Vrije Universiteit Brussel

FACULTEIT INGENIEURSWETENSCHAPPEN &
WETENSCHAPPEN
DEPARTMENT OF COMPUTER SCIENCE AND APPLIED COMPUTER
SCIENCE

Software Design Description

Software Engineering

Nicolas Carraggi, Youri Coppens, Christophe Gaethofs, Pieter Meiresone, Sam Van den Vonder, Fernando Suarez, Tim Witters

Academiejaar 2013-2014

Versiegeschiedenis

Tabel 1: Versiegeschiedenis

Versie	Datum	Auteur(s)	Commentaar
0.1	17/11/2013	Youri Coppens	Initiële versie
0.2	10/12/2013	Youri Coppens	Opstart inhoud document
1.0	13/12/2013	Youri Coppens	Oplevering Iteratie 1
2.0	4/03/2014	Youri Coppens	Oplevering Iteratie 2

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Hoofdstuk 1

Introductie

1.1 Doel en scope

Dit document beschrijft de softwarearchitectuur en het design van de CalZone webapplicatie. Het zal gebruikt worden door de programmeurs, de designers en de testers van dit project als naslagwerk en documentatie om de werking van het systeem te vatten. Hierdoor kan dit document gebruikt worden voor uitbreidingen en aanpassingen van het systeem mogelijk te maken.

1.2 Acroniemen

Tabel 1.1: Acroniemen

API	Application Programming Interface
DAO	Data Access Object
DB	Database
EE	Enterprise Edition
GUI	Graphical User Interface
HTML	HyperText Markup Language
JSP	Java Server Page
IDE	Integrated Development Environment
MVC	Model View Controller
SDD	Software Design Description
SDK	Software Development Kit
SRS	Software Requirements Specification
XML	Extensible Markup Language

1.3 Overzicht

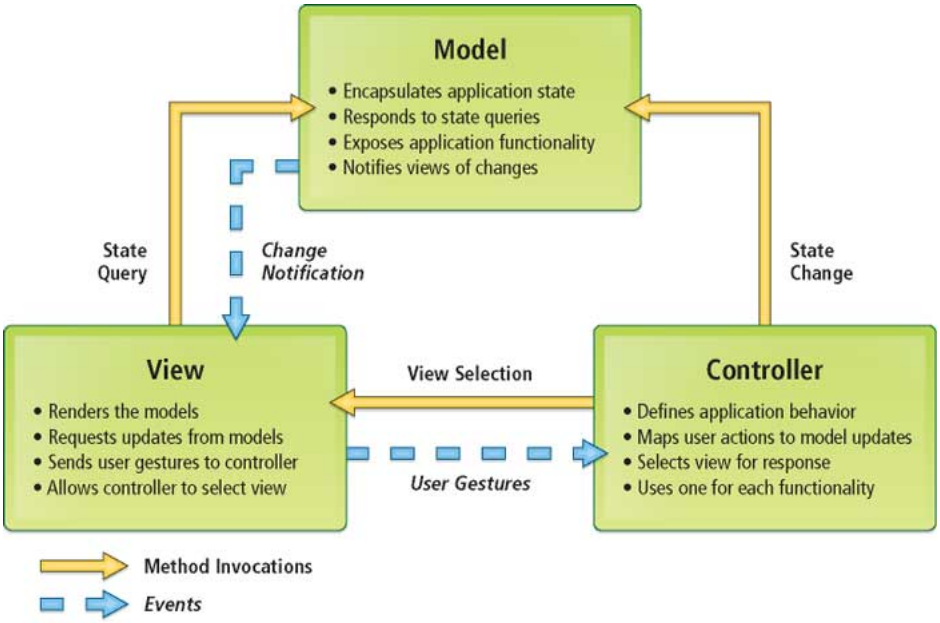
Dit document volgt de IEEE Std 1016-2006^{TM} standaard voor het opstellen van Software Design Descriptions. Dit document is beïnvloed door de requirements beschreven in de Software Requirements Specification van dit project.[3]
In de huidige fase van dit document wordt het design van het systeem in de eerste iteratie beschreven. In hoofdstuk 2 wordt de gebruikte systeemarchitectuur toegelicht en in hoofdstuk 3 worden verschillende 'design viewpoints' besproken die relevant zijn voor het systeem.

Hoofdstuk 2

Systeemarchitectuur

2.1 Model

CalZone is een webapplicatie. Gebruikers van het systeem bezoeken de applicatie via hun webbrowser. Deze browser kan de browser op hun computer zijn of op de Android browser op hun smartphone. CalZone heeft als architectuur gekozen voor het MVC-patroon.[6]



Figuur 2.1: Een voorstelling van de werking van een systeem gebruikmakende van een MVC architectuur

2.2 Gebruikte technologie

De programmeertaal die gebruikt wordt, is Java. In Java wordt gebruik gemaakt van het Spring MVC framework[5, 4] voor het ontwikkelen van CalZone. De IDE waarin geprogrammeerd wordt, is de meest recente versie van 'Eclipse Classic' met volgende uitbreidingen:

- De gehele collectie 'Web, XML, Java EE and OSGi Enterprise Development'
- Spring Tool Suite (uit de Eclipse Marketplace)

- De gehele collectie 'Maven Integration for Eclipse'

Het uitvoeren van de applicatie wordt mogelijk gemaakt door middel van Apache Maven[1] en Apache Tomcat[2]. De gebruikte databank voor de back-end is MySQL. Voor de view wordt gebruik gemaakt van JSP's: een tehnologie om dynamisch webpagina's te genereren.

