VAKOVERSCHRIJDEND EINDPROJECT

**Racespel**Team 11

**Jan De Geest**

**Michaël Depover**

**Thomas Debrauwer**

Inhoudsopgave

1. [1. Abstract 2](#_Toc388227449)
2. [1.1. Nederlands 2](#_Toc388227450)
3. [1.2. English 2](#_Toc388227451)
4. [2. Situering 3](#_Toc388227452)
5. [3. Taakverdeling 3](#_Toc388227453)
6. [4. Analyse 5](#_Toc388227454)
7. [4.1. ER-diagram 5](#_Toc388227455)
8. [4.2. Use Cases 5](#_Toc388227456)
9. [4.3. UML ontwerpen 13](#_Toc388227457)
10. [4.3.1. Activity diagrammen 13](#_Toc388227458)
11. [4.3.2. Use casediagrammen 15](#_Toc388227459)
12. [4.3.3. Deployment diagram 17](#_Toc388227460)
13. [5. Ontwerpbeslissingen 17](#_Toc388227461)
14. [5.1. Singleton pattern 17](#_Toc388227462)
15. [5.2. Grafische user interface 17](#_Toc388227463)
16. [5.3. DAO Design 18](#_Toc388227466)
17. [6. Implementatiekeuzes 18](#_Toc388227467)
18. [6.1. Webapplicatie 18](#_Toc388227468)
19. [6.2. Desktop 18](#_Toc388227469)
20. [6.2.1. Jersey REST client API 18](#_Toc388227470)
21. [6.2.2. Concurrency in Swing 19](#_Toc388227471)
22. [6.2.3. FEST (Fixtures for Easy Software Testing) 19](#_Toc388227472)
23. [7. Kwaliteitscontrole 20](#_Toc388227473)
24. [8. Gekende problemen, verbeterpunten en uitbreidingsmogelijkheden 21](#_Toc388227474)
25. [9. Nabeschouwing en besluit 22](#_Toc388227475)

# Abstract

# Nederlands

Dit projectdossier handelt over de uitwerking van een vakoverschrijdend project (VOP). Voor dit project werd de opdracht gegeven om een 2D racegame te ontwerpen. De applicatie bestaat uit een web- en desktopgedeelte, beiden gekoppeld aan MySQL databank. Voor de desktopapplicatie gebruiken we Java Swing, in het webgedeelte wordt Representational state transfer (REST) en JavaServer Pages (JSP) gebruikt.

Eerst bespreken we hoe deze opdracht zich situeert in de opleiding van Industrieel Ingenieur Informatica. Daarna wordt de taakverdeling binnen het project besproken. De taken zijn onderverdelingen van de gevraagde features die gevraagd werden. Aan de hand van backlogs die werden bijgehouden tijdens het project maken we daar een overzicht van.

Voor delen van de features van het project was het noodzakelijk om een analyse te maken. Door use cases gekoppeld aan verschillende diagrammen geeft de analyse een overzicht in hoe verschillende programmaonderdelen werken. Na de analyse bespreken we interessante keuzes en beslissingen die we genomen hebben inzake ontwerp en implementaties.

Daarna gaan we over naar het testen. Hierbij wordt besproken op welke manier we de belangrijke delen van het programma onderworpen hebben aan verschillende testen. Vervolgens is er een overzicht van de gekende problemen en onderdelen die nog voor verbetering vatbaar zijn. Ook eventuele uitbreidingsmogelijkheden worden uitgelicht. Ten slotte geven de teamleden een korte nabeschouwing van hun ervaring.

# English

This project dossier handles about how we made a 2D-race game. This race game was a project for an multicourse project. The race game consists of two parts: a web-application and desktop-application, using a MySQL database. The desktop application uses Java Swing while the web part uses Representation state transfer(REST) and JavaServer Pages(JSP).

At first we situate this assignment in our educational journey to an engineer’s degree in Computer Science. Further on the division of labor among the team is discussed. This was divided in features which the team subdivided in individual tasks for each member. The overview of the division is made up from backlogs who were kept along the work.

For the important parts of the application it was necessary to do a proper analysis. By making use cases which related to other diagrams this gives a review how different parts work in this application. After the analysis the important choices and discussions we made about design and implementations are written down.

After that we move on to the testing. Here the manner of testing is discoursed. Then we have an analysis about the known problems or bugs that exist. Also possible expansions and improvement points are discussed. Finally the team members give an experience review about their assignment.

# Situering

Deze opdracht heeft de bedoeling om de studenten in teamverband succesvol een gedistribueerde applicatie te laten ontwerpen. Binnen deze opdracht moeten de studenten ontwerpen, realiseren, demonstreren en presenteren van de applicatie. Hierbij moeten de studenten gebruik maken van softwareontwikkelingsmethodologieën en –tools. Door verschillende disciplines van informatica te combineren leren zij een volwaardige applicatie af te leveren die meertalig is en bestaat uit een desktop- en webapplicatie gebruik makend van een databank.

Door in een realistische omgeving te werken aan deze applicatie is dit een belangrijk onderdeel van de opleiding.

