



FACULTEIT INGENIEURSWETENSCHAPPEN EN  
ARCHITECTUUR

Bachelor of Science Industrial Science : Informatics  
Academic year 2013-2014

---

# Vakoverschrijdend Eindproject Team 12

---

Stambomen

Submitted on 19 May 2014

*Students:*

Kenzo Clauw

Axl François

Lowie Huyghe

Sander Trypsteen

Jelle Verreth



Master of Science Industrial Science : Informatics  
Academic year 2013-2014

---

# **Vakoverschrijdend Eindproject Team 12**

---

Stambomen

Submitted on 19 May 2014

*Student:*

Kenzo Clauw

Axl François

Lowie Huyghe

Sander Trypsteen

Jelle Verreth

## **Abstract**

Iedereen heeft zicht ooit al afgevraagd waar hij of zij vandaan kwam. Het is niet altijd even makkelijk om een antwoord op deze vraag te formuleren. Enerzijds omdat dit een levensvraag is anderzijds omdat het niet altijd alles correct werd bijgehouden doorheen de geschiedenis. Met ons project proberen we exact dit probleem op te lossen. We willen mensen de kans bieden om op een eenvoudige manier, hun stamboom bij te houden. Op hetzelfde moment proberen te genieten van de interconnectiviteit die vandaag de dag bestaan dankzij sociale media. Zo wordt het opbouwen van een stamboom, een echte sociale gebeurtenis. Om dit project te realiseren hebben wij gebruik gemaakt van Scrum, een agile-softwareontwikkelings-techniek. Hierdoor werd ons project opgedeeld in 3 sprints waarbij we na elke sprint een deel van het gehele programma gemaakt hadden. Het resultaat is een prachtige applicatie, gebouwd in Java. De applicatie bestaat uit een manipulatie deel, in de vorm van een desktop applicatie en een website. Er is integratie met de sociale media site Facebook. Mensen kunnen bomen delen, personen linken met Facebook accounts en vrienden van op Facebook toevoegen aan de vrienden binnen onze applicatie. Wij hopen met deze applicatie de aanzet te geven tot een nieuwe generatie van stamboom-software. De toekomst van stambomen ligt overduidelijk bij integratie van sociale media en een toegankelijke, gebruiksvriendelijke user-interface.

## **Abstract**

Where do you come from? Who were your parents? Who were the parents of your parents? It isn't always easy to formulate an answer to these questions. Not only because we are dealing with a life-question but also because not everything has been tracked perfectly throughout history. With our project we are trying to solve these issues. We want to give people the opportunity to keep up with their pedigree in a simple user-friendly way. At the same time we are piggybacking on the success of social media. Building a pedigree becomes a social event. To accomplish this project we have used Scrum which is a part of agile-development. Our project was split into three sprints. Each sprint we would implement a portion of the application. The result is a beautiful application built in Java. The application consists of a manipulation part, where user can add trees, persons. We also have a website where a user can consult their pedigrees. There is integration with social media. People can share their tree on Facebook or link persons with Facebook-accounts. We hope that with this application, we have shaped the future of all new pedigree-software. Because the future for pedigrees lies in social-integration and user-friendly interface.

# Contents

<b>1</b>	<b>Voorwoord</b>	<b>5</b>
<b>2</b>	<b>Introductie en situering</b>	<b>6</b>
2.1	Introductie . . . . .	6
2.2	Technische . . . . .	7
<b>3</b>	<b>Taak-verdeling</b>	<b>8</b>
3.1	Methodologie . . . . .	8
3.2	Sprint 1 . . . . .	9
3.2.1	Kenzo Clauw . . . . .	9
3.2.2	Axl François . . . . .	9
3.2.3	Lowie Huyghe . . . . .	9
3.2.4	Sander Trypsteen . . . . .	10
3.2.5	Jelle Verreth . . . . .	10
3.2.6	Niet opgenomen . . . . .	10
3.3	Sprint 2 . . . . .	11
3.3.1	Kenzo Clauw . . . . .	11
3.3.2	Axl François . . . . .	11
3.3.3	Lowie Huyghe . . . . .	12
3.3.4	Sander Trypsteen . . . . .	12
3.3.5	Jelle Verreth . . . . .	13
3.3.6	Niet opgenomen . . . . .	13
3.4	Sprint 3 . . . . .	14
3.4.1	Kenzo Clauw . . . . .	14
3.4.2	Axl François . . . . .	14
3.4.3	Lowie Huyghe . . . . .	14
3.4.4	Sander Trypsteen . . . . .	14
3.4.5	Niet-gerealiseerd . . . . .	14
3.5	Overzicht . . . . .	15

<b>4</b>	<b>Analyse</b>	<b>18</b>
4.1	DB-structuur . . . . .	18
4.2	Deployment diagram . . . . .	20
4.3	DCD Service . . . . .	22
4.4	DCD User . . . . .	25
4.5	DCD Person . . . . .	27
4.6	Aanmaken van gebruiker . . . . .	28
4.6.1	Use case . . . . .	28
4.6.2	Sequence Diagram Aanmaken van gebruiker . . . . .	29
4.7	Inloggen . . . . .	30
4.7.1	Use case . . . . .	30
4.7.2	Sequence Diagram Inloggen . . . . .	31
4.8	Aanmaken stamboom . . . . .	32
4.8.1	Use case . . . . .	32
4.8.2	Sequence Diagram Aanmaken stamboom . . . . .	33
4.9	Toevoegen persoon . . . . .	34
4.10	Use case . . . . .	34
4.10.1	Sequence Diagram Toevoegen persoon . . . . .	35
4.11	Versturen Van Vriendschapsverzoek . . . . .	36
4.11.1	Use case Versturen Van Vriendschapsverzoek . . . . .	36
4.11.2	Sequence Diagram Versturen Van Vriendschapsverzoek . . . . .	37
<b>5</b>	<b>Ontwerpbeslissingen</b>	<b>38</b>
5.1	Best Practice . . . . .	38
5.1.1	JSP vs Servlets . . . . .	38
<b>6</b>	<b>Implementatiekeuzes</b>	<b>39</b>
6.1	Bibliotheken . . . . .	39
6.1.1	Jersey . . . . .	39
6.1.2	Swagger Jersey . . . . .	39
6.1.3	SLF4J . . . . .	40
6.1.4	Junit 4.x . . . . .	40
6.1.5	Gedcom . . . . .	46
6.1.6	Gedcom4j . . . . .	47
6.1.7	Sardine . . . . .	48
6.1.8	RestFB . . . . .	50
6.1.9	JCalendar . . . . .	52
6.1.10	Abego TreeLayout . . . . .	53
6.1.11	Java FX . . . . .	54
6.1.12	JQuery . . . . .	54
6.1.13	JQuery UI . . . . .	54

---

6.1.14 JSTL . . . . .	55
<b>7 Kwaliteitscontrole</b>	<b>56</b>
<b>8 Gekende problemen</b>	<b>57</b>
8.1 WebApp . . . . .	57
8.2 DesktopApp . . . . .	58
<b>9 Verbeterpunten en uitbreidingsmogelijkheden</b>	<b>59</b>
9.1 Gedcom . . . . .	59
9.2 Stambomen . . . . .	60
9.3 Admin . . . . .	60
<b>10 Nabeschouwing en besluit</b>	<b>62</b>
 <b>List of Figures</b>	 <b>64</b>
 <b>List of Tables</b>	 <b>65</b>
 <b>A Appendix</b>	 <b>66</b>
A.1 Opdracht sprint 1 . . . . .	67
A.2 Opdracht sprint 1 . . . . .	68
A.3 Opdracht sprint 2 . . . . .	69
A.4 Opdracht sprint 2 . . . . .	70



# Chapter 1

## Voorwoord

Dit projectdossier is een schriftelijk neerslag van onze prestaties voor het vakoverschrijdend project<sup>1</sup>. Hierin bespreken we de verschillende facetten van ons project. Allereerst zullen wij beginnen met de opdracht voor te stellen en ze te situeren. Hierbij zullen we proberen van een zo volledig mogelijk overzicht te geven van alle technische en niet-technische onderdelen. Vervolgens krijgt u meer uitleg over de manier waarop onze takenverdeling tot stand komt en het volledig overzicht van onze takenverdeling. Daarna bespreken we de analyse van het project aan de hand van enkele diagrammen en use cases. Vervolgens lichten we enkele belangrijke aspecten van ons project toe. We gaan hier zowel in op ontwerpbeslissingen als op implementatiekeuzes. Dan komen we aan de kwaliteitscontrole voldoet onze applicatie aan de functionele en niet-functionele vereisten zoals deze zijn opgesteld door de klant? Uiteraard zijn er in elk project wel problemen en we zullen het niet na laten om deze te bespreken. Als voorlaatste geven we nog even onze visie op het verdere verloop van de applicatie. Waar kan het beter, welke uitbreidingen zijn mogelijk. Ten slotte eindigen we met een nabeschuiving en ons besluit over het VOP.

