Ege Saraydar 547873 Docent: Jeroen Claassens

ET.saraydar@student.han.nl

Dit document bevat inhoudelijk technische aspecten over de applicatie VodaGone

VOdagone

Programmeeropdracht DEA Course

Inhoud

[1. Inleiding 2](#_Toc469002967)

[2. Functionele requirements 3](#_Toc469002968)

[3. Niet functionele requirements 4](#_Toc469002969)

[2.1. Testbaarheid 4](#_Toc469002970)

[2.2. Technische beperkingen 4](#_Toc469002971)

[2.3. Interoperabiliteit 5](#_Toc469002972)

[2.4. Onderhoudbaarheid 5](#_Toc469002973)

[2.5 Interface 6](#_Toc469002974)

[2.6. Implementatie 6](#_Toc469002975)

[2.7. Design 7](#_Toc469002976)

[4. Package diagram 7](#_Toc469002977)

[5. Deployment diagram 8](#_Toc469002978)

[6. User Interface 8](#_Toc469002979)

# 1. Inleiding

In dit technische ontwerp zal Vodagone beschreven worden op grond van technische implementaties en eisen die eraan vast gekoppeld zijn. Vodagone is een gebruiksvriendelijke applicatie waar men zijn/haar abonnementen kan beheren, bekijken, delen of uitproberen als het ware.

# 2. Functionele requirements

In dit hoofdstuk worden de functionele eisen behandeld. Om aan te tonen dat er weldegelijk voldaan is aan deze eisen, zijn de tabellen voorzien van commentaar dat refereert naar de inhoudelijke code.

De verwijzing staat tussen blokhaken [] genoteerd.

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| F1 | Toon abonnementen | De abonnee heeft de mogelijkheid om vanaf het abonnementenoverzicht een of meerdere nieuwe diensten aan te vragen.  *[Dit is te zien in Abonnementen.jsp]* | Ja |
| F2 | Dienst uitproberen | De abonnee kan kiezen om een dienst uit te proberen.  *[Dit is te zien in Abonnementen.jsp de knop dienst uitproberen & DienstUitproberen.jsp]* | Ja |
| F3 | Dienst opzeggen | De abonnee kan kiezen om een dienst op te zeggen.  *[Dit is te zien in Abonnementen.jsp de knop opzeggen]* | Ja |
| F4 | Dienst upgraden | De abonnee kan kiezen om een dienst up te graden  *[Dit is te zien in Abonnementen.jsp de knop upgraden]* | Ja |
| F5 | Dienst delen | De abonnee kan zijn dienst delen met een andere abonnee.  *[Dit is te zien in Abonnementen.jsp de knop dienst delen & DienstDelen.jsp]* | Ja |
| F6 | Zoeken dienst | De abonnee kan zoeken naar een dienst om hier meer informatie over te krijgen. *[Dit is te zien in DienstUitproberen.jsp het zoekveldje]* | Ja |

# 3. Niet functionele requirements

In dit hoofdstuk worden de niet-functionele eisen behandeld. Om aan te tonen dat er weldegelijk voldaan is aan deze eisen, zijn de tabellen voorzien van commentaar dat refereert naar de inhoudelijke code. De verwijzing staat tussen blokhaken [] genoteerd.

## 2.1. Testbaarheid

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| T1 | De communicatielaag en databasetoegang moeten los van elkaar testbaar zijn. *[De test die is gemaakt gaat over de datasource layer. Deze test is terug te vinden in het testmapje onder AbonnementDAOIT]* | Zie IM2. Toon dit aan middels minimaal een unit-test. | Ja |

## 2.2. Technische beperkingen

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| TC1 | De applicatie moet draaien in Tomcat v8. | Een andere applicatieserver is een mogelijk onderzoeksonderwerp.  *[Run de applicatie in intelliJ op tomcat]* | Ja |

## 2.3. Interoperabiliteit

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| IO1 | De front-end en back-end van de applicatie moeten samen in 1 container kunnen draaien zodat ze lokaal met elkaar kunnen communiceren. | JSPs en Servlets moeten services en/of DAO's kunnen aanroepen via directe methodeaanroepen.  *[Aangezien we met de layer structuur hebbe gewerkt is dit mogelijk, zie de packages in het project]* | Ja |
| IO2 | De front-end en back-end van de applicatie moeten elk in een eigen container kunnen draaien zodat ze via een protocol met elkaar kunnen communiceren. | De servicelaag is Restful waardoor een nieuw JavaScript of App-frontend deze zou kunnen aanroepen.  Je hoeft dus geen JSP/Servlet oplossing via remoting met de servicelaag te koppelen, dat mag lokaal. Je hoeft geen REST-communicatie tussen front- en back-end mogelijk te maken, dit is een optie voor de onderzoeksopdracht.  *[De servicelaag is volledig gerealiseerd, dit houdt ook in dat de REST aanwezig is en apart ge-runt kan worden.]* | Ja |