# Taakverdeling

### Sprint 1 (10-02-2014 - 24-02-2014)

|  |  |  |
| --- | --- | --- |
| Status | Feature | Assignee |
| Resolved | De gebruiker maakt een account aan | Sander Meyfroot |
| Resolved | De gebruiker logt in op de webapplicatie | Sander Meyfroot, Michaël Depover |
| Resolved | De gebruiker logt in op de desktopapplicatie | Jan De Geest |
| In Progress | De circuitontwerper maakt een nieuw circuit aan | Thomas De Brauwer |
| In Progress | De circuitontwerper bewerkt een van zijn reeds aangemaakte circuits | Thomas De Brauwer |
| In Progress | De circuitontwerper plaatst wegtegels in het raster | Thomas De Brauwer |
| In Progress | De circuitontwerper geeft de starttegel en -richting aan | Thomas De Brauwer |
| In Progress | De speler kiest een circuit | Jan De Geest |
| Resolved | Opvullen databank met spelers | Michaël Depover |
| In Progress | De circuitontwerper slaat het huidige circuit op | Michaël Depover, Thomas De Brauwer |

### Sprint 2 (06-03-2014 - 24-04-2014)

|  |  |  |
| --- | --- | --- |
| Status | Subject | Assignee |
| Resolved | De circuitontwerper slaat het huidige circuit op | Jan De Geest |
| Resolved | De circuitontwerper bewerkt een van zijn reeds aangemaakte circuits | Jan De Geest, Thomas De Brauwer |
| Resolved | De circuitontwerper geeft de starttegel en -richting aan | Thomas De Brauwer, Jan De Geest |
| Resolved | De speler rijdt rondjes op het circuit. Het spel houdt het traject van zijn snelste ronde bij. | Michaël Depover |
| Resolved | De speler herbekijkt het traject van zijn snelste ronde. | Michaël Depover |
| Resolved | De speler bekijkt de snelste rondetijden voor een circuit. | Michaël Depover |
| Resolved | De speler kiest de ghosts die hij onder het racen te zien wil krijgen. | Michaël Depover |
| Resolved | De gebruiker wijzigt de weergavetaal | Thomas De Brauwer |
| In Progress | De gebruiker doorzoekt de openbare gebruikerslijst | Sander Meyfroot |
| In Progress | De gebruiker bekijkt een openbaar profiel. | Sander Meyfroot |
| In Progress | De gebruiker maakt zijn/haar profiel openbaar. | Sander Meyfroot |
| Resolved | De gebruiker organiseert een toernooi, bestaande uit meerdere races. | Jan De Geest |
| Resolved | De speler neemt deel aan eeen toernooi. | Jan De Geest |
| Resolved | Indien het circuit niet op het scherm van de speler past, geeft het spel een zo groot mogelijk gebied rond de wagen weer. | Michaël Depover |
| Resolved | De speler pauzeert het spel. | Michaël Depover |

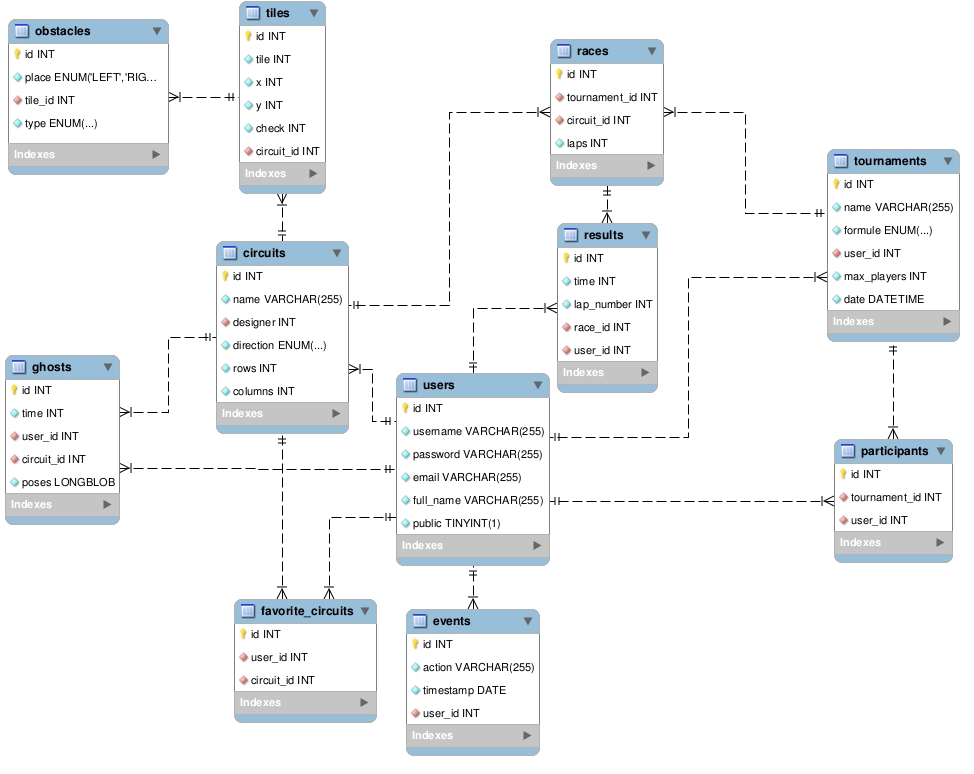
### Sprint 3 (25-04-2014 - 15-05-2014)

