

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jan De Geest Michaël Depover Thomas De Brauwer Sander Meyfroot |  | Schakeljaar INFORMATICA |  | Sprint 1-2  24/04/2014 |

Racing

Team 11

Inhoudsopgave

1. [Algemeen 2](#_Toc386021853)
2. [Projectstructuur 2](#_Toc386021854)

[Opbouw modulestructuur](#_Toc386021855)

1. [Gebruikte technologiëen 5](#_Toc386021856)
2. [Use Cases 6](#_Toc386021857)

[De circuitontwerper maakt een nieuw circuit aan](#_Toc386021858)

[De circuitontwerper plaatst wegtegels in het raster](#_Toc386021859)

[De circuitontwerper bewerkt een van zijn reeds aangemaakte circuits](#_Toc386021860)

[De speler rijdt rondjes op het circuit. Het spel houdt het traject van zijn snelste ronde bij](#_Toc386021861)

[De speler herbekijkt het traject van zijn snelste ronde](#_Toc386021862)

[De gebruiker organiseert een toernooi, bestaande uit meerdere races. Er zijn verschillende formules mogelijk: hoogste totaalscore, snelste totaaltijd, langste afstand afgelegd in gegeven tijdspanne](#_Toc386021863)

[De speler pauzeert het spel](#_Toc386021864)

# Algemeen

We ontwerpen een racing game waarbij het mogelijk moet zijn om te racen op een circuit, om de snelste rondetijden te rijden en de overwinning te bereiken. Er zijn twee rollen: de circuitontwerper en de speler. Met éénzelfde account kan men beide rollen op zich nemen.

Het ontwerpen van het circuit gebeurt alleen via de webappicatie en speciale functionaliteit is daar beschikbaar:

* De ontwerper kan zelf de grootte van het circuit instellen
* De ontwerper kan zelf de juiste tegels op het raster plaatsen om zo het circuit te bouwen
* De ontwerpet duidt de starttegel en –richting aan

Je kan racen via de desktop-applicatie die de API aanspreekt om het circuit op te bouwen. Je kan niet real-time met je vrienden racen maar via *ghosts* kan je hun afgelegde trajecten herbekijken of zelf tijdens het racen weergeven. De snelste rondetijden van een circuit kunnen ook via de webapplicatie bekeken worden

# Projectstructuur

## Opbouw structuur

We hebben twee projecten:

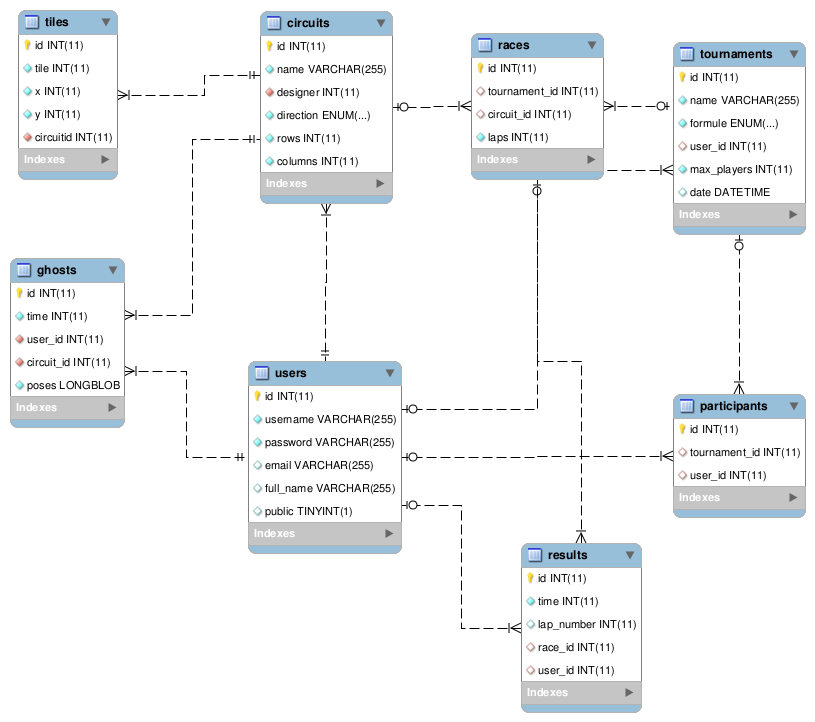
1. Webapplicatie met de website en API service (dus met de bijhorende Backend gedeelte)
2. Desktop-applicatie (Java Swing applicatie)

**1) Webapplicatie met de website en API service**

We gebruiken een Maven parent POM die bestaat uit meerdere child modules:

* RacingDomain

Hier vinden we de persistentielaag terug. Aan de hand van ons ontworpen ER-Diagram definieren we de klassen (pojo’s) die we zullen gebruiken, en implementeren we de juiste interfaces en DAO’s. Via een DAOManager die verkregen wordt via onze DAOFactory kunnen we elke interface van elke DAO op een gemakkelijke manier aanspreken in de controller, deze bevindt zich in de API. Het ER-diagram van de database staat hieronder beschreven. Het voordeel van onze DAOFactory is dat we aan de hand van Spring profiling de desgewenste databank aanspreken eenmaal we kunnen deployen op de Tomcat 7 server. Via de DAOFactory kunnen we ook een DAOManager verkijgen die onze tests uitvoeren op een in-memory database (HSQLD).