---

<sup>1</sup>VOP

# Chapter 2

## Introductie en situering

### 2.1 Introductie

In deze paper kan u meer informatie vinden rond ons VOP. Het onderwerp zoals u wellicht al gelezen hebt op het titel blad zijn stambomen. Het woord stambomen kan in verschillende contexten bekeken worden. We beperken ons echter enkel tot de genealogie. In genealogie zitten twee Griekse woorden verborgen namelijk Genea en Logos. Genea staat voor afkomst of afstamming en logos betekent wetenschap of kennis. Dus we bestuderen hier de afkomst, afstamming van een persoon. In dit project zijn er echter twee beperkingen:

Zonder dieper in te gaan op het onderwerp genealogie willen ook nog melden dat in dit project we ons slechts zullen beperken tot natuurlijk mogelijke afstammelingen. Hieronder verstaan wij dat er enkel man - vrouw relaties mogelijk zijn. We houden dus geen rekening met man - man of vrouw - vrouw relaties die dan kinderen zouden hebben.

Verder houden we ook geen rekening met de onderlinge relaties tussen echt-paren. Op zich zijn dat onze zaken niet maar wanneer een persoon een buitenechtelijke kind heeft dan zal hij dat niet kunnen invoeren in ons programma. Hieronder vallen dus ook half-broers of half-zussen. Deze zullen niet weergegeven worden in ons stamboom overzicht.

Nu de scope van de opdracht duidelijker is kunnen ingaan op wat er gevraagd is. Het is de bedoeling om een applicatie te maken waar een gebruiker stambomen kan ingeven en bekijken. De gebruiker moet deze stambomen kunnen raadplegen via een website en de bomen worden ingegeven via een desktop applicatie. Verder moet de gebruiker ook zijn bomen kunnen delen met zijn vrienden zowel binnen de applicatie als op sociale netwerk sites. Voor dit project zullen we ons qua sociale netwerk sites beperken tot Facebook.

## 2.2 Technische

We hebben op 10/02/2014 de inleidende presentatie van het VOP gekregen. Hierin werden een aantal vereisten uitgelegd. Deze vereisten staan uitgebreid beschreven in VOP: de richtlijnen <sup>1</sup>. De belangrijkste hierin zijn dat er een docent de rol van klant zal spelen. Er zullen afspraken moeten worden gemaakt met de klant over wat al dan niet mogelijk is. Wat er moet gerealiseerd worden? We kwamen ook te weten dat het project in Java moest geschreven worden. Verder moest dit gebeuren door aan de hand van een drielagen architectuur.

**Presentatie laag** Deze laag bestond uit twee onderdelen namelijk een Java Desktop Client gemaakt in Java Swing en een website (HTML5, Javascript) die gebruikt maakt van Java Servlets. De nadruk uiteraard ligt hier bij het scheiden van onze logica van de presentatie.

**Domein logica** Hierbij was het de bedoeling om een REST-service in Java op te zetten waarvan de presentatie gebruik kan maken. We hebben hier van in het begin gekozen om Jersey te gebruiken.

**Data tier** Een relationele databank ging in staan voor de persistentie van onze gegevens. De databank die ons aangeboden werd door UGent is MySQL. We mochten in deze opdracht geen gebruik maken van ORM-tools<sup>2</sup>.

---

<sup>1</sup>ADD REFERENCE

<sup>2</sup>Object-Relational Mapping zoals Hibernate.

# Chapter 3

## Taak-verdeling

### 3.1 Methodologie

Tijdens dit project zullen we gebruik maken van Scrum. Hierbij verdelen we het project in drie sprints. Aan het begin van elke sprint zitten we samen om te bespreken welke features we graag zouden opnemen. Hoe lang een bepaalde feature duurt om uit te werken. Na deze bespreking maken we een afspraak bij de klant.

Met de klant bespreken we dan welke features voor ons mogelijk zijn en welke features voor de klant noodzakelijk zijn. Hieruit volgt een contract, alle features die we tijdens een bepaalde sprint zullen realiseren.

Op het einde van de sprint kijken we dan welke features we gerealiseerd hebben en welke niet. Dankzij deze methode kunnen we inschatten of we tijdens het project moeten bijsturen. Het kan zijn dat we niet alle features gerealiseerd hebben dan moeten we harder werken tijdens de volgende sprint.

Tijdens elke sprint krijgen we van de docenten een voorgestelde lijst met features. Deze lijsten kan je vinden als appendix bij dit document, A.2 en A.4. In het vervolg van dit hoofdstuk kan er dus wel eens verwezen worden naar de nummering die voor de taken gebruikt wordt in deze documenten.

## 3.2 Sprint 1

### 3.2.1 Kenzo Clauw

- 22. Het systeem bevat een grote hoeveelheid stamboomgegevens

### 3.2.2 Axl François

- Het aanmaken van de volledige structuur van de applicatie
- 1. De gebruiker maakt een account aan.
- 3. De gebruiker logt in op de desktopapplicatie.
- 4. De gebruiker maakt een nieuwe stamboom aan.
- 5. De gebruiker opent een bestaande stamboom.
- 7. De gebruiker wijzigt de gegevens van een persoon in de huidige stamboom.
- 9. De gebruiker verwijdert een persoon uit de huidige stamboom (Niet afgewerkt)

### 3.2.3 Lowie Huyghe

#### WEB

- 2. De gebruiker logt in op de webapplicatie.
- 14. De gebruiker bekijkt zijn vriendenlijst.
- 11. De gebruiker stuurt een vriendschapsverzoek.
- 12. De gebruiker aanvaardt een vriendschapsverzoek.
- 13. De gebruiker weigert een vriendschapsverzoek.
- 15. De gebruiker verwijdert iemand uit zijn vriendenlijst.

- 16. De gebruiker consulteert zijn samengestelde stamboom.
- 17. De gebruiker bekijkt de gegevens van een persoon in een stamboom.
- 18. De gebruiker wijzigt de referentiepersoon van een stamboom.

### 3.2.4 Sander Trypsteen

#### API

- 14. De gebruiker bekijkt zijn vriendenlijst.
- 11. De gebruiker verstuurt een vriendschapsverzoek.
- 12. De gebruiker aanvaardt een vriendschapsverzoek.
- 13. De gebruiker weigert een vriendschapsverzoek.
- 15. De gebruiker verwijdert iemand uit zijn vriendenlijst.

### 3.2.5 Jelle Verreth

- 4. De gebruiker maakt een nieuwe stamboom aan.
- 6. De gebruiker voegt een persoon toe aan de huidige stamboom (Niet afgewerkt)
- 8. De gebruiker verplaatst een persoon in de huidige stamboom (Niet afgewerkt)

### 3.2.6 Niet opgenomen

- 10. De gebruiker importeert een GEDCOM-bestand.
- 19. De gebruiker kiest de referentiepersoon voor de teletijdmachine.
- 20. De gebruiker kiest de datum voor de teletijdmachine

- 21. De teletijdmachine markeert de geboorteplaatsen van de familieleden die op de ingevoerde datum in leven waren.

## 3.3 Sprint 2

### 3.3.1 Kenzo Clauw

#### Sprint 1

- 10. De gebruiker importeert een GEDCOM-bestand.

#### Sprint 2

- 22. De moderator wijzigt de gegevens van een persoon.
- 23. De moderator wijzigt het profiel van een gebruiker.
- 24. De moderator blokkeert tijdelijk het account van een gebruiker.

### 3.3.2 Axl François

#### Sprint 1

- 6. De gebruiker voegt een persoon toe aan de huidige stamboom
- 8. De gebruiker verplaatst een persoon in de huidige stamboom
- 9. De gebruiker verwijdert een persoon uit de huidige stamboom

#### Desktop + API

- 3. De gebruiker voegt een foto toe aan een persoon.
- 4. De gebruiker verwijdert een foto van een persoon.
- 5. De gebruiker zoomt in op de stamboom. Naarmate meer ruimte beschikbaar is, worden meer gegevens weergegeven bij een persoon

- 15. De gebruiker geeft aan of een stamboom wordt gedeeld met alle gebruikers van de applicatie, enkel met zijn/haar vrienden, of met niemand.
- 16. De gebruiker registreert zich met zijn/haar Facebookaccount.
- 17. De gebruiker koppelt zijn/haar account aan zijn/haar persoonlijke Facebookaccount.
- 18. De gebruiker logt in met Facebook.
- 30. De ontwikkelaar wijzigt de huisstijl van de applicatie.

### 3.3.3 Lowie Huyghe

#### Sprint 1

- 19. De gebruiker kiest de referentiepersoon voor de teletijdmachine.
- 20. De gebruiker kiest de datum voor de teletijdmachine
- 21. De teletijdmachine markeert de geboorteplaatsen van de familieleden die op de ingevoerde datum in leven waren.

#### Sprint 2

- 7. De gebruiker doorzoekt de voor hem beschikbare stambomen
- 25. De gebruiker bekijkt een animatie van de teletijdmachine.
- 26. De gebruiker vertraagt of versnelt de teletijdmachine.
- 27. De gebruiker pauzeert de teletijdmachine.
- 28. De gebruiker navigeert naar een tijdstip met de teletijdmachine
- 29. De gebruiker wijzigt het thema van de applicatie.

### 3.3.4 Sander Trypsteen

- 12. De gebruiker doorzoekt de openbare gebruikerslijst.



- 13. De gebruiker bekijkt een openbaar profiel.
- 14. De gebruiker maakt zijn/haar profiel openbaar.
- 1. De gebruiker wijzigt de weergavetaal (Web).