## 2.4. Onderhoudbaarheid

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| M1 | Views (JSP-bestanden) bevatten geen Java-code, alleen JSP of JSTL markup. | Scheiding van view, controller en model. *[Er zit wel java code in een aantal .jsp bestanden]* | Nee/Ja |
| M2 | PageControllers (Servlets) bevatten geen markup, alleen Java-code. | Scheiding van view, controller en model. *[Nee, de controllers bevatten geen markup, dit is te zien in de controllerbestanden]* | Ja |

## 2.5 Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| IN1 | De front-end en back-end moeten Restful (JSON over HTTP) kunnen communiceren. | De servicelaag is Restful waardoor een nieuw JavaScript of App-frontend deze zou kunnen aanroepen.  Je hoeft dus geen JSP/Servlet oplossing via remoting met de servicelaag te koppelen, dat mag lokaal.  Je hoeft geen REST-communicatie tussen front- en back-end mogelijk te maken, dit is een optie voor de onderzoeksopdracht. *[De front en back-end zijn restful, er kan dus gecommuniceerd worden over JSON en http, dit gebeurt lokaal. Dit is terug te vinden in het mapje services]* | Ja |

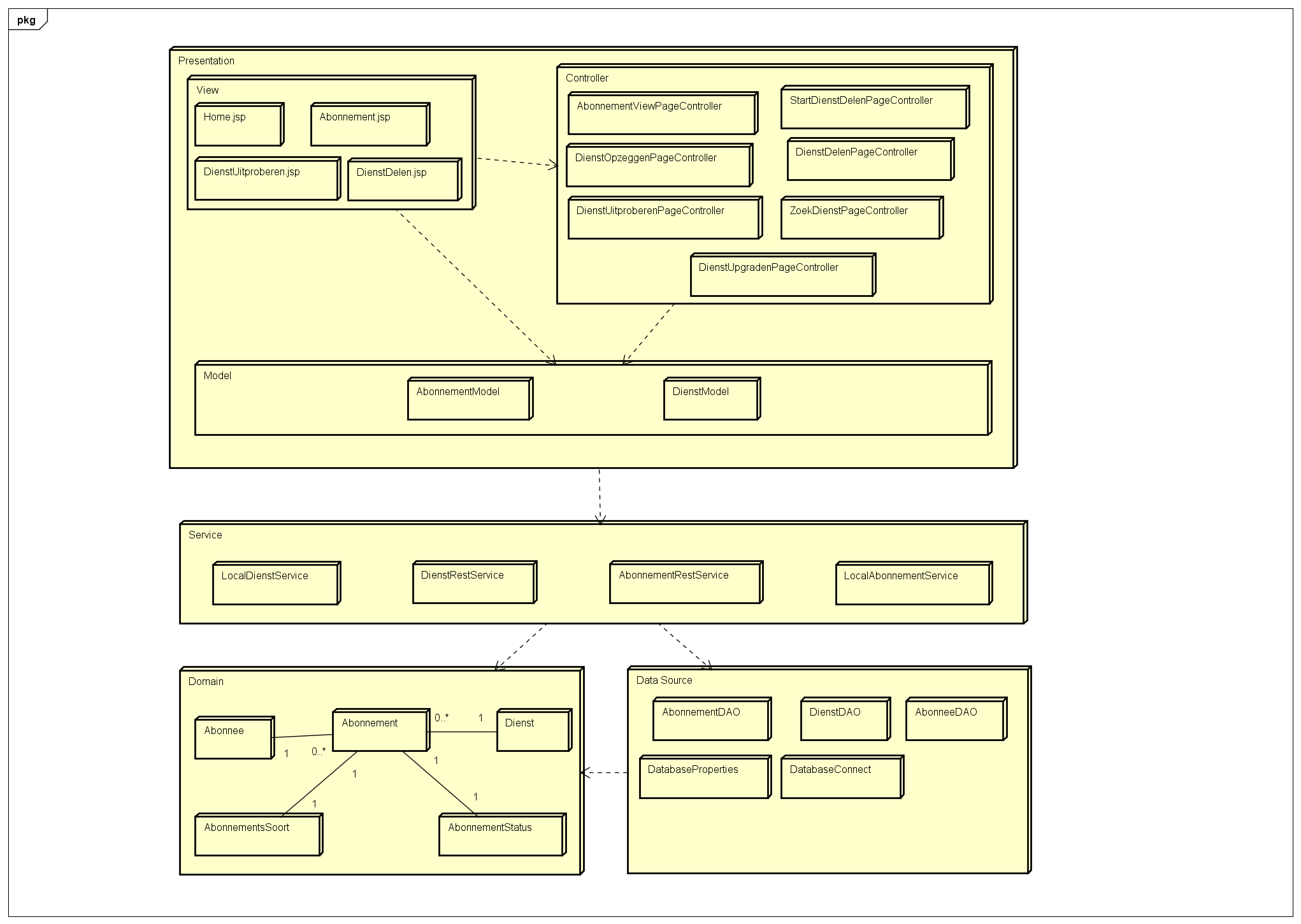