|  |  |  |
| --- | --- | --- |
| Status | Subject | Assignee |
| Resolved | De speler voegt een circuit toe aan zijn favorietenlijst. | Thomas De Brauwer |
| Resolved | De speler verwijdert een circuit uit zijn favorietenlijst. | Thomas De Brauwer |
| Resolved | De speler kiest (per race of per toernooi) een wagen. Zijn/haar keuze beinvloedt de snelheid, versnelling en stuurinrichting tijdens het racen. | Michaël Depover |
| Resolved | De speler neemt deel aan eeen toernooi. | Michaël Depover |
| Resolved | Het systeem houdt een overzicht bij van relevante gebeurtenissen. | Jan De Geest |
| In Progress | De gebruiker logt in met Facebook. | Thomas De Brauwer |
| In Progress | De circuitontwerper plaatst een kruispunt op het circuit. Via checkpoints geeft hij/zij aan wat de te volgen route is. | Jan De Geest |
| Resolved | De circuitontwerper plaatst obstakels en/of bonussen op het circuit. | Jan De Geest |
| Resolved | De circuitontwerper tekent een circuit met zijn vinger of met de muis. Het systeem kiest automatisch de gepaste tegels. | Jan De Geest |
| Resolved | Circuitbuilder in webapp hermaken met HTML5 canvas | Jan De Geest |

Gegeven is de taakverdeling a.d.h.v de gerealiseerde features. Voor een gedetaileerd logboek met een takenbord wordt verwezen naar de project website.

# Analyse

# ER-diagram



# Use Cases

### De circuitontwerper maakt een nieuw circuit aan

### Beschrijving

De gebruiker kruipt in de rol van circuitontwerper een maakt een nieuw circuit aan.

### Prioriteit

Hoog

### Complexiteit met ontwerp en code.

Hoog

### Precondities

* De gebruiker is ingelogd

### Postcondities

* De ontwerper plaatst wegtegels (Zie aparte use case)
* Het circuit is geldig
* Het circuit werd opgeslagen in de database

### Primaire actoren

* Gebruiker in de rol van circuitontwerper

### Successcenario

1. De ontwerper kiest de grootte van het raster (?)
2. De ontwerper geeft een naam aan het circuit
3. De ontwerper legt de nodige wegtegels
4. Het resultaat is een gesloten circuit.
5. De ontwerper submit zijn circuit

### Alternatieve scenario’s

3.a De circuitontwerper legt de tegels foutief op het raster (Zie aparte use case)

4.a Het circuit is niet gesloten, terug naar stap 3.

### De circuitontwerper plaatst wegtegels in het raster

### Beschrijving

Tijdens het ontwerpen van het circuit kiest de gebruiker een wegtegel. Er zijn twee soorten wegtegels, een recht stuk en een L-vormige bocht.

### Prioriteit

Gemiddeld

### Complexiteit met ontwerp en code.

Gemiddeld

### Precondities

* De ontwerper is ingelogd
* De ontwerper heeft reeds een circuit aangemaakt

### Postcondities

* Het circuit wordt geschreven naar de database

### Primaire actoren

* Ontwerper

### Successcenario

1. De ontwerper kiest een reeds bestaand circuit van zijn hand
2. De ontwerper maakt het circuit door de wegtegels op het raster te leggen.
3. Het circuitontwerp is geldig.
4. De ontwerper slaat het circuit op.
5. Het circuit worden weggeschreven naar de databank

### Alternatieve scenario’s

3 a Het ontwerp is niet geldig

5 a De wijzigingen konden niet worden opgeslaan, terug naar stap 4.

### De circuitontwerper bewerkt een van zijn reeds aangemaakte circuits

### Beschrijving

Een ontwerper kan een van zijn vorige ontworpen circuits aanpassen

### Prioriteit

Gemiddeld

### Complexiteit met ontwerp en code.

Gemiddeld

### Precondities

* De gebruiker is ingelogd
* De gebruiker heeft reeds één of meerdere circuits ontworpen

### Postcondities

* De wijzigingen worden upgedate in de database
* De tijden van het circuit vallen weg.

### Primaire actoren

* Ontwerper

### Successcenario

1. De gebruiker selecteert een vorig ontwerp

### De speler rijdt rondjes op het circuit. Het spel houdt het traject van zijn snelste ronde bij

### Beschrijving

De speler rijdt rondjes op het circuit. Na elke afgelegde ronde wordt er gekeken of hij zijn tijd heeft verbeterd

### Prioriteit

Hoog

### Complexiteit met ontwerp en code.

Hoog

### Precondities

* De speler is ingelogd
* De ontwerper heeft reeds een circuit aangemaakt waarop een speler kan rijden

### Postcondities

* Het afgelegde traject van zijn snelste ronde wordt geschreven naar de database

### Primaire actoren

* Speler

### Successcenario

1. De speler kiest een reeds bestaand circuit
2. De speler geeft aan dat hij de race wenst te starten
3. De speler rijdt rondjes op het circuit
4. Het spel houdt het traject van zijn snelste ronde bij
5. Het systeem schrijft het traject van zijn snelste ronde naar de database