### 3.3.5 Jelle Verreth

- 1. De gebruiker wijzigt de weergavetaal.
- 2. Het systeem houdt een overzicht bij van relevante gebeurtenissen.
- 21. De gebruiker bekijkt een overzicht van de activiteiten van zijn/haar vrienden

### 3.3.6 Niet opgenomen

- 6. De gebruiker koppelt een persoon aan zijn/haar Facebookaccount.
- 8. De gebruiker geeft aan dat twee personen uit verschillende stambomen aan elkaar gelijk zijn.
- 9. Het systeem detecteert gelijkaardige personen uit verschillende stambomen en meldt de gelijkenissen aan de betrokken gebruikers.
- 10. De gebruiker bevestigt dat twee personen aan elkaar gelijk zijn.
- 11. De gebruiker geeft aan dat twee personen niet aan elkaar gelijk zijn.
- 19. De gebruiker deelt stamboomgegevens op Facebook.
- 20. De gebruiker voegt een Facebookvriend toe aan zijn/haar vriendenlijst.

## 3.4 Sprint 3

### 3.4.1 Kenzo Clauw

- 1. De gebruiker wijzigt de weergavetaal.
- 2. Het systeem houdt een overzicht bij van relevante gebeurtenissen.

### 3.4.2 Axl François

- 6. De gebruiker koppelt een persoon aan zijn/haar Facebookaccount.

### 3.4.3 Lowie Huyghe

- 20. De gebruiker voegt een Facebookvriend toe aan zijn/haar vriendenlijst.

### 3.4.4 Sander Trypsteen

- 19. De gebruiker deelt stamboomgegevens op Facebook.
- 18. De gebruiker logt in met Facebook. (Web)
- 16. De gebruiker registreert met Facebook (Web)

### 3.4.5 Niet-gerealiseerd

- 8. De gebruiker geeft aan dat twee personen uit verschillende stambomen aan elkaar gelijk zijn.
- 9. Het systeem detecteert gelijkaardige personen uit verschillende stambomen en meldt de gelijkenissen aan de betrokken gebruikers.
- 10. De gebruiker bevestigt dat twee personen aan elkaar gelijk zijn.
- 11. De gebruiker geeft aan dat twee personen niet aan elkaar gelijk zijn.

## 3.5 Overzicht

	Kenzo Clauw	Axl Francois	Lowie Huyghe	Sander Trypsteen	Jelle Verreth
SPRINT 1					
1.		S1			
2.			S1		
3.		S1			
4.		S1			S1
5.		S1			
6.		S2			S1 AF → S2
7.		S1			
8.		S2			S1 AF → S2
9.		S1 AF → S2			
10.	OP in S1, S2				
11.			S1	S1	
12.			S1	S1	
13.			S1	S1	
14.			S1	S1	
15.			S1	S1	
16.			S1		
17.			S1		
18.			S1		
19.			OP in S1, S2		
20.			OP in S1, S2		
21.			OP in S1, S2		
22.	S1				

	Kenzo Clauw	Axl Francois	Lowie Huyghe	Sander Trypsteen	Jelle Verreth
SPRINT 2					
1.	S3			S2	S2 AF → S3
2.	S3				S2 AF → S3
3.		S2			
4.		S2			
5.		S2			
6.		OP in S2, S3			
7.			S2		
8.	GER	GER	GER	GER	GER
9.	GER	GER	GER	GER	GER
10.	GER	GER	GER	GER	GER
11.	GER	GER	GER	GER	GER
12.				S2	
13.				S2	
14.				S2	
15.		S2			
16.		S2			
17.		S2			
18.		S2		S3	
19.				OP in S2, S3	
20.			OP in S2, S3		
21.					S2
22.	S2				
23.	S2				
24.	S2				
25.			S2		
26.			S2		
27.			S2		
28.			S2		
29.			S2		
30.		S2			

**legende**

- Sprint 1: S1
- Sprint 2: S2
- Sprint 3: S3

- Sprint Niet opgenomen: OP
- Sprint Niet afgewerkt: AF
- Sprint Niet gerealiseerd: GER

# Chapter 4

## Analyse

### 4.1 DB-structuur

De databasestructuur is opgebouwd rond de user die een tree kan aanmaken die op zich weer personen kan bevatten.

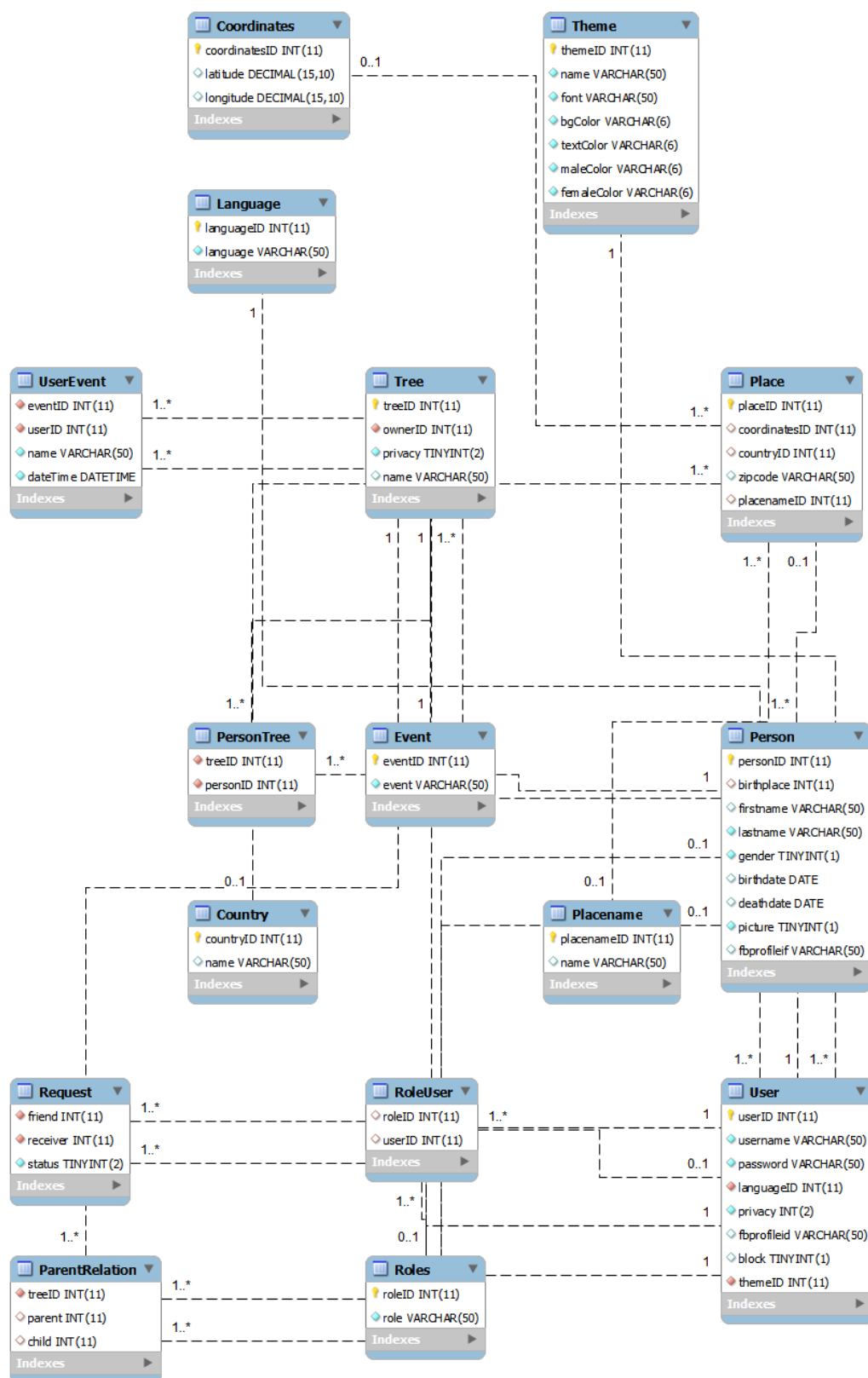
Een user heeft een username, paswoord, heeft een standaard taal, een privacy instelling om aan te geven of zijn profiel publiek is, een standaard thema. De user kan geblokkeerd worden zodat deze niet kan inloggen. Een user kan zich op twee manieren registreren met een facebookprofiel waarvan het facebookprofielid bijgehouden wordt en een gewoon gebruikersaccount. Ook heeft deze een role zodat er gebruikersmachtigingen kunnen toegepast worden. Deze heeft ook een mogelijkheid om vrienden te maken door een friendrequest te sturen. Ook worden Usevents bijgehouden per user.

Een tree is van een user, heeft een naam, een privacy zodat deze gedeeld kan worden met vrienden, of openbaar gemaakt kan worden.

Een persoon is van een tree, heeft een naam, achternaam, geslacht, geboorteplaats, geboortedatum, mogelijks een sterfdatum, een afbeelding en eventueel een facebookprofielid. Deze kan een relatie hebben met een andere persoon die dan zijn ouder of kind kan zijn.

Een plaats heeft een landnaam, zipcode, plaatsnaam, en mogelijks coördinaten.

