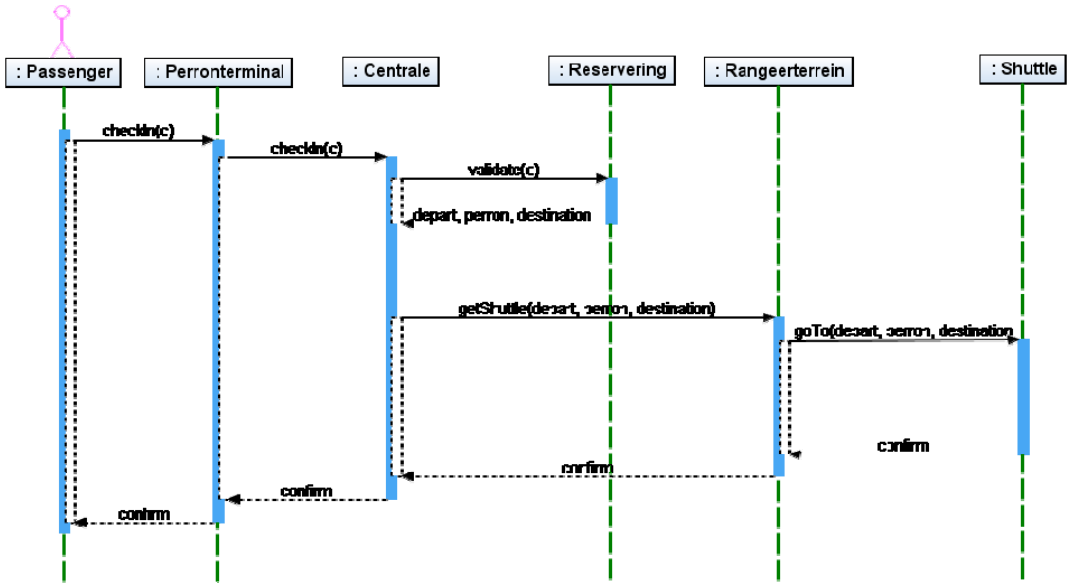
Reservation



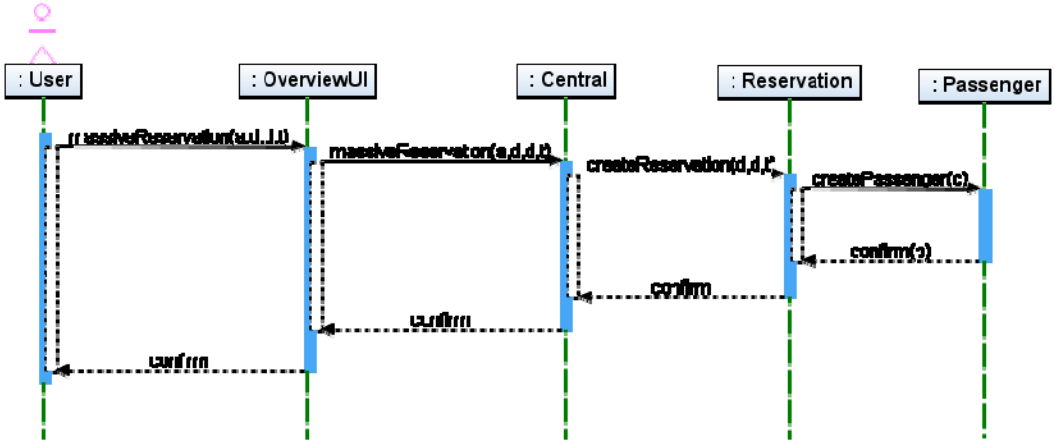
Checkin
Station



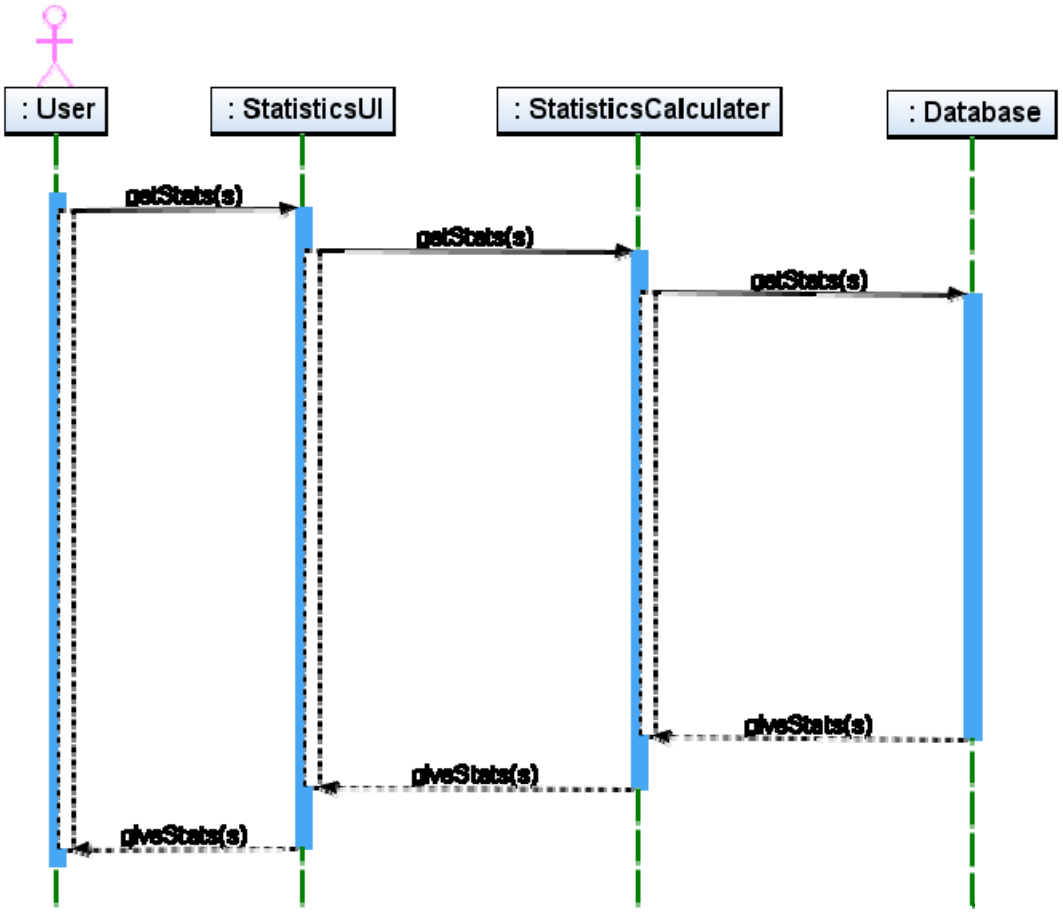
Checkin
Perron



Stress
test



Statistics



Api definitie

In dit hoofdstuk zie je via welke waardes de verschillende layers communiceren.

Business Layer > Data Layer

createReservation(Station department, Station destination, simDate departDate, String code):Boolean confirmation;	Het creëren van een reservering.
checkStationLogin(String code):SimDate departDate;	Functie om de inlogcode en reservering te controleren voor het inloggen op een stationterminal. Stuurt een data object terug met (geplande) vertrek tijd.
checkPerronLogin(String code):Boolean allowed	Functie om te bepalen of een passagier toegang heeft tot het perron.
addPersonWaiting(Station station); removePersonWaiting(Station station);	Hoogt /verlaagt het aantal personen wachtend op. (alleen statistiek gezien)
addPersonTraveling(Station station); removePersonTraveling(Station station);	Hoogt /verlaagt het aantal personen reizend op (alleen statistiek gezien)