### Alternatieve scenario’s

3.a Na elke ronde vragen of de speler nog wenst verder te rijden

### De speler herbekijkt het traject van zijn snelste ronde

### Beschrijving

De speler kan een herhaling zien van zijn snelste ronde van een bepaald circuit

### Prioriteit

Hoog

### Complexiteit met ontwerp en code.

Hoog

### Precondities

* De speler is ingelogd
* De racer heeft minstens één ronde afgelegd op een bepaald circuit

### Postcondities

* De speler vragen of hij zijn replay wil herbekijken of iets ander wenst te doen

### Primaire actoren

* Speler

### Successcenario

1. De speler kiest een reeds bestaand circuit
2. De speler selecteert de herhaling van hemzelf die hij wilt bekijken
3. De speler herbekijkt de herhaling

### De gebruiker organiseert een toernooi, bestaande uit meerdere races. Er zijn verschillende formules mogelijk: hoogste totaalscore, snelste totaaltijd, langste afstand afgelegd in gegeven tijdspanne

### Beschrijving

De gebruiker organiseert een toernooi waar hij een aantal circuits (races) selecteert waarop de spelers zullen rijden. Er zijn verschillende formules mogelijk waarop het eindresultaat dan wordt berekend

### Prioriteit

Hoog

### Complexiteit met ontwerp en code.

Gemiddeld

### Precondities

* De gebruiker is ingelogd
* Het systeem moet al een reeds aantal circuits ter beschikking hebben waaruit de gebruiker kan selecteren

### Postcondities

* De gebruiker selecteer minstens twee circuits
* Het zijn geldige circuits
* Het toernooi moet starten en eindigen met vooraf bepaalde datums

### Primaire actoren

* Gebruiker (circuitontwerper of speler)

### Successcenario

1. De gebruiker geeft een naam aan het toernooi
2. De gebruiker kiest welke formule wordt toegepast op het toernooi
3. De gebruiker legt de start en eind datum van het toernooi vast
4. De gebruiker legt het aantal maximum aantal spelers vast die kunnen deelnemen
5. De gebruiker voegt meerdere circuits toe aan het toernooi
6. De gebruiker submit zijn toernooi

### Alternatieve scenario’s

6.a Geen circuits geselecteerd, terug naar stap 5.

### De speler pauzeert het spel

### Beschrijving

De speler kan tijdens de race het spel pauzeren

### Prioriteit

Gemiddeld

### Complexiteit met ontwerp en code.

Gemiddeld

### Precondities

* De speler is ingelogd
* De racer heeft reeds een race gekozen en gestart

### Postcondities

* De speler kan via een knop het pauze menu weergeven

### Primaire actoren

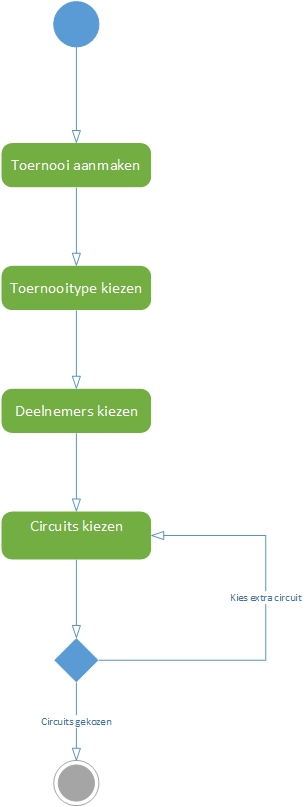
* Speler

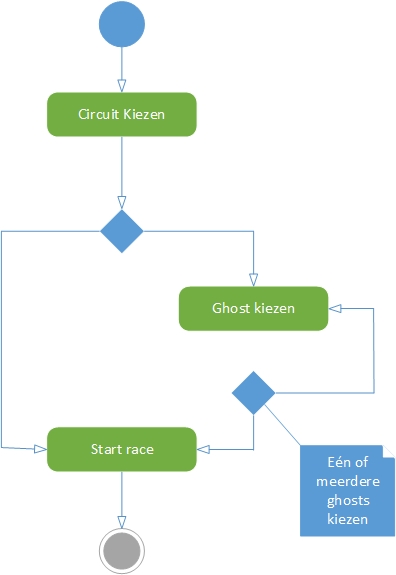
### Successcenario

1. De speler heeft een race reeds gestart
2. De speler geeft de pauze aan via de pauze knop
3. De speler krijgt het pauze menu te zien