Naarmate het project evolueerde en er nieuwe features bijkwamen zijn we steeds gestart vanuit de database.



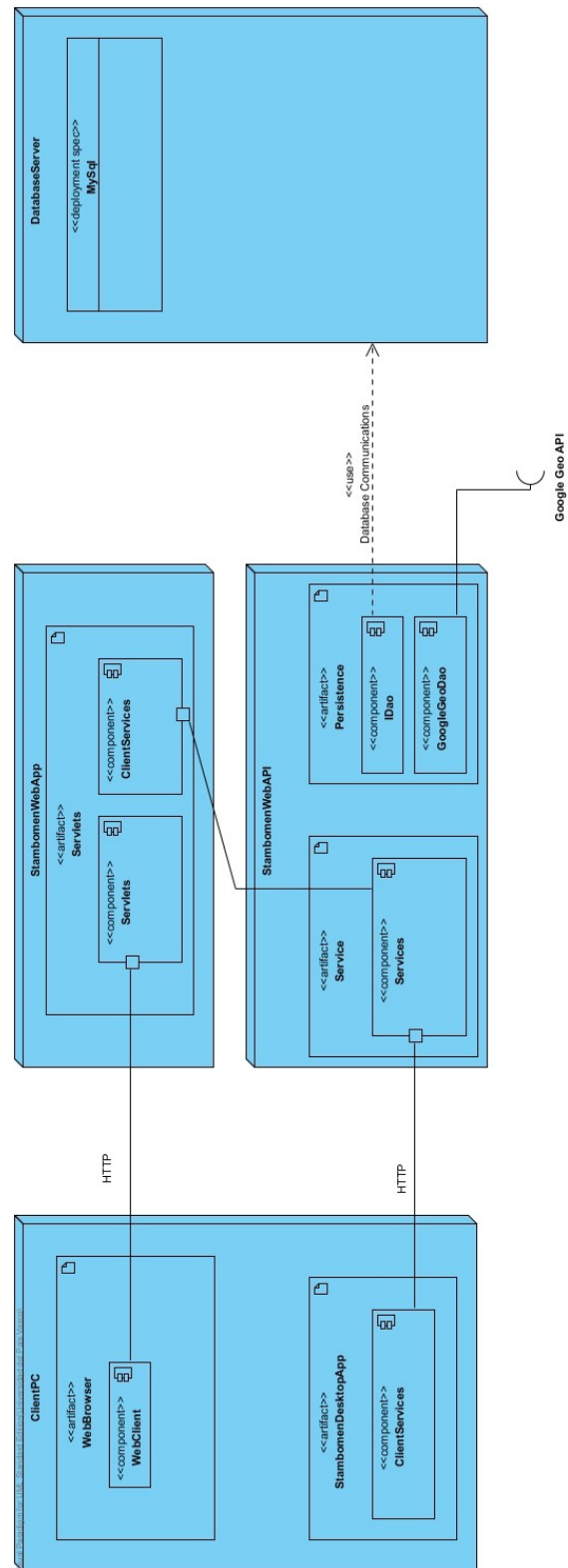
## 4.2 Deployment diagram

Het deployment diagram bestaat uit de clientPC, de StambomenWebApp, de StambomenWebAPI en de DatabaseServer. De client kan via de WebBrowser of de StambomenDesktopApp verbinden met de applicatie. Het verschil ligt hem in het feit dat de webclient eerst met een webserver (de WebApp) verbindt en dan pas met de webAPI vooraleer deze gegevens kan veranderen. Terwijl bij de desktopApp deze rechtstreeks verbindt met de webAPI.

Op de desktopApp zal de user de gegevens van de bomen ingeven en al de nodige bewerkingen hierop kunnen uitvoeren. Terwijl op de webApp deze alle gegevens kan bekijken en gebruiken met het sociale netwerk. Zo kunt je vrienden worden met andere gebruikers en stambomen delen, evenals het delen via facebook en contact leggen met facebookvrienden.

De webAPI bevat de services in de service package die gebruikt worden door de desktopApp en de webApp om bewerkingen van gegevens te kunnen uitvoeren. Via de DAO's (Data Acces Object) in de persistence package worden de gegevens uitgewisseld met de databank. Verder maken we ook gebruik van googleGeoData om coördinaten van een plaats te kunnen opvragen en bijhouden.





## 4.3 DCD Service

De service package bevat de volgende aanspreekpunten: In de services, behalve de basicAuthFilter en de BasicAuthentication wordt alles gelogd, zo kan er gemakkelijk nagegaan worden waar er iets fout loopt.

De UserService:

Deze heeft betrekking op alles wat te maken heeft met een user. De belangrijkste zijn hierbij het aanmaken van een user en de login functie, verder zijn er nog de friendrequest, privacy, language, theme functies.

De TreeService:

Deze heeft betrekking op alles wat te maken heeft met een tree. Zo kan een user een tree aanmaken en trees opvragen.

De PersonService:

Deze heeft betrekking op alles at te maken heeft met een persoon. Zo zal een user eerst een persoon of meerdere aan moeten maken om deze te kunnen manipuleren in de boom en de persoonsgegevens te wijzigen.

De facebookService:

Deze heeft betrekking op alles wat te maken heeft met het facebookgebeuren. Zo kan een user registreren en inloggen met een facebookaccount.

De GedcomService:

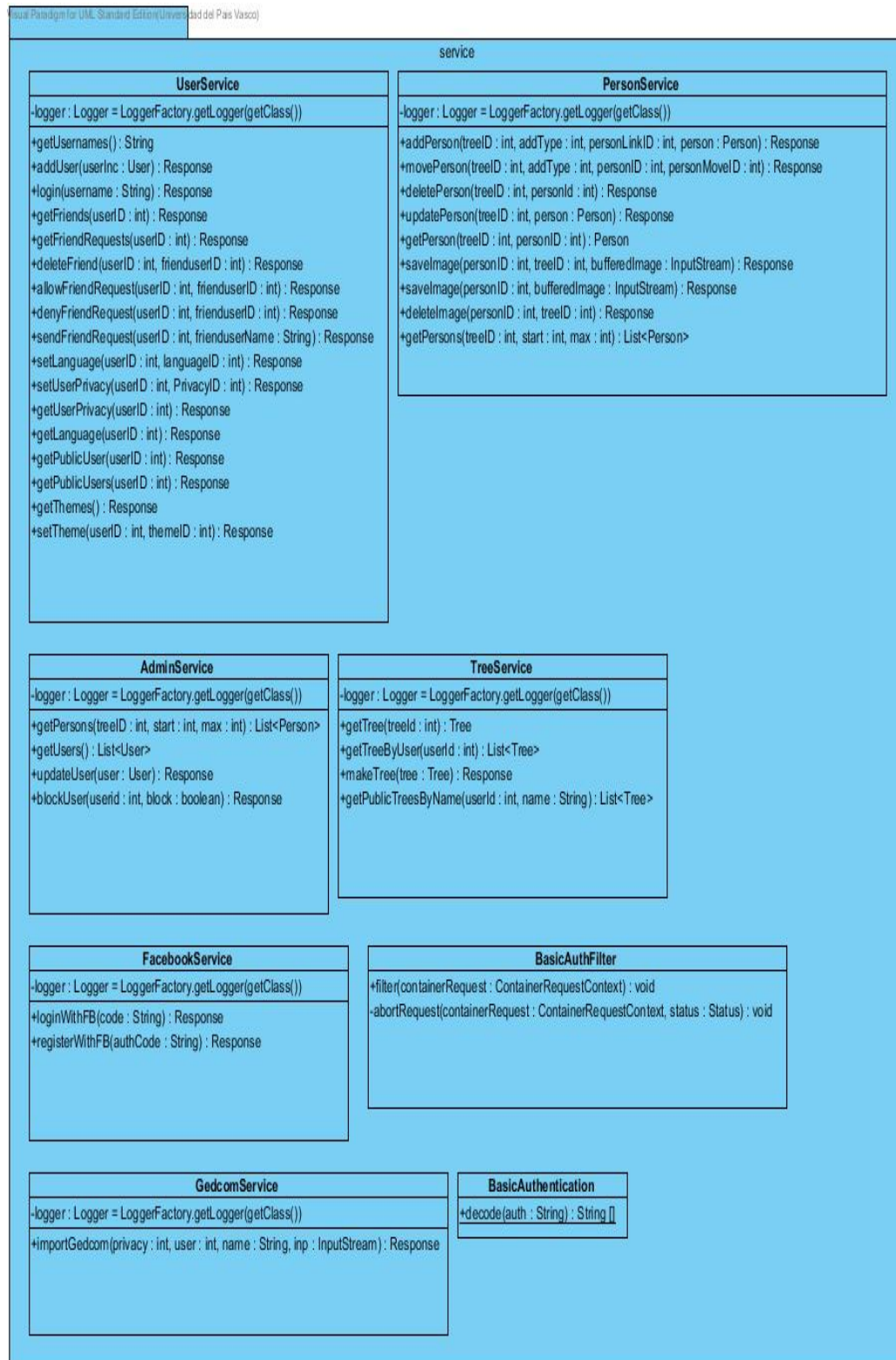
Deze heeft betrekking op alles wat te maken heeft met de gedcombestanden. Met deze service kan de gebruiker een gedcom bestand importeren en zo een hele tree van een ander systeem ophalen en toevoegen aan zijn trees.