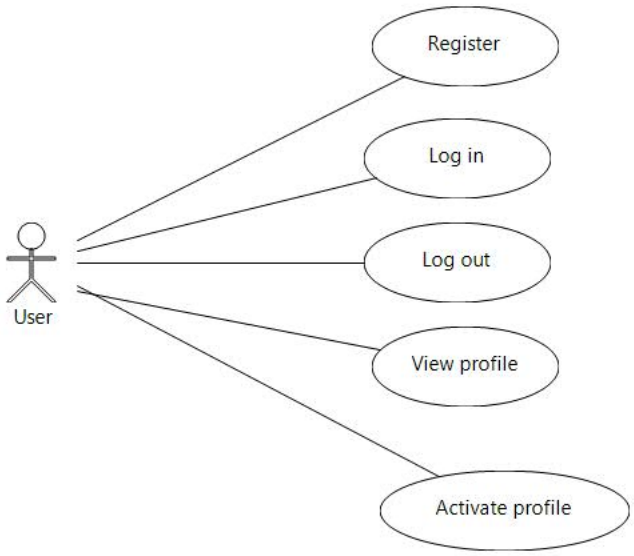
Hoofdstuk 3

Viewpoints

In dit hoofdstuk worden *design viewpoints* besproken die relevant zijn voor het systeem. Naarmate de ontwikkelingen van CalZone vorderen, zullen deze viewpoints uitgebreid worden en eventueel aangepast worden. Op het einde van de laatste iteratie wordt verwacht dat alle requirements uit het SRS van dit project ontworpen zijn. Voorlopig wordt er in de volgende secties enkel het design besproken van de eerste en tweede iteratie.

3.1 Context

De gebruikers van het systeem zijn onder te verdelen in 5 categoriën: externen, studenten, professoren, assistenten en programmabeheerders. Elk soort gebruiker moet in de finale versie van CalZone in staat zijn de functionaliteiten die specifiek aan deze gebruikers zijn toegekend toe te passen zoals beschreven in het SRS van dit project.
In de huidige fase van het project is er nog geen onderscheid te merken in de verschillende soorten gebruikers naar de buitenwereld toe, hoewel er in de databank wel reeds rekening mee is gehouden (zie sectie 3.3). Daarom wordt er voortaan in deze tekst enkel over gebruikers in zijn meest algemene vorm gesproken.
Gebruikers zijn in staat om zich te registreren in het systeem. Hierdoor kunnen ze inloggen en bezitten deze gebruikers een profielpagina. Eenmaal ingelogd



Figuur 3.1: Use case diagram

3.2 Logica

In deze sectie worden de verschillende modules en packages behandeld die tesamen het huidige systeem vormen.

3.2.1 Controllers

Deze klassen zijn verantwoordelijk om de HTTP-requests te verwerken. Deze klassen zorgen dus voor een propagatie van (functionaliteits)verzoeken van de front-end naar de back-end. Eveneens zorgen deze klassen ook voor een propagatie van data van de back-end naar de front-end. Dit laatste uit zich in het voorzien van webpagina's. Met andere woorden zorgen de controllers dus voor de mogelijke views en voorzien dus de mogelijkheid om de functionaliteiten opgesomd in het use case diagram van sectie 3.1 uit te voeren. Voor volgende concepten zijn er nu controllers voorzien:

- Login
- Profieloverzicht
- Registratie
- Accountactivatie

3.2.2 Databaseklassen

CalZone maakt gebruik van een relationele databank, MySQL, als back-end voor dataopslag. Om informatie vanuit deze databank te lezen en er naartoe te schrijven zijn er enkele klassen voorzien die een abstractie bieden voor het openen en sluiten van de connectie met de databank en het sturen van queries naar de databank. Volgende klassen zijn hiervoor voorzien:

- DbConfig: Deze klasse voorziet de mogelijkheid om gegevens op te halen om toegang te kunnen krijgen tot een bepaalde databank. Deze gegevens de gebruikersnaam, het wachtwoord en de locatie van de databank.