addPersonTransfered(Station station);	Hoogt het aantal personen, die reeds vervoerd zijn (alleen statistiek gezien)
addTimeToWait(simDate date, Station station);	Methode om de gemiddelde wachttijd toe te voegen / berekenen
addTimeOfDelay(simDate date, Station station);	Methode om de gemiddelde vertraging toe te voegen / berekenen
getTimeToWait(Station station):XYDateset averageTimeToWait;	Zie *getPersonsWaiting()*
getTimeofDelay(Station station):XYDateset averageTimeOfDelay;	Zie *getPersonsWaiting()*
addShuttleEmpty(SimDate date); removeShuttleEmpty(SimDate date);	Hoogt / verlaagt het aantal lege shuttles op
addShuttleTravelingEmpty(SimDate date); RemoveShuttleTravelingEmpty(SimDate date);	Hoogt / verlaagt het aantal lege shuttles op die momenteel aan het reizen zijn
addShuttleEquiped(SimDate date); removeShuttleEquiped(SimDate date);	Hoogt / verlaagt het aantal bemande shuttles op
addShuttleTravelingEquiped(SimDate date); removeShuttleTravelingEquiped(SimDate date);	Hoogt / verlaagt het aantal bemande shuttles op die momenteel aan het reizen zijn.
getPersonsWaiting(Station station):XYDataset personsWaiting;	Geeft een dataset terug welke gebruikt kan worden in de User Interface. **Deze XYDataset komt uit JFreeChart library!**
getPersonsTraveling():XYDateset personsTraveling;	Zie *getPersonsWaiting()*
getPersonsTransfered(Station station):XYDateset personsTransfered;	Zie *getPersonsWaiting()*
getShuttlesEmpty():XYDateset shuttlesEmpty;	Zie *getPersonsWaiting()*
getShuttlesTravelingEmpty():XYDateset shuttlesTraverlingEmpty;	Zie *getPersonsWaiting()*
getShuttlesEquiped():XYDateset shuttlesEquiped;	Zie *getPersonsWaiting()*
getShuttlesTravelingEquiped():XYDateset shuttlesTravelingEquiped;	Zie *getPersonsWaiting()*

User Interface Layer > Business Layer

createReservation(Station department, Station destination, simDate departDate, String code):Boolean confirmation;	Het creëren van een reservering.
checkStationLogin(String code):SimDate departDate;	Functie om de inlogcode en reservering te controleren voor het inloggen op een stationterminal. Stuurt een data object terug met (geplande) vertrek tijd.
checkPerronLogin(String code):Boolean allowed	Functie om te bepalen of een passagier toegang heeft tot het perron.
getTimeToWait(Station station):XYDateset averageTimeToWait;	Zie *getPersonsWaiting()*
getTimeofDelay(Station station):XYDateset averageTimeOfDelay;	Zie *getPersonsWaiting()*
getPersonsWaiting(Station station):XYDataset personsWaiting;	Geeft een dataset terug welke gebruikt kan worden in de User Interface. **Deze XYDataset komt uit JFreeChart library!**

getPersonsTraveling():XYDateset personsTraveling;	Zie *getPersonsWaiting()*
getPersonsTransfered(Station station):XYDateset personsTransfered;	Zie *getPersonsWaiting()*
getShuttlesEmpty():XYDateset shuttlesEmpty;	Zie *getPersonsWaiting()*
getShuttlesTravelingEmpty():XYDateset shuttlesTraverlingEmpty;	Zie *getPersonsWaiting()*
getShuttlesEquiped():XYDateset shuttlesEquiped;	Zie *getPersonsWaiting()*
getShuttlesTravelingEquiped():XYDateset shuttlesTravelingEquiped;	Zie *getPersonsWaiting()*

Klassen specificaties

Invarianten

Klasse	*Invarianten*
Passenger	Geen
Reservation	Geen
Shuttle	acceleration, brakeDistance
Station	stationID, StationTerminal
Perron	perronTerminal, perronID
ShuntArrea	capacity, shuntAreaTable
Central	stations, shuttles
Terminal	Geen
TerminalStation	station, stationTerminalID
TerminalPerron	perron, perronTerminalID
SimDate	Geen
Overview	Geen
Statistics	amountShuntArea
StressTest	Geen
StressTestUI	Geen
StatisticsUI	Geen
Database	Geen