De BasicAuthFilter:

Deze service heeft betrekking op het authenticatie gebeuren van de app., zodat niet geautoriseerde gebruikers geen toegang hebben tot het systeem.

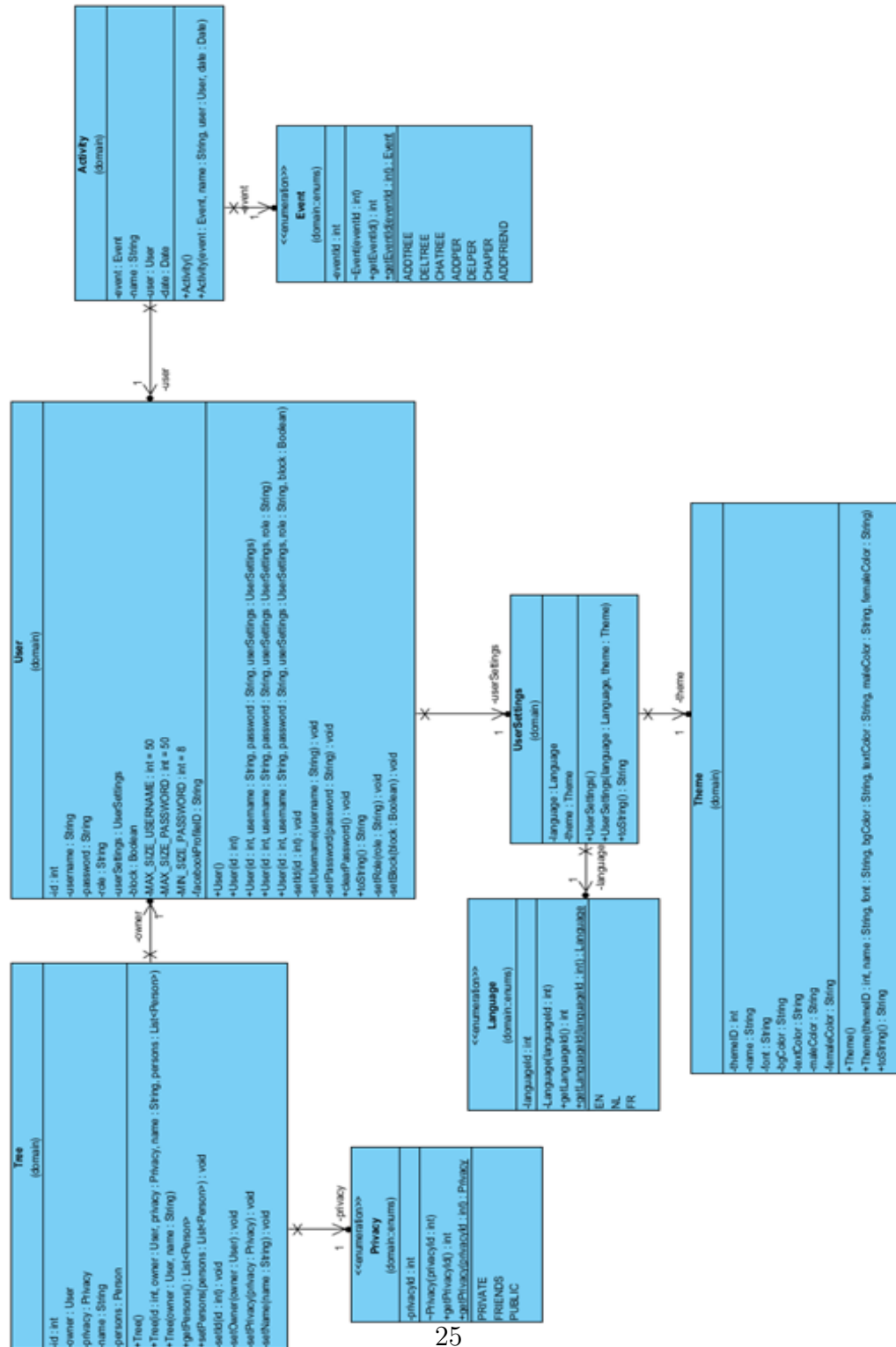
BasicAuthentication:

Hier wordt er data gedecodeerd.



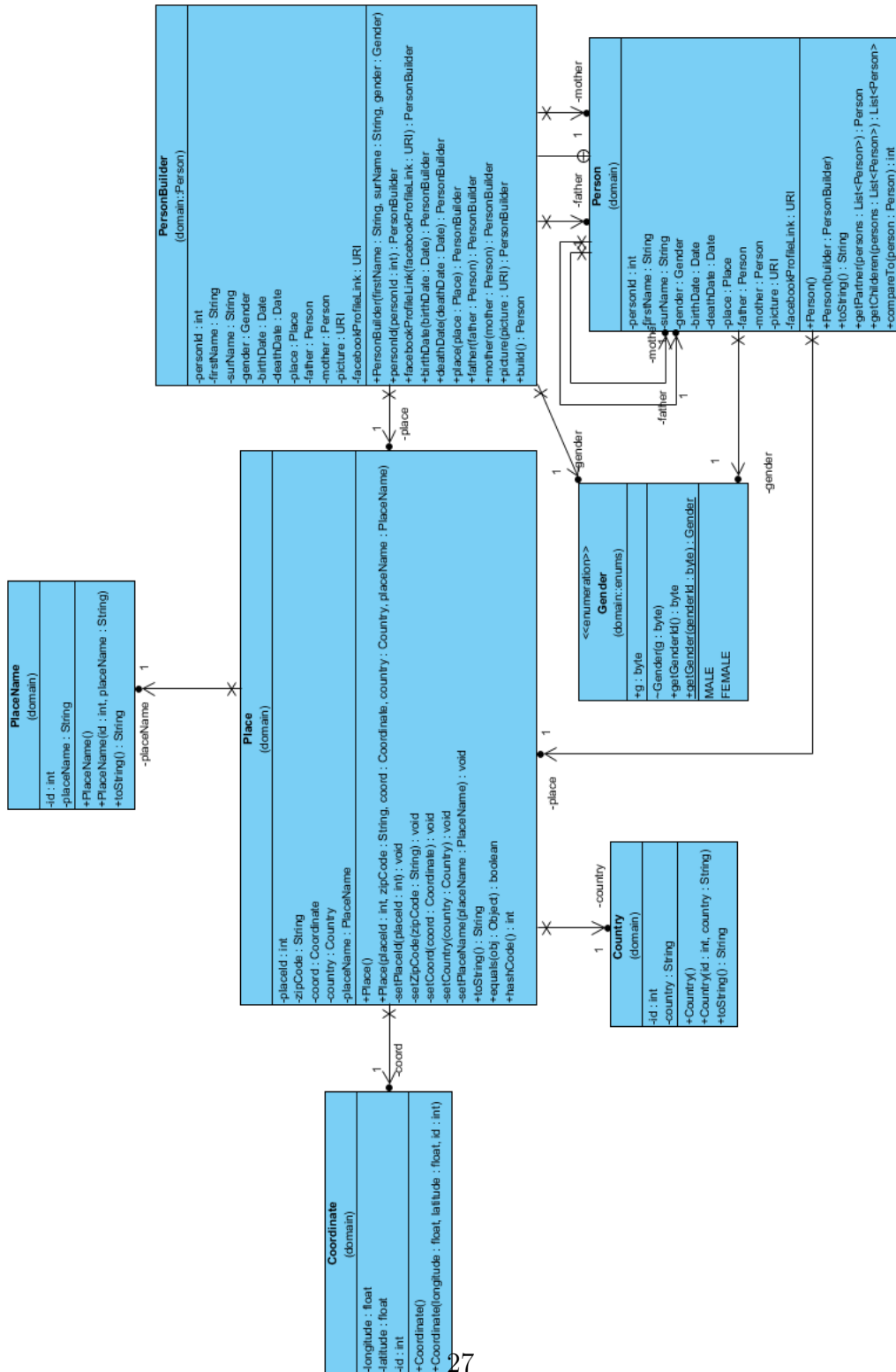


## 4.4 DCD User





## 4.5 DCD Person



## 4.6 Aanmaken van gebruiker

### 4.6.1 Use case

Een gebruiker wordt geregistreerd in het systeem en kan daarna inloggen.

Prioriteit

Hoog

Complexiteit

Gemiddeld

Precondities

De gebruiker heeft nog geen gebruikersnaam en wachtwoord.

Postcondities

De gebruiker heeft zichzelf geregistreerd en kan nu inloggen met de opgegeven gebruikersnaam en wachtwoord.

Primaire actoren

Een gebruiker

Secundaire actoren

Successscenario

1. Het systeem vraagt gebruikersnaam, wachtwoord.
2. De gebruiker geeft het gegeven in.
3. Het systeem valideert.
4. Het systeem registreert de gebruiker.

Alternatieve scenario's

3A. Niet alle gegevens zijn ingevuld.

3A1. Het systeem geeft een melding.

3A2. Terug naar stap 2.

3B. Combinatie wachtwoord en gebruikersnaam zijn ongeldig.