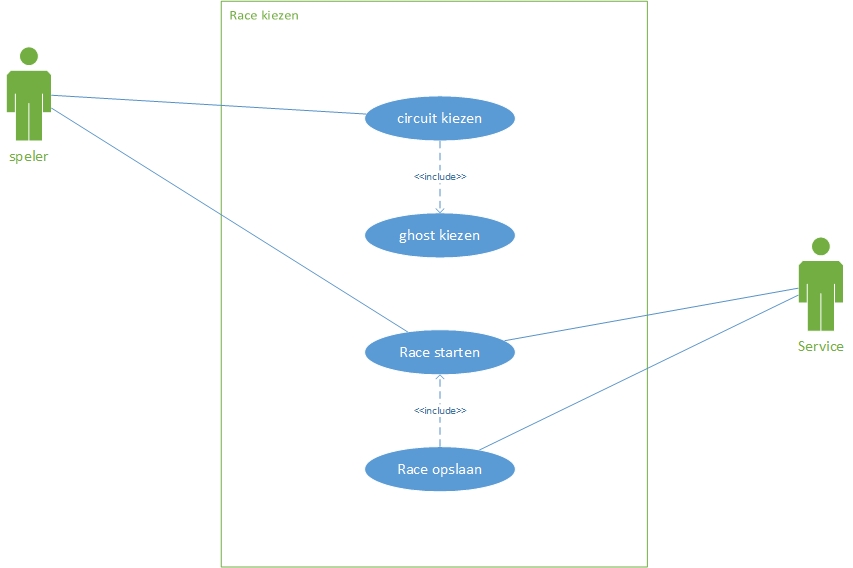
# UML ontwerpen

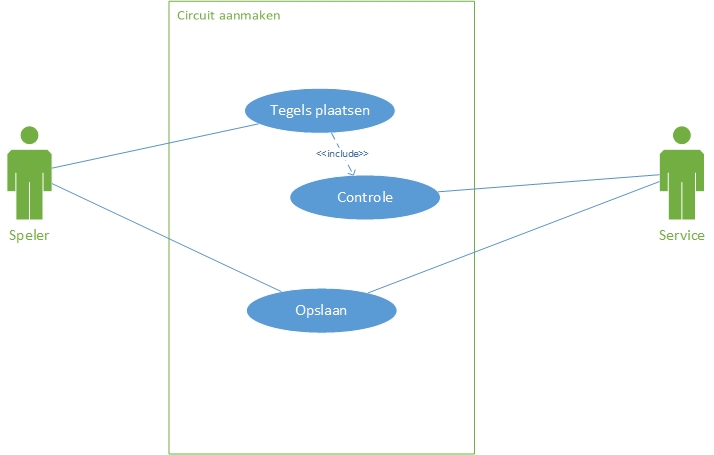
# Activity diagrammen



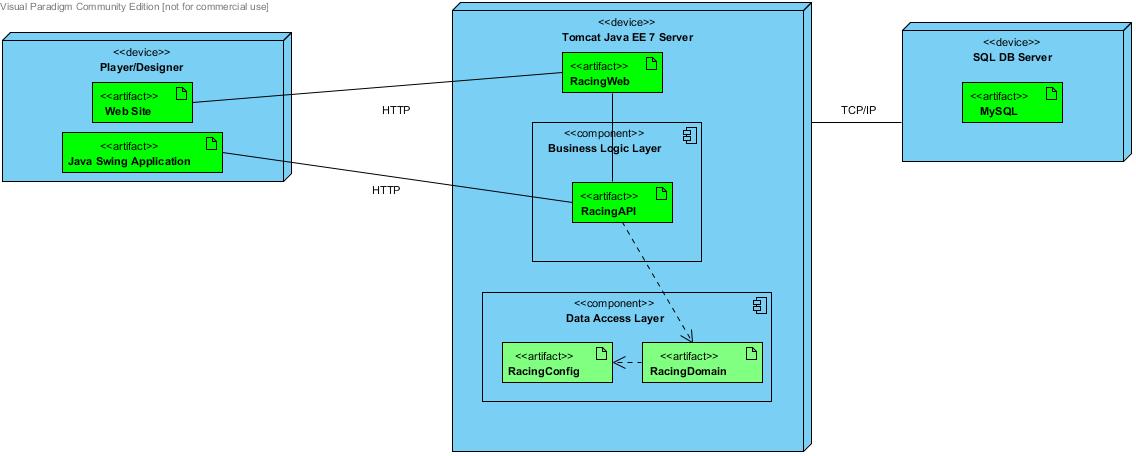


# Use casediagrammen



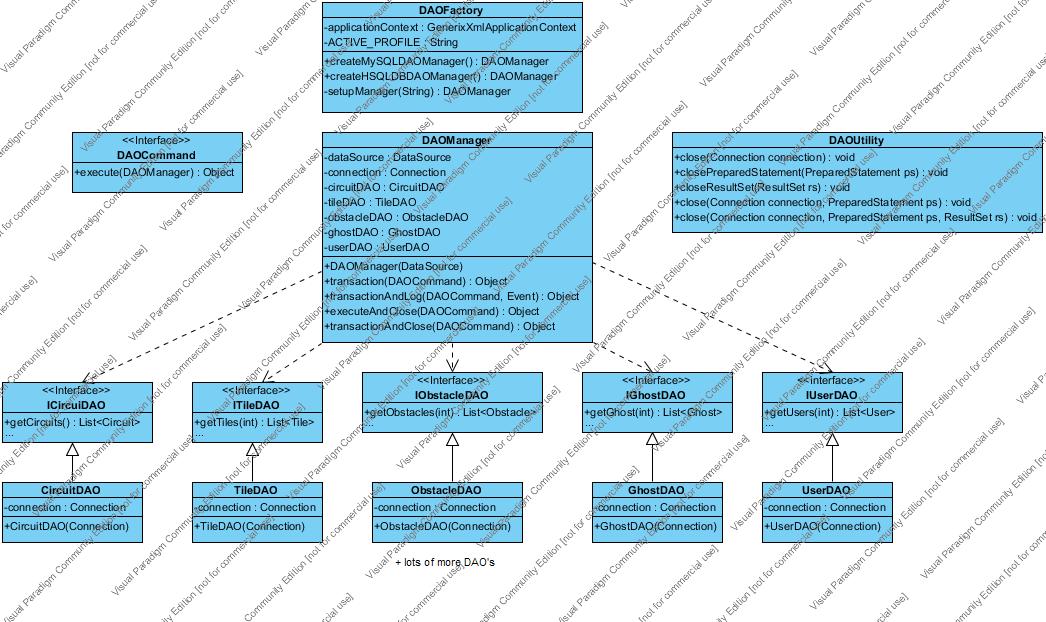


# Deployment diagram

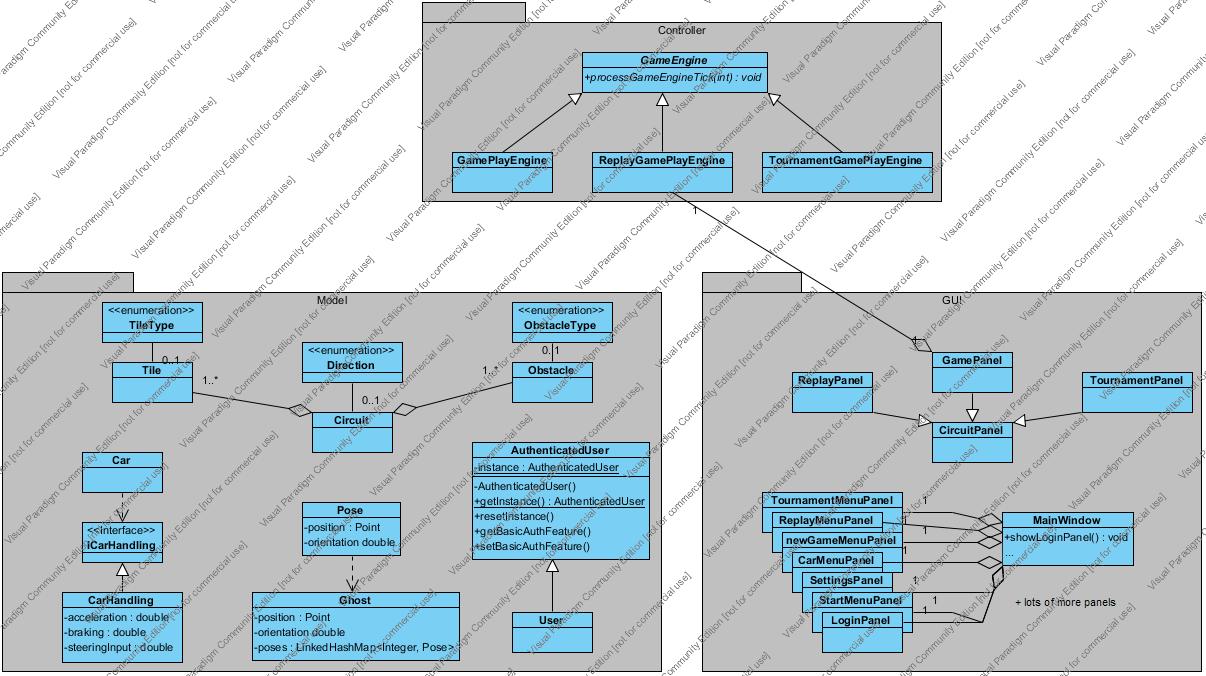


# Klassendiagrammen

### Datalaag klassendiagram



### Desktop klassendiagram



# Ontwerpbeslissingen

# Singleton pattern

Omdat we in onze applicatie de ingelogde user niet zoals in de webapplicatie kunnen bijhouden hebben we gebruik gemaakt van een Singleton pattern. Dit omdat we de gegevens van de gebruiker nodig hebben voor het ontvangen en verzenden van data naar de REST webservice. We hebben een thread-safe singleton voorzien zonder performantie verlies te hebben. Want omdat we in een multi-threaded applicatie werken zorgt het keyword synchronized immers voor sterke reductie van performantie. Dus synchroniseren we enkel bij de eerste oproep van een instantie.

# Grafische user interface

### GridBagLayout

In het project makgen we gebruik van GridBagLayout. Een GridBagLayout is gelijkaardig aan een GridLayout waarin alle componenten mooi in een raster staan met dat verschil dat bij een GridBagLayout individuele cellen van dat raster dynamisch aangemaakt kunnen worden en zelfs de breedte of hoogte van meerdere cellen kunnen innemen.

Het grootste voordeel dat uit een GridBagLayout kan gehaald worden is de dynamiek bij het vergroten of verkleinen van een venster. Alle elementen en tussenliggende lege ruimte zullen daardoor steeds in verhouding blijven, ongeacht de grootte van het venster.