- DbLink: Het openen en sluiten van de databankconnecties samen met het sturen van queries en het ontvangen van resultaten.
- DbTranslate: Een collectie van methodes die gemapt worden op queries

3.2.3 Data Access Objects

Het ophalen van data uit de MySQL databank gebeurt via *Data Access Objects* of kortweg DAO's. Deze objecten gebruiken de algemene databasemethoden voorzien door de databaseklassen aangehaald in subsectie 3.2.2 om specifieke data op te halen uit de databank en in te laden in de beschikbare klasses in het systeem. Ook wordt specifieke informatie via deze DAO's weggeschreven naar de databank. Volgende DAO's zijn aangemaakt in deze iteratie:

- ActivationKeyDao: de activatiesleutels gebonden aan een geregisteerd account.
- UserDao: gebruikergegevens
- SessionDao: gebruikerssessies met het systeem.

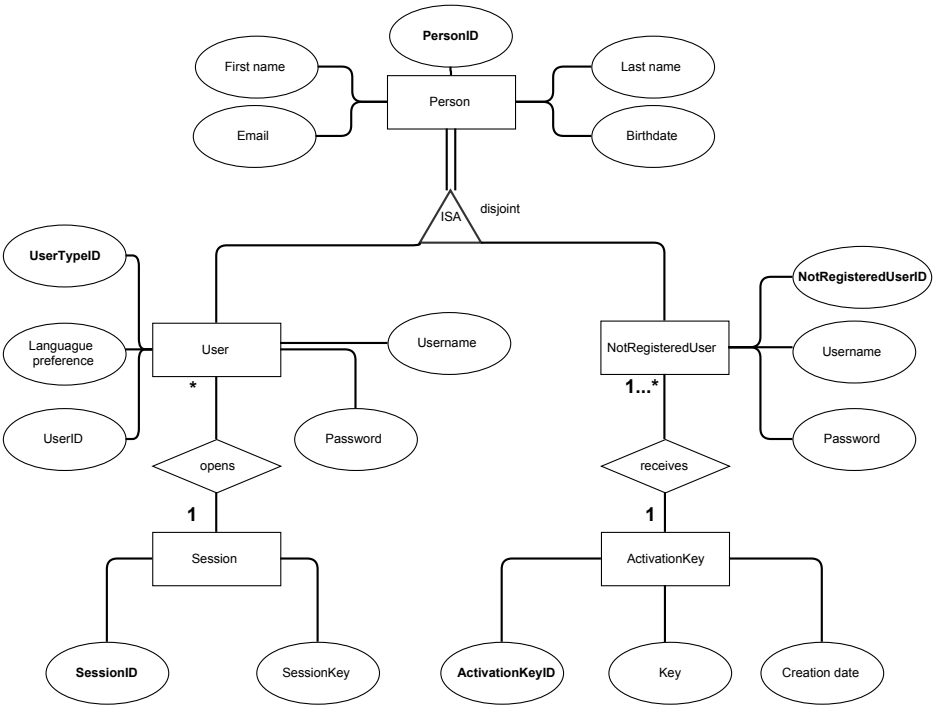
3.2.4 Validators

Binnenin het systeem dient sommige data gecontroleerd te worden op geldigheid. Deze validatorklassen zijn verantwoordelijk om geldigheid van bepaalde informatie te controleren en aan te geven indien deze info niet geldig is. Volgende validators zijn in het huidige systeem aanwezig:

- Email: controle op het emailadres
- Gebruikers: controle op de gebruikersnaam om profiel

3.3 Data

De evolutie die het design in deze iteratie heeft ondervonden is vooral te merken in de database. Het datamodel De huidige databank is ontworpen om gebruikers, die zich willen registreren in het systeem, op te slaan. Men maakt een onderscheid tussen gebruikers die al dan niet hun account geactiveerd hebben. Een gebruiker die niet geactiveerd is, bezit een lijst van activatiesleutels. Dit is een lijst omdat activatiesleutels slechts tijdelijk geldig zijn en gebruikers in staat moeten kunnen zijn om een nieuwe activatiesleutel aan te vragen indien ze hun account als dan nog willen activeren nadat hun huidige activatiesleutel vervallen is. Geactiveerde gebruikers kunnen inloggen. Hierdoor wordt een sessie aangemaakt in het systeem. Deze sessies worden gelogd in de databank. Figuur 3.2 toont het EER-model van de databank.



Figuur 3.2: EER diagram

Bibliografie

[1] The Apache Software Foundation. *Apache Maven 3.x*. Versie 3.1.1. 2013. URL: https://maven.apache.org/ref/3.1.1/.

[2] The Apache Software Foundation. *Apache Tomcat 7 Documentation*. Versie 7.047. 2013. URL: https://tomcat.apache.org/tomcat-7.0-doc/index.html.

[3] Fernando Suarez Groen, Tim Witters en Youri Coppens. *CalZone: Software Requirements Specification*. Versie 1.0. 2013.

[4] Rod Johnson e.a. *Spring Framework Reference Documentation*. Versie 3.2.5. 2013. URL: http://docs.spring.io/spring/docs/3.2.5.RELEASE/spring-framework-reference/htmlsingle/.

[5] Pivotal Software. *Spring*. 2013. URL: http://spring.io/.

[6] Onbekende auteurs. *Model–view–controller*. 2013. URL: https://en.wikipedia.org/wiki/Modelviewcontroller.