3B1. Het systeem geeft een melding.

3B2. Terug naar stap 2.



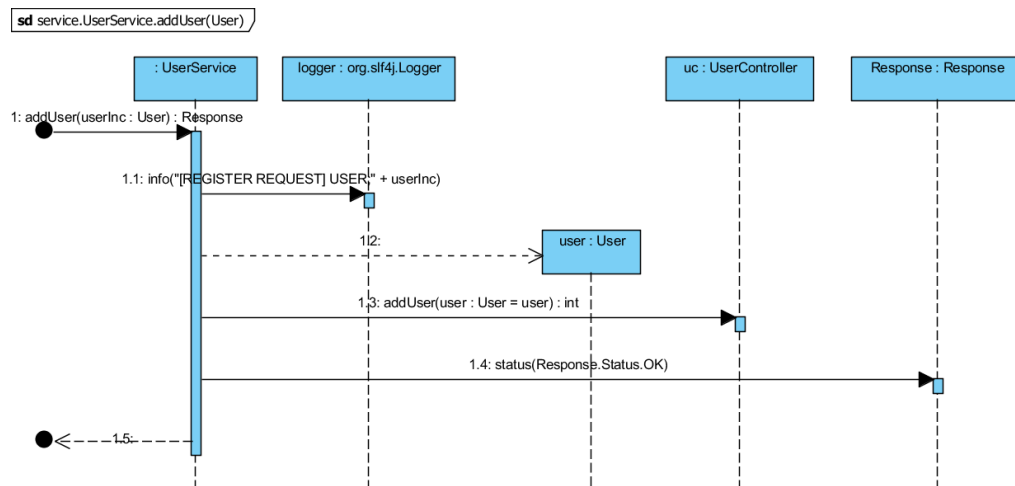
3C. Het gebruikersaccount bestaat reeds.

3C1. Het systeem geeft een melding.

3C2. Terug naar stap 2.

Domeinregels voor gebruikersnaam en wachtwoord? (lengte, tekens, ...?)

## 4.6.2 Sequence Diagram Aanmaken van gebruiker



## 4.7 Inloggen

### 4.7.1 Use case

Een gebruiker logt in op het systeem.

Prioriteit  
Hoog

Complexiteit  
Gemiddeld

Precondities  
De gebruiker is nog niet ingelogd

Postcondities  
De gebruik is ingelogd en heeft toegang tot het systeem. (krijgt rechten)

Primaire actoren  
De gebruiker

Secundaire actoren

Successscenario

1. Het systeem vraagt gebruikersnaam en wachtwoord.
2. De gebruiker geeft de gegevens in.
3. Het systeem valideert de gegevens.
4. Het systeem kent de gepaste rechten toe aan de gebruiker.

Alternatieve scenario's

3A. Niet alle gegevens zijn ingevuld.

3A1. Het systeem geeft een melding.

3A2. Terug naar stap 2.

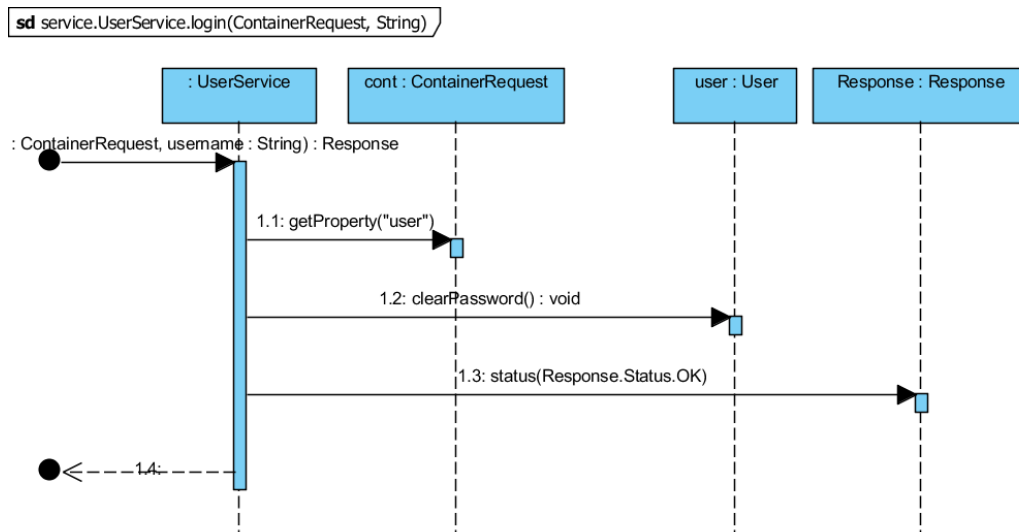
3B. Combinatie wachtwoord en gebruikersnaam zijn ongeldig.

3B1. Het systeem geeft een melding.

3B2. Terug naar stap 2.

Domeinregels voor gebruikersnaam en wachtwoord? (lengte, tekens, ...?)

### 4.7.2 Sequence Diagram Inloggen



## 4.8 Aanmaken stamboom

### 4.8.1 Use case

Een gebruiker maakt een stamboom aan.

Prioriteit

Hoog

Complexiteit

Hoog

Precondities

De gebruiker is ingelogd.

Postcondities

De gebruiker heeft een nieuwe stamboom kunnen maken en deze is gekoppeld aan zijn account.

Primaire actoren

De gebruiker

Secundaire actoren

Successscenario

1. Het systeem vraagt de gegevens (naam, privacy) van een nieuwe stamboom.
2. De gebruiker geeft de gegevens in.
3. Het systeem valideert.
4. Het systeem voegt de stamboom toe aan het account van de gebruiker.

Alternatieve scenario's

3A. Het systeem detecteert dat de naam al in gebruik is voor een andere stamboom.

3A1. Het systeem geeft een melding.

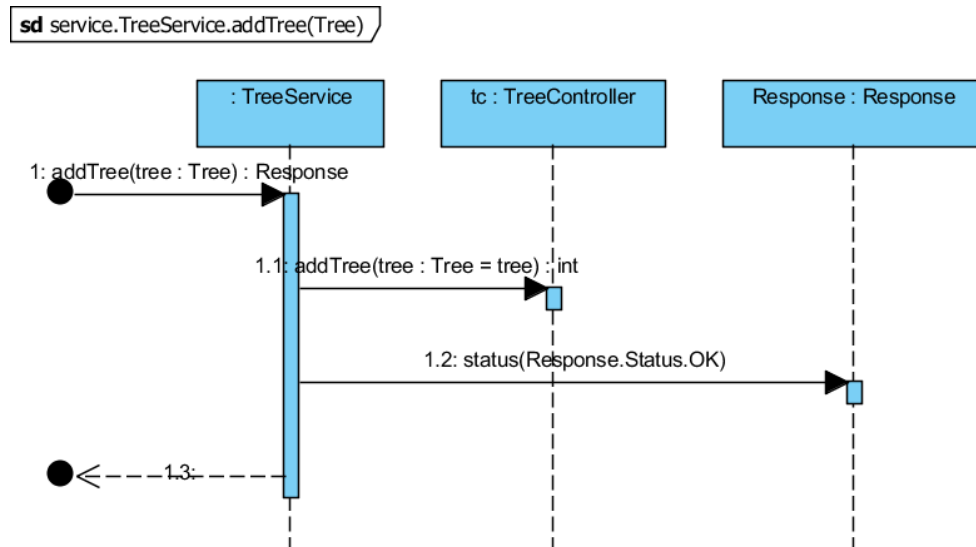
3A2. Terug naar stap 2.

3B. Het systeem detecteert foutieve invoer.

3B1. Het systeem geeft een melding.

3B2. Terug naar stap 2.

### 4.8.2 Sequence Diagram Aanmaken stamboom



## 4.9 Toevoegen persoon

### 4.9.1 Use case

Een gebruiker voegt een persoon toe.

Prioriteit

Hoog

Complexiteit

Hoog

Precondities

De gebruiker is ingelogd en heeft een stamboom geselecteerd.

Postcondities

De gebruiker heeft een persoon toegevoegd aan de geselecteerde stamboom.

Primaire actoren

De gebruiker

Secundaire actoren

Successscenario

1. Het systeem vraagt waar de persoon moet toegevoegd worden.
2. De gebruiker geeft de plaats
3. Het systeem valideert.
4. Het systeem vraagt de persoonlijke gegevens van de persoon.
5. Het systeem valideert.
6. Het systeem voegt de gebruiker toe aan de stamboom.

Alternatieve scenario's

3A. Het systeem detecteert een ongeldige plaats.

3A1. Het systeem geeft een melding.

