KillerApp SE2

Ontwerpdocument

Vai van Ommeren

Table of Contents

[Inleiding 3](#_Toc471590796)

[Database Ontwerp 4](#_Toc471590797)

[Domeinmodel 6](#_Toc471590798)

[Toelichting Domein Model. 6](#_Toc471590799)

[Architectuur 7](#_Toc471590800)

[MVC 7](#_Toc471590801)

[Repository Pattern 7](#_Toc471590802)

[Infrastructuur 8](#_Toc471590803)

# Inleiding

In dit bestand vind je de meer technische beslissingen en uitwerkingen zoals het database ontwerp, het domeinmodel en informatie over de infrastructuur en architectuur.

Dit document zal worden bijgehouden indien er later in de ontwikkelingsfase aanpassingen worden gedaan aan het oorspronkelijke ontwerp zodat er altijd duidelijk informatie te vinden is over de applicatie.

# Database Ontwerphttps://lh6.googleusercontent.com/3FSIh7dII9Us-uGSxpL89mX3EKqp5SBgPPR8ixGJFcgo99T0PwpgJ6zghmlty_mgy6DAOuaV7chTCNAl6Ke4-TT8QbEfXUbBbZVNxLN2FJ0RoHQ0zwGvTa_5T8LeuoT9nPWlBFbe

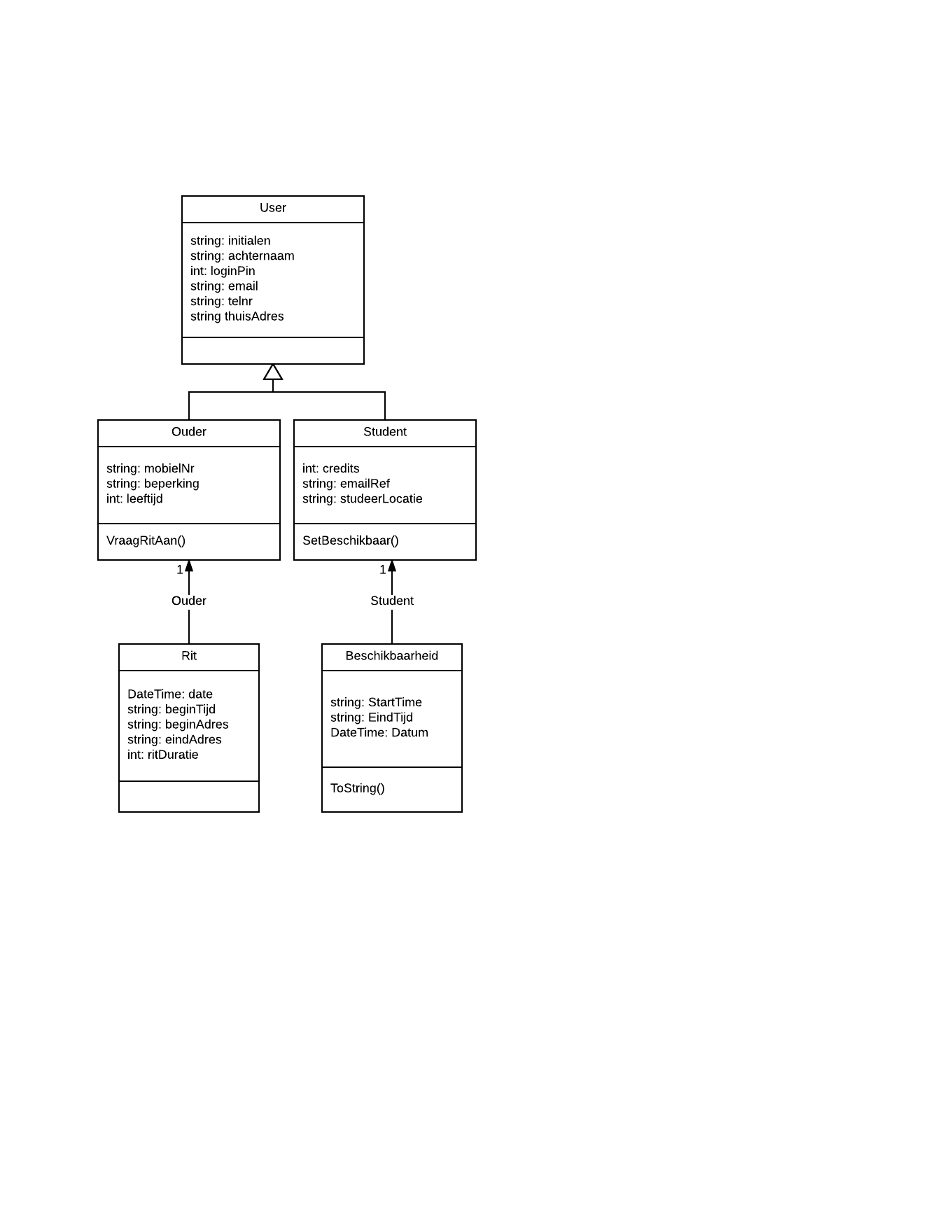
Op de foto op de vorige pagina is het databaseontwerp van de applicatie te vinden.