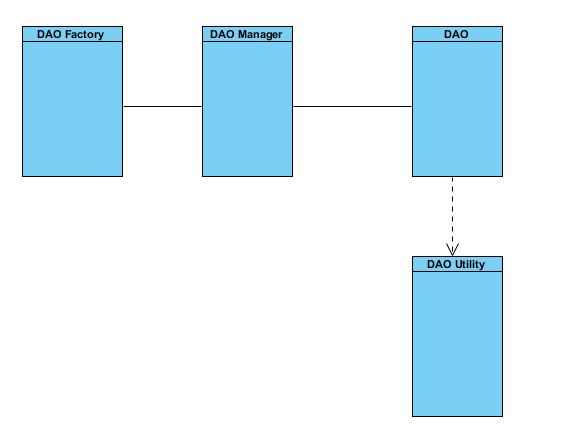
# DAO Design

In de datalaag van het project maken we gebruik van een DAO design pattern. Hierdoor scheiden we alle logica die rechtstreeks met de databank communiceert volledig af van de andere onderdelen in het project. Dit doen we door een abstracte interface te implementeren (de DAO). In de DAO worden alle queries op de databank uitgevoerd. Elke tabel waar we met willen communiceren is gelinkt aan een aparte DAO.

Een belangrijk onderdeel van het DAO design ligt bij de afhandeling van de connecties naar de databank. Na elke query moet deze connectie correct afgesloten worden. Daarom is er gekozen voor een ‘Instance Scope’ connectie afhandeling. Dit wil zeggen dat elke DAO klasse een connectie als veld heeft. Deze connectie wordt na elke interactie met de databank afgesloten, en wordt door elke methode van de klasse gebruikt.

De verschillende DAO’s hebben voor hun constructie dus een connectie als parameter nodig. Daarom worden de DAO klassen altijd aangemaakt en beschikbaar gesteld door de DAO Manager. Deze klasse zorgt ervoor dat de DAO’s telkens een nieuwe connectie krijgen, en geeft de mogelijkheid om gemakkelijk transactions te gebruiken. De DAO Manager is een single point of access voor de DAO’s, dit maakt het gemakkelijk om een DAO te vinden en hergebruiken in de applicatie. De DAO Manager zorgt er ook voor dat de connecties zo laat mogelijk worden geopend en zo snel mogelijk worden gesloten na een interactie met de databank.

In de DAO Factory worden de juiste parameters ingeladen aan de hand van de deployment omgeving. Zo kan er eenvoudig gewisselt worden van omgeving zonder telkens de datasource te moeten aanpassen. De DAO Utility klasse voorzien statische methoden om PreparedStatements, Resultsets en conncties te manipuleren.



# Servlet Request afhandeling

In de webapplicatie maken we veel gebruik van Ajax calls vanuit javascript naar de API om de interactie tussen de client en de server te verzorgen. Omdat er een Basic HTTP Authentication gebruikt wordt voor de beveiliging van deze requests is het niet mogelijk om rechtstreeks vanuit javascript deze ajax calls naar de API te maken (de Authentification string wordt opgebouwd met het wachtwoord dat niet beschikbaar is aan de client side). Daarom worden de ajax calls eerst via de Servlets afgehandeld. Deze servlets kunnen dan vanuit Java gebruik maken van de Jersey REST Client API (zie ook 6.2.1) om de requests naar de API met HTTP Authentication te verzekeren. De Servlets kunnen via zowel GET als POST requests afhandelen, en we maken hier gebruik van om via een extra parameter ‘action’ de gewenste request te selecteren en af te handelen.

Voorbeeld in java code vanuit CircuitServlet:



In deze methode worden alle get parameters afgelopen en op basis van de parameter ‘action’ wordt de correcte API call gemaakt.

# Implementatiekeuzes

# Webapplicatie

# JCanvas

Voor de nieuwe circuit-builder in de webapplicatie is er gekozen om gebruik te maken van JCanvas, een JQuery plugin die toelaat om via een eigen API het HTML5 canvas te bewerken. In de circuit builder is gebruik gemaakt van de Layer API van JCanvas om de tegels uit te tekenen. Dit laat toe om enkele features zoals het tekenen van circuits met muis/hand of het plaatsen van obstakels gemakkelijk te implementeren en de code flexibel te houden.

# Desktop

# Jersey REST client API

Omdat we in de webapplicatie gebruik maakten van Jersey Framework om onze REST service op te zetten moesten we niet ver zoeken om een geschikte client API te voorzien. Jersey bevat eigenlijk een client bibliotheek dat we zowel in onze web client als onze desktop-client applicaties kunnen gebruiken. Jersey is een eigen implementatie van het JAX-RS specificatie die zowel het creeëren van XML en JSON ondersteunt. Via GSON kunnen we de overeenkomstige Java klasse serializeren zodat we dit in onze client applicaties dan weer kunnen deserializeren.