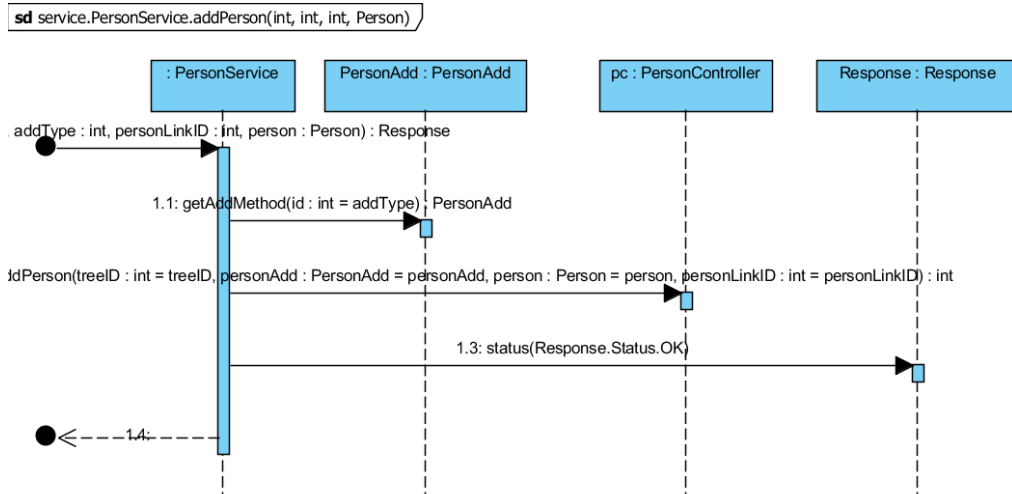
3A2. Terug naar stap 1.

5. Het systeem detecteert foutieve invoer.

5A1. Het systeem geeft een melding

5A2. Terug naar stap 4.

### 4.9.2 Sequence Diagram Toevoegen persoon



## 4.10 Versturen Van Vriendschapsverzoek

### 4.10.1 Use case Versturen Van Vriendschapsverzoek

Een gebruiker verzendt een vriendschapsverzoek.

Prioriteit  
Gemiddeld

Complexiteit  
Gemiddeld

Precondities  
De gebruiker is ingelogd.

Postcondities  
De gebruiker heeft een nieuwe vriend in zijn vriendenlijst.

Primaire actoren  
De gebruiker

Secundaire actoren  
De mogelijke vriend

Successscenario

1. De gebruiker verstuurd een vriendschapsverzoek naar een mogelijke vriend.
2. Het systeem meldt aan de mogelijke vriend dat hij een vriendschapsverzoek heeft.
3. De mogelijke vriend aanvaardt het vriendschapsverzoek.
4. Het systeem valideert.
5. Het systeem heeft een mogelijke vriend toegevoegd aan de vriendenlijst van de gebruiker.

Alternatieve scenario's

- 3A1. De mogelijke vriend weigert het vriendschapsverzoek.  
3A2 Het systeem heeft de mogelijke vriend niet toegevoegd aan de vriendenlijst van de gebruiker.

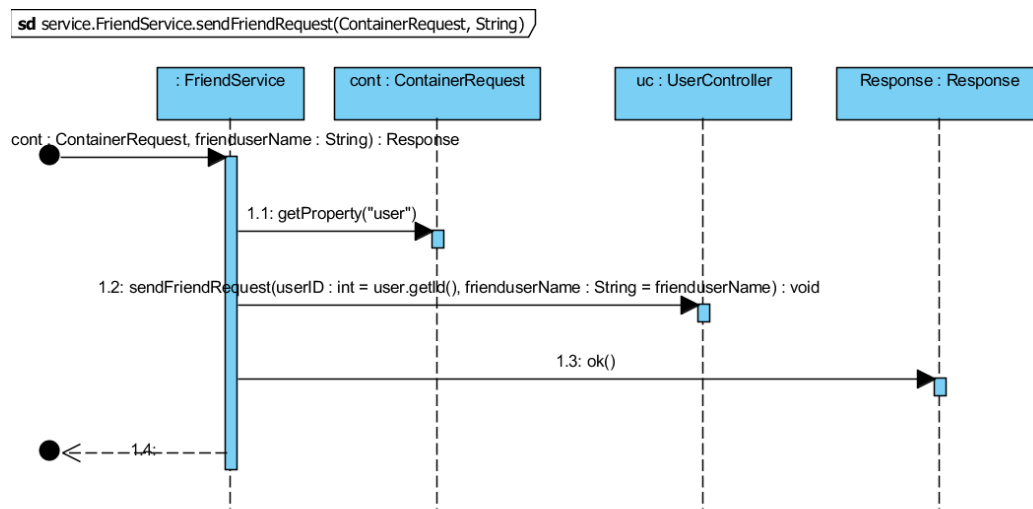
- 4A1. Het systeem detecteert foutieve invoer.



4A2. Terug naar stap 2.

Statuscode weigering is 2, verzoek 0 en geaccepteerd 1.

## 4.10.2 Sequence Diagram Versturen Van Vriendschapsverzoek



# Chapter 5

## Ontwerpbeslissingen

### 5.1 Best Practice

#### 5.1.1 JSP vs Servlets

Voor de WebApplicatie kan er zowel in de JSP-pagina's als in de Servlets Java-code worden geschreven. Dit kan er al snel voor zorgen dat de code onoverzichtelijk wordt omdat ze her en der verspreid is.

Wij hebben er voor gekozen om de Java-code binnen de Servlets te houden. Dit zorgt er voor dat de code gecentraliseerd blijft en gemakkelijker te onderhouden is. In de JSP-pagina's wordt de informatie die verkregen wordt via requests of sessies verwerkt met de JSTL-bibliotheek. Dankzij de eenvoudige html-syntax integreert deze gemakkelijk met de rest van de code.

# Chapter 6

## Implementatiekeuzes

### 6.1 Bibliotheken

#### 6.1.1 Jersey

Jersey is een JAX-RS referentie-implementatie die een eigen API aanbied door de JAX-RS toolkit uit te breiden met extra functies en hulpprogramma's om op een eenvoudige manier RESTful Webservices en client development aan te bieden. JAX-RS : Java API voor REST Web Services biedt ondersteuning bij het creëren van web services volgens het Representational State Transfer (REST) pattern. REST is een pattern die beschrijft hoe resources geadresseerd en gebruikt kunnen worden. Een resource kan aangesproken worden dankzij een gemeenschappelijke interface op basis van de HTTP standaard methodes. Een REST server zorgt ervoor dat de client een verbinding kan maken om de resources op te halen en te wijzigen. REST wordt veel gebruikt voor het bouwen van webservices en noemen we respectievelijk RESTful Webservices. Voor de applicatie hebben we gebruik gemaakt van Jersey die zorgt voor de REST server en een REST client.

#### 6.1.2 Swagger Jersey

Swagger is een specificatie en compleet uitvoeringskader voor het beschrijven, het produceren , consumeren en visualiseren van REST webservices. Het doel van Swagger om client en documentatie systemen de mogelijkheid te geven op

hetzelfde tempo te werken als de server. De documentatie van de methoden, parameters en modellen zijn nauw geïntegreerd in de server-code, waardoor de Apis altijd gesynchroniseerd zijn.

Door gebruik te maken van Swagger kunnen cliënten diensten consumeren en toegang tot de server code zonder in aanraking te komen de implementatie van de server. De interface van het framework zorgt ervoor dat er interactie mogelijk is met de API in een sandbox omgeving. Swagger ondersteunt JSON en XML en zal in de toekomst ook beschikbaar zijn in andere formaten.

### 6.1.3 SLF4J

SLF4J is een Java logging api die gebruik maakt van een facade pattern, dit is een software design pattern die gebruikt wordt in object-oriented programming om een complex systeem voor te stellen als een interface. De facade pattern is ideaal bij het werken met een groot aantal onderling afhankelijke klassen of klassen die meerdere methoden gebruiken, vooral wanneer deze te ingewikkeld zijn om te gebruiken of moeilijk te begrijpen. De onderliggende logging backend van SLF4J wordt bepaald op runtime door het toevoegen van de gewenste binding aan het classpath ( java.util.logging). Om gebruik te maken van SLF4J moet je de slf4j-api-1.7.7.jar en slf4j-simple-1.7.7.jar plaatsen in de classpath van het project. Bij het volgende voorbeeld roepen we de logger-factory op om dan vervolgens het toevoegen van een persoon te registreren :

```
private final Logger logger = LoggerFactory.getLogger(getClass());

public Person getPerson(int treeID, int personID)
{
    logger.info("[PERSON CONTROLLER] Getting person by id " + personID);
    return pc.getPerson(treeID, personID);
}
```

### 6.1.4 Junit 4.x

JUnit is een unit testing framework voor Java die gebruikt wordt in test-driven development die deel uit maakt van xUnit. Een unit test is een stukje code die een specifieke functionaliteit uitvoert om de code testen. Het percentage van de code die wordt getest door unit tests wordt meestal de test coverage genoemd. Een unit test richt zich op een kleine eenheid van de code, bv een

methode of een klasse. Een integratie of functionele test heeft als doel om het gedrag van een component of de integratie tussen een set van componenten te testen. Deze soort van testen kunnen user stories vertalen in een test suite, dit zou een soort sandbox omgeving zijn waarin alle methoden kunnen getest worden op bijvoorbeeld verkeerde input, lege waarden enz. Naast integratie testen kan JUnit ook gebruikt worden voor het benchmarken van verschillende software componenten.

Aan de hand van een voorbeeld waarbij we een test unit maken voor de UserController is het eenvoudig om de verschillende mogelijkheden van JUnit uit te leggen.

Om in Netbeans een JUnit test aan te maken voor de UserController navigeer je naar het tools menu en klik je op Create Tests.