Hier zal ik nog wat toelichten over dit ontwerp.

Er word gebruik gemaakt van de volgende constraints:

* UNIQUE:
  + Unique word bij meerdere velden gebruikt, een paar voorbeeld zijn;
    - Email
      * Hierbij word een unique gebruikt omdat de email ook gebruikt wordt om in te loggen, je wil dus geen duplicaten hebben.
    - Telnr
      * Een telefoonnummer mag niet gekoppeld zijn aan meerdere mensen omdat het dan verwarrend kan worden verschillende data door te geven
* NOT NULL:
  + Er word bij meerdere velden gebruikt gemaakt van NOT NULL
    - LoginPin
      * Er moet verplicht een wachtwoord zijn om in te loggen, dit veld mag dus niet leeg gelaten worden want dan is er geen wachtwoord.
    - Email
      * Een Email adres is nodig om in te loggen, dit veld mag dus niet leeg gelaten worden want dan kan er niet worden ingelogd
* CHECK:
  + Er word een CHECK toegepast op de studeerlocatie in de tabel UserStudent, dit word gedaan omdat er alleen studenten in omgeving Eindhoven mee mogen doen aan StuMobiel, mocht de locatie dus afwijken van Eindhoven is het dus niet mogelijk.
* DEFAULT:
  + DEFAULT word toegepast bij de aantal zitplaatsen van de auto. Er zullen in het begin alleen maar autos beschikbaar zijn met max 4 zitplaatsen, maar er is later mogelijk voor uitbreiding. De waarde bij zitplaatsen zal dus altijd minstens 4 zijn en is een DEFAULT handig om te gebruiken
* PRIMARY KEY:
  + Primary keys worden bij elke tabel gebruikt als Identifiers, dit werkt ook weer in combinaties met foreign keys en helpt bij het combineren tabellen bij het uitvoeren van complexere queries
* FOREIGN KEY:
  + Foreign keys worden gebruikt voor het koppelen van verschillende tabellen, het zorgt er ook voor dat er in de rit informatie alleen maar users gebruikt kunnen worden die daadwerkelijk bestaan. Zo blijft de integriteit van je database intact.

# Domeinmodel



## Toelichting Domein Model.

In de bijgevoegde foto is het domein model van de applicatie te zien.

Er is gekozen om gebruik te maken van inheritance, dit omdat ook al deelde student en ouder veel overeenkomsten ze toch te veel verschillen hadden om ze als 1 user door te laten gaan.

Om deze 5 main classes zal ook nog het repository pattern en mvc omheen gebouwd worden.

# Architectuur

Om er voor te zorgen dat de website zo ‘interchangeable’ mogelijk is word er gebruik gemaakt van een aantal technieken die hier voor zorgen.

## MVC

MVC is afgeleid van de drie sleutelwoorden binnen dit patroon: Model – View – Controller.

Dit bied je de volgende mogelijkheden:

* Scheiding tussen datamodel, applicatielogica, applicatie presentatie(GUI)
* Hergebruik van code wat tijd bespaard
* Gemakkelijk een nieuwe applicatie presentatie maken(GUI)

Je **Views** vertegenwoordigd je GUI.

Je **Model** vertegenwoordigen het datamodel, het is de laag waar de informatie van je database in opgeslagen word en gebruik van word gemaakt in je applicatie.

Je **Controller** stuurt de gebruikers naar de juiste Views.

## Repository Pattern

Het repository pattern word gebruikt voor het communiceren met de database.

Het voordeel hier aan is dat er voor alle verschillende functies van je applicatie verschillende repositories komen, hier door blijft je code netjes gescheiden en overzichtelijk.

Ook door middel van verschillende Contexten te maken gebaseerd op verschillende soorten databases is het heel makkelijk om een andere database onder je applicatie te schuiven.

Door dat bijna al de code verwerkt word in de repositories word dit grotendeels je business laag en is het heel makkelijk om te wisselen van UI en Database zonder dat er te veel aan de code veranderd hoeft te worden.

Je UI, Business en Database laag komen dus zo goed als los van elkaar te staan.

# Infrastructuur

De infrastructuur wat de website op zal komen te draaien is al eerder gemaakt bij INS2.

Het gaat hier om een IIS server waar ook de MSSQL database op gehost word.

De website zal vanuit de juiste URL benaderbaar moeten zijn.

De infrastructuur die gebruikt gaat worden zal geïmplementeerd worden op het infralab.

Hier zal een database server gehost worden voor de database van de website, en een webserver voor het hosten van de website.

Dit zullen een MSSQL Server zijn en een IIS server.

Deze worden allemaal geconfigureerd precies goed voor de applicatie.