## 2.6. Implementatie

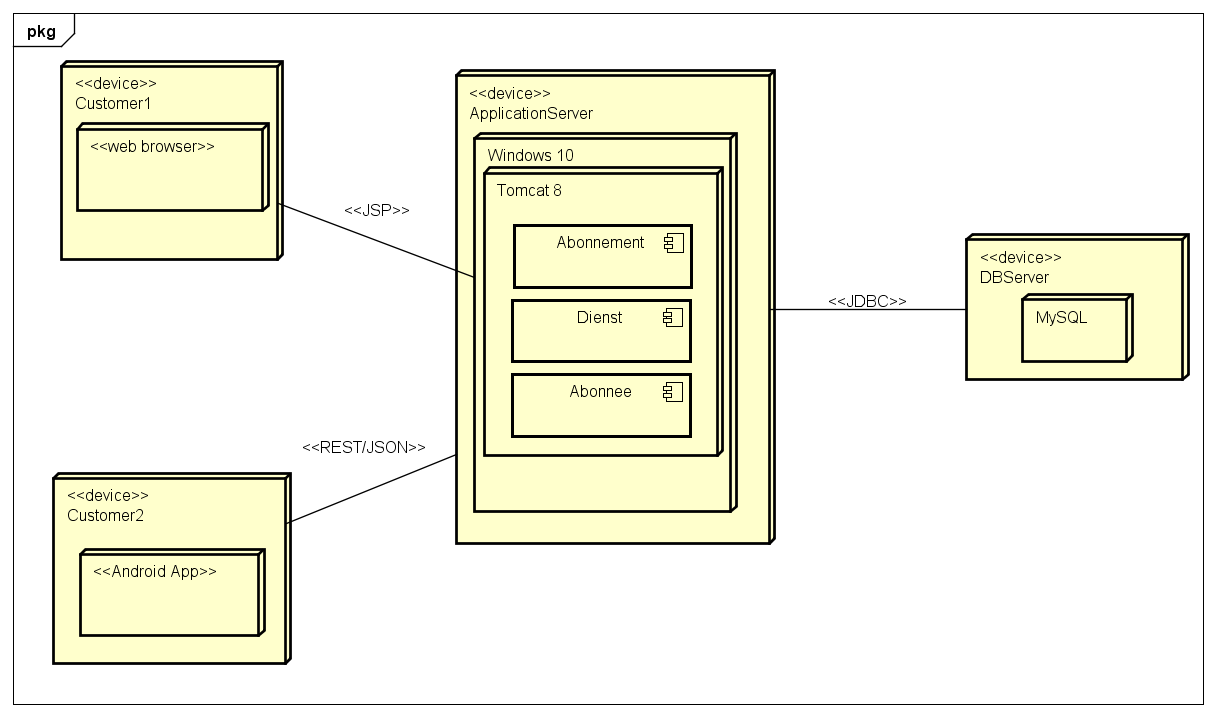
|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| IM1 | De applicatie moet naast MySQL5.1 ook minimaal 1 andere relationele database kunnen ondersteunen. | Implementeer minimaal MySQL en zet je code zo op dat je snel kunt wisselen naar een andere relationele database. *[Dit is heel simplistisch opgelost door een database.properties bestand te maken waar alle gegevens te vinden zijn over de database, loskoppeling is dus mogelijk.* | Ja |
| IM2 | De applicatie maakt gebruik van de volgende APIs en frameworks:  JSP  Servlet v3.0  JAX-RS v2.0 (REST, JSON)  Jersey v2.17 (REST)  Guice v4.0 (Dependency injection)  JDBC  JDBC driver v5.1.34 voor MySQL | Architectuurlagen zijn van elkaar gekoppeld op basis van interfaces. Creatie van classes (m.u.v. domein-klassen) uit deze architectuurlagen vindt zoveel mogelijk plaats op basis van het DIP-principe (technisch geïmplementeerd met Guice).  *[Er is gebruik gemaakt van Guice, te zien aan de @Inject notaties en terug te vinden in mapje config]* | Ja |