Pre-condities en Post-condities

Methode	Pre condities	Post condities
PerronTerminal.checkInPerron(code)	Een passenger is ingechecked op het station. En bevind zich op het juiste perron.	Passenger word toegelaten op het perron.
StationTerminal.checkInStation(code)	Een passenger heeft een reservering en de bijbehorende code.	Passenger krijgt vertrektijd te zien en op welk perron zijn shuttle vertrekt.
Centrale.cReservation(departstation, destinationstation, time)	*Departstation en destinationstation moeten bestaan. Er dienen minimaal 2 stations aanwezig te zijn in het systeem.*	Passenger ontvangt een code. Er is een shuttle gereserveerd voor de passenger.
StressTest.massiveReservation(number, departmentstation, destinationstation, time)	*Departstation en destinationstation moeten bestaan. Er dienen minimaal 2 stations aanwezig te zijn in het systeem.*	De Passengers ontvangen hun codes. Er worden voor alle Passengers shuttles gereserveerd.
getPersonsWaiting(Station station):XYDataset personsWaiting;	*geen*	Geeft een dataset terug welke gebruikt kan worden in de User Interface. **Deze XYDataset komt uit JFreeChart library!**
getPersonsTraveling():XYDateset personsTraveling;	*geen*	Zie *getPersonsWaiting()*
getPersonsTransfered(Station station):XYDateset personsTransfered;	*geen*	Zie *getPersonsWaiting()*
getShuttlesEmpty():XYDateset shuttlesEmpty;	*geen*	Zie *getPersonsWaiting()*
getShuttlesTravelingEmpty():XYDateset shuttlesTraverlingEmpty;	*geen*	Zie *getPersonsWaiting()*
getShuttlesOccupied():XYDateset	*geen*	Zie *getPersonsWaiting()*

shuttlesEquiped;		
getShuttlesTravelingOccupied():XYDateset shuttlesTravelingEquiped;	*geen*	Zie *getPersonsWaiting()*

Exceptie aanpak

In dit onderdeel zal aan bod komen hoe ons programma om zal gaan met excepties.

-Het fout invullen van de reservering:

Het systeem zal een bericht terug sturen met daarin dat de aanvraag niet behandeld kan worden door een verkeerde invoer. In dit bericht zal ook staan hoe de informatie wel ingevoerd moet worden.

-Het fout invullen van de code bij de stationterminal.

-Wanneer de code niet in de database voorkomt zal terug worden gekoppeld dat de code onjuist is. Bij deze terugkoppeling word ook weer gegeven wat de mogelijke oorzaken zijn:

-De reservering is verlopen.

-De code is onjuist.

-Wanneer de code wel in de database voorkomt, maar niet van de betreffende reiziger is, zal de reiziger moeten letten op de terugkoppeling die het systeem geeft. In deze terugkoppeling staat alle informatie van de reis, deze moet de reiziger accepteren voor hij te zien krijgt naar welk perron hij moet. Mocht deze informatie niet juist zijn, dan zal de reiziger terug moeten om nogmaals de code in te voeren.

-Het fout invullen van de code bij de perronterminal.

-Wanneer de reiziger bij het verkeerde perron staat en daar zijn code in de terminal in voert, zal de reiziger er op worden gewezen dat hij of zij bij het verkeerde perron staat. Ook zal er terug worden gegeven bij welk perron de reiziger wel moet zijn.

-Wanneer de reiziger bij de perron terminal een onjuiste code invoert die niet in de database voorkomt, zal er terug worden gekoppeld dat de code onjuist is en deze opnieuw moet worden ingevoerd.

-Het te laat aankomen op het station voor de gereserveerde reis.

-Wanneer de reiziger te laat zijn code invoert op de stationsterminal zal deze terugkoppelen dat de code onjuist is. Bij deze terugkoppeling zullen de twee mogelijke oorzaken staan:

-De reservering is verlopen.

-De code is onjuist.

Verklarende woordenlijst

Tier
Een tier is een fysiek onderdeel (server, computer die een service verleend, et cetera) benodigd om het gehele systeem naar behoren te laten functioneren.

User Interface
De user interface zorgt voor een vriendelijke weergave voor de simulatie. Het zorgt ervoor dat de gebruiker van de simulatie alle informatie overzichtelijk tot zich krijgt.

Packages
Een package dient ervoor om het overzicht te bewaren voor ontwikkelaars. Door klassen in te delen in diverse packages moet het programmeren van dit systeem eenvoudiger en duidelijker worden.

Threads
Een thread op een computer is een proces dat binnen een proces uitgevoerd wordt. Met behulp van threads kan een computerprogramma verschillende taken "tegelijkertijd" uitvoeren net zoals in een besturingssysteem met multitasking meerdere processen tegelijk kunnen draaien.

Encryptie
Het versleutelen van data zodat deze niet direct leesbaar is.