* RacingAPI

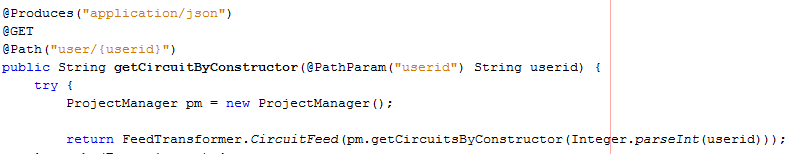
We gebruiken een restfull service om alle aanvragen van de Webapp en Java desktop app te behandelen. Deze service zal de aanvragen behandelen door de controller aan te spreken. Deze zal op zijn beurt naar de DAOManager in het RacingDomain gebruiken om de data uit de database op te halen. Uiteraard wordt er gebruik gemaakt van de nodige authenticatie voor elke aanvraag aan de API. Alle uitwisseling van gegevens van de API gebeurt via JSON.

De authenticatie wordt verzorgt dmv een custom header. Tijdens het aanmelden wordt deze custom header opgevult met een base64 encoding van de gebruikersnaam in combinatie met het passwoord. Deze gegevens zullen bij elke aanvraag aan de service gebruikt worden om de toegang te verkrijgen

De API is aanspreekbaar op volgende manier:

Vb: Circuits ophalen van een gebruiker: <ip>:<port>/circuits/user/<userid>

Om de id uit de url te halen gebruiken we de @PathParam annotation



* RacingWebApp

Voor de acties die via de webapplicatie moeten gebeuren hebben we een apparte module. We zullen alle call’s naar de service laten verlopen via Servlets. Een formulier gaat dus zijn gegevens posten naar een Servlet die dan op zijn beurt de data in JSON naar de service zal sturen. De service zal ten aller tijde zijn antwoord formuleren in JSON, en terug geven naar de Servlet, die dan op zijn beurt de gegevens eventueel kan terug geven naar de webpagina’s. Voor de lay-out gebruiken we HTML5 & CSS3, twitterbootstrap en javascript & jQuery.

**2) Desktop-applicatie**

Via de desktopapplicatie kunnen we rechtstreeks de API aanspreken om de nodige informatie te versturen of ontvangen. We gebruiken hier een apparte klasse waarin alle informatie wordt bij gehouden en dus over de verschillende formulieren en forms kan gebruikt worden. De call’s naar de API gebeuren via threading, omdat de applicatie niet zou vastlopen als de connectie even op zich laat wachten.

# Gebruikte technologiëen

* Webapplicatie (website en API)
  + Java EE 6 (gedeployed op Tomcat 7 Java web server)
  + Maven
  + Spring Framework
  + Twitterbootstrap
  + Javascript & jQuery
  + HTML5 & CSS3
* Desktop-applicatie
  + Java 6
  + Maven
  + Swing
  + FEST (unit tests)
* Database
  + MySql
  + HSQLDB (unit tests)
* API:
  + Restfull service (Jersey Framework)
* Authenticatie (structuur)
  + Jersey met BASIC authenticatie

# 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

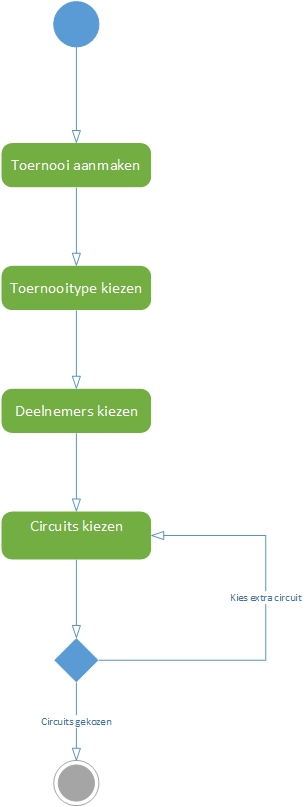
* Speler

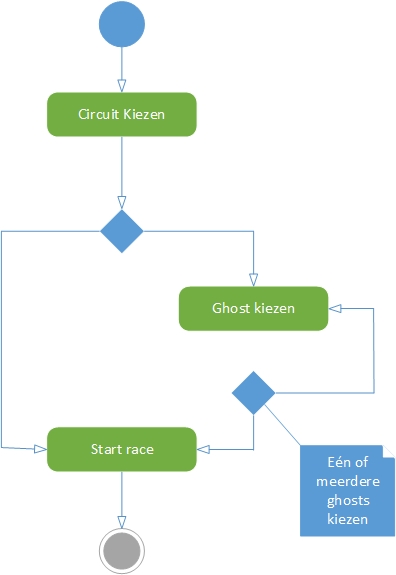
### Successcenario

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