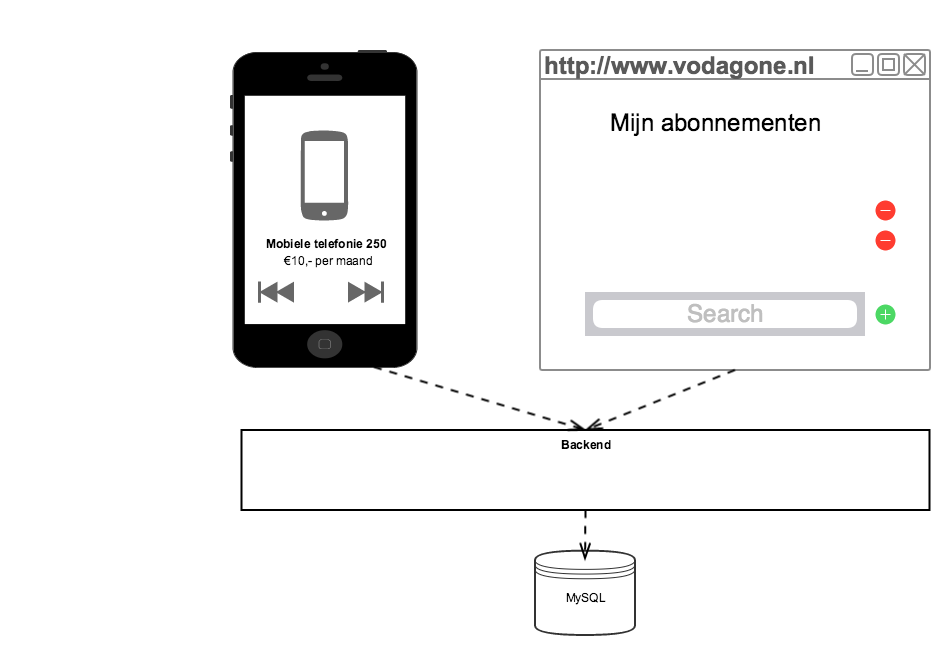
## 2.7. Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Description** | **Explanation** | **Voldaan** |
| D1 | Bij het wisselen van database moet de code niet opnieuw gecompileerd hoeven te worden. | Databaseconfiguratie opslaan in een properties of XML bestand. *[Database.properties file staat in het project en is dus mogelijk]* | Ja |
| D2 | De applicatie moet eenvoudig kunnen wisselen van een relationele (RDBMS) opslag naar een andere opslag zoals NoSQL of flat files. | Pas het Table Data Gateway pattern toe (ofwel  Data Acces Object, DAO) *[De DAO pattern is toegepast, dit is te zien aan de naamgeving van de classes in mapje domain]* | Ja |
| D3 | De broncode is georganiseerd op functionaliteit, niet op technische rol. | Packages zijn dus primair ingedeeld op functie dan pas op architectuurlaag. *[Alles is op volgorde ingedeeld]* | Ja |

# 4. Package diagram

Dit is het diagram dat aantoont wat de structuur van de applicatie is. Hierin valt duidelijk te zien welke layers er zijn gebruikt en hoe ze met elkaar communiceren.

5. Deployment diagram  
In dit hoofdstuk wordt er naar de technische facetten gekeken. Hieruit kan de programmeur zien uit welke componenten het systeem bestaat.

6. User Interface  
In onderstaand diagram wordt het hoofdscherm globaal weergegeven samen met een later te ontwikkelen app. Beiden maken gebruik van een REST-ful API die door de backend geboden wordt.