# Concurrency in Swing

Bij het ontwerpen van een Swing-applicatie moet je zeker rekeninghouden met ‘concurrency’. De GUI-interface mag immers niet ‘freezen’. Het programma moet ten alle tijd reageren op de handelingen van de gebruiker.Swing maakt gebruik van één thread, namelijk de EDT (Event Dispatch Thread). Alle GUI-componenten worden immers gecreeêrd door deze thread. Omdat we onze REST server moeten aanspreken om data op te halen of te verzenden maken we gebruik van de Callable interface (vergelijkbaar met de Runnable interface)omdat dit ons toelaat om iets terug te krijgen uit de thread zodat we dit kunnen gebruikten. Door het gebruik van de Future interface kunnen we de status te weten komen van de Callable task en het op te vragen object terug krijgen. We maken ook gebruik van het Executor framework dat een thread pool voorziet in java. Op zo een Executor object kan je dan zo één of meerdere threads uivoeren zodat we verschillende taken tegelijkertijd in parallel of na elkaar kunnen uitvoeren.

# FEST ([Fixtures for Easy Software Testing](https://code.google.com/p/fest/))

Deze module biedt een eenvoudige en intuïtieve API voor het functioneel testen van Swing user interfaces , wat resulteert in tests die compact , makkelijk te schrijven en te lezen zijn. Tests geschreven met FEST zijn ook robuust. FEST simuleert feitelijke gebruiker gebaren op het niveau van het besturingssysteem, zodat de aanvraag correct zal gedragen in de presentatie-zijde van de gebruiker . Het biedt ook een betrouwbaar mechanisme voor het opzoeken van de GUI- componenten die ervoor zorgt dat veranderingen in de GUI van de lay-out of look- and-feel niet de tests zal breken .

# Kwaliteitscontrole

Gedurende het project hebben we gebruik gemaakt van HSQLDB, JUnit en FEST om tests uit te voeren op onze applicaties. Jammer genoeg hebben we dit niet uitgebreid genoeg gedaan en hebben we enkel unit tests geschreven en elke module afzonderlijk getest. Dus integratietesten ontbreken om de verschillende modules met elkaar te testen.

Voordat we begonnen met het testen hebben we zo veel mogelijk het TDD methodologie toe te passen. We probeerden eerst onze tests te schrijven om dan daarna onze werkelijke productie code te schrijven. Hiervoor hadden we natuurlijk ook data nodig en schema die we via Spring aanmaakten. Dit omdat we onze verschillende databanken en hun gegevens ophaalden via Spring. Telkens als we nieuwe aanpassingen in onze databank of domein module deden keken we of onze tests nog werkten. Dit moet ook want anders hadden we moeilijkheden met dit te testen op onze staging tomcat server omdat we door de falende testen niet konden deployen.

Om ons spel (desktop-applicatie) te testen maakten we gebruik van FEST. Dit was wel moeilijk om te testen Bv. voor het simuleren van een race. Daarom hebben we enkel zo goed mogelijk de verschillende menu’s getest en ook of ons pauze menu/dialogs te voorschijn kwamen indien nodig.

# Gekende problemen, verbeterpunten en uitbreidingsmogelijkheden

### Problemen

Bij het circuit ontwerpen:

De implementatie van deze feature gaf problemen voor sommige nieuwe features die voor Sprint 2 geintroduceerd werden. Daarom hebben we deze webpagina volledig vernieuwd voor de finale versie van het project.

Too many user connections?

Uitbreidingsmogelijkheden

Een eerste aantal uitbreidingsmogelijkheden van onze applicatie zijn de features van onze klant die niet (volledig) afgewerkt zijn. Wegens een tekort aan mankracht gedurende het grootste deel van de ontwikkelingsfase zijn volgende toch wel belangrijke uitbreidingen niet in ons project ingewerkt:

* Integratie met Facebook: We zijn hier in de laatste weken aan gestart, maar wegens tijdgebrek hebben we deze uitbreiding niet vervolledigt. De implementatie ligt voor de hand: een mogelijkheid om in te loggen via Facebook in de webapplicatie en het account te linken aan een gebruiker in ons systeem.
* Uitgebreide weergave van highscores: een mogelijkheid voor de gebruikers om steeds de laatste scores te raadplegen.

Verder zijn er een aantal uitbreidingsmogelijkheden die we zelf zouden voorstellen:

* Een multiplayer optie: de mogelijkheid om over het netwerk tegen meerdere spelers te racen. Een mogelijke implementatie hiervan is via Sockets in Java.

-

# Nabeschouwing en besluit

### Eindresultaat

We zijn als groep vrij tevreden over het behaalde product aangezien we voor een groot deel van het project onderbemand waren. Op enkele features na was het project grotendeels afgewerkt, en zijn deze kwalitatief zeker in orde.

### Scrum & Agile

Dit project was een goede kennismaking met de werkmethoden die gebruikt worden in Scrum en Agile. Doordat het werk verdeeld werd in 3 sprints konden we tijdens de tussentijdse evaluatie een goed beeld krijgen van hoever we stonden. Tijdens de eerste sprint hadden we de taken nog niet goed ingeschat, en was het resultaat tegengevallen. Maar hierdoor hadden we voor sprint 2 en 3 een betere taakverdeling, waardoor het werk beter ingeschat kon worden. Hierdoor gaven we de klant realistischere verwachtingen die we in de werkelijkheid konden nastreven.