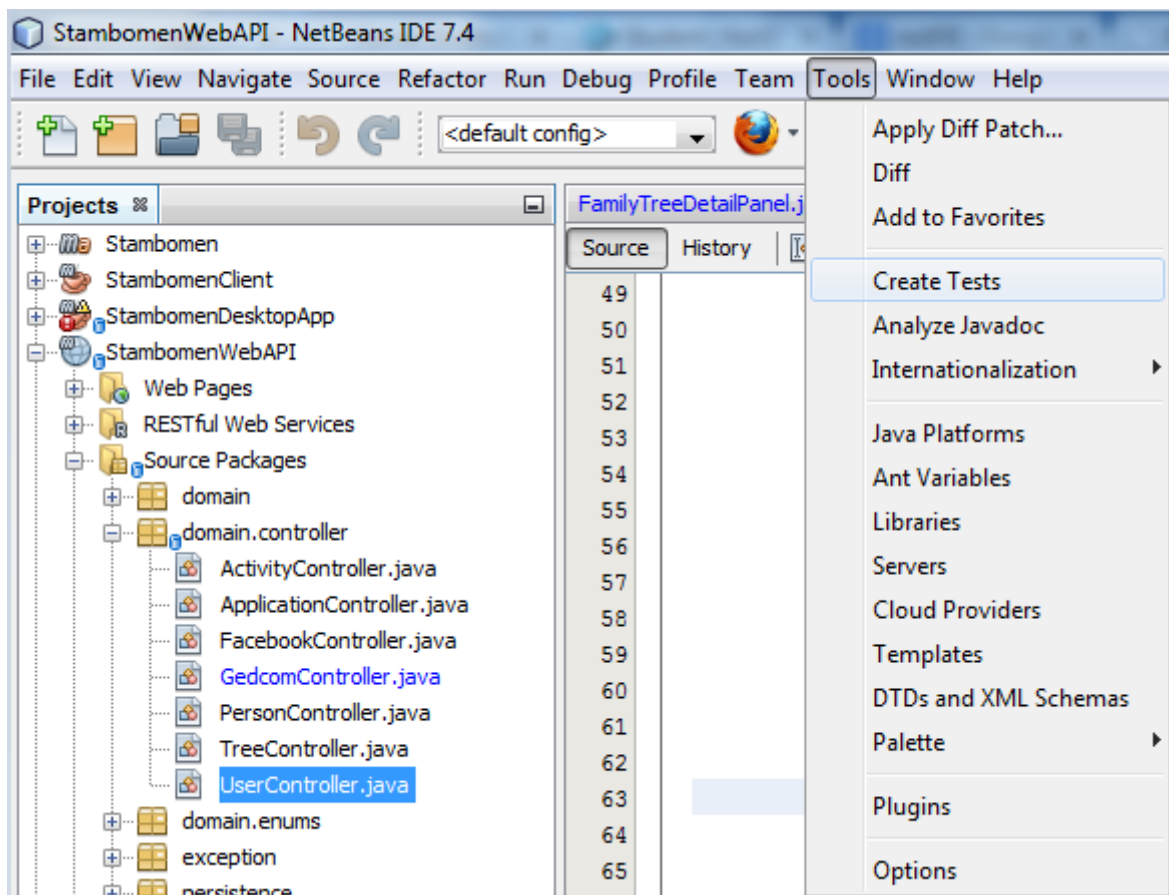
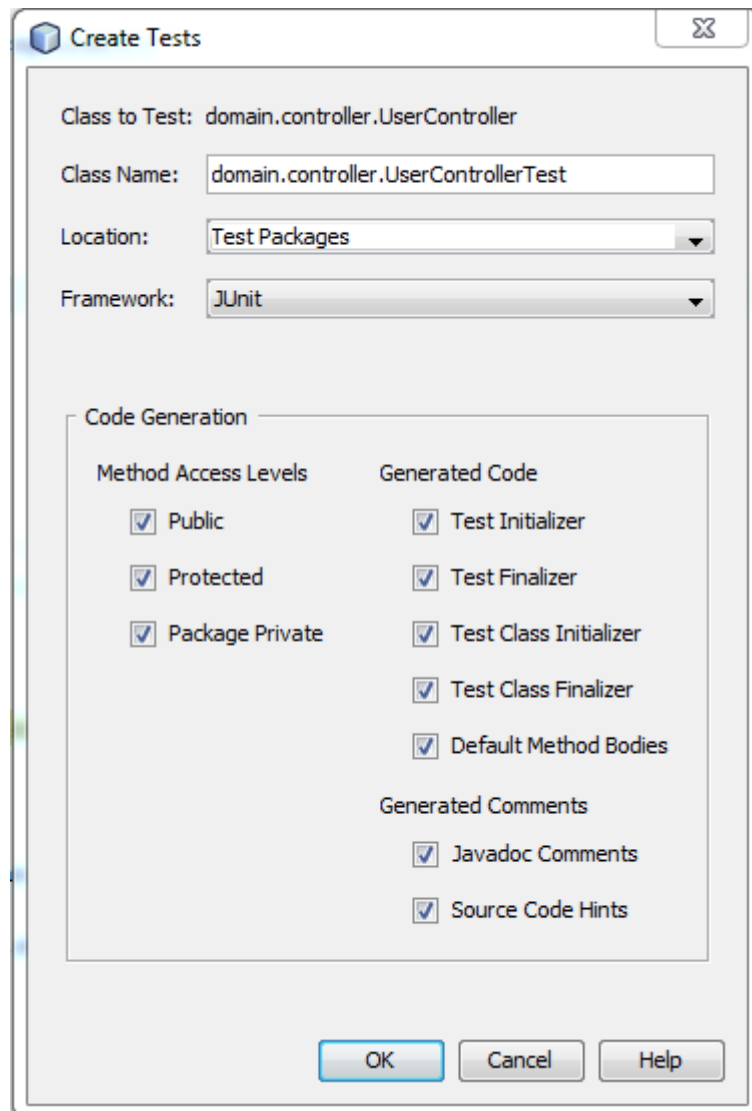


Figure 6.1: SLF4J Logging

Er verschijnt een nieuw venster waarin er gevraagd wordt om een naam op te geven voor de JUnit test en welke code gegenereerd moet worden. Belangrijk hierbij is dat de class steeds moet eindigen op Test.



Netbeans zal nu een nieuwe class aanmaken die een default constructor bevat die bij de UserControllerTest gebruikt wordt om de usercontroller, theme en user object aan te maken.

```

public class UserControllerTest
{

    private static UserController uc;
    private static User user;
    private static Theme theme;

    public UserControllerTest()
    {
        uc = new UserController();
        theme = new Theme(1, "Default", "Valera", "FFFFFF", "252525", "334455", "B03A3A");
        user = new User(-1, "Testuser", "1234567", new UserSettings(Language.EN, theme));
    }
}

```

JUnit maakt gebruik van annotaties om de methoden die een test uitvoeren te identificeren en worden enkel opgenomen in een Test class. Een overzicht van de annotaties :

@Before and @After Before en After zal worden aangeroepen voor en na elke test case. Voor de UserControllerTest wordt dit gebruikt om de databank op te kuisen door de aangemaakte test gebruiker te verwijderen.

```

@Before
public void setUp()
{
    System.out.println("Een methode met de Before annotation wordt uitgevoerd voor de JUnit test");
}

@After
public void tearDown()
{
    System.out.println("Een methode met de After annotation wordt uitgevoerd na de JUnit test");
    user = uc.getUser(user.getUsername());
    try
    {
        uc.deleteUser(user.getId());
    }
    catch (NullPointerException e)
    {
    }
}
}

```

@BeforeClass and @AfterClass Vergelijkbaar met Before and After maar deze zal pas worden uitgevoerd na de JUnit Test en net voor of na de Class.

```

@BeforeClass
public static void setUpClass()
{
    System.out.println("Een methode met de BeforeClass annotation wordt uitgevoerd na de JUnit test");
}

@AfterClass
public static void tearDownClass()
{
    System.out.println("Een methode met de AfterClass annotation wordt uitgevoerd na de JUnit test");
}

```

@Test Een test annotatie wordt gebruik voor een normale test;

@Ignore Wordt gebruikt om een bepaalde test over te slaan indien deze nog niet

```

@Ignore("Nog niet geïmplementeerd")
@Test
public void testMergeUser() {
    System.out.println("testMergeUser");
    fail("@Wordt niet uitgevoerd");
}

```

geïmplementeerd. }

@Test(timeout=500) Het is mogelijk om een unit te testen na een bepaalde periode in milliseconden.

@Test(expected=IllegalArgumentException.class) Soms is het nodig om te controleren of een bepaalde exceptie gesmeten wordt , in ons voorbeeld met de UserControllerTest is dit bij het opvangen van een gebruiker die al bestaat in de databank.

```

@Test(expected = UserAlreadyExistsException.class)
public void testAddUserExists() throws UserAlreadyExistsException
{
    testAddUser();
    System.out.println("testAddUserExists");
    uc.addUser(user);
}

```

Assert statements

JUnit biedt statische methoden aan in de Assert class die ervoor zorgen dat er testen kunnen opgesteld worden met bepaalde voorwaarden. Een assert methode



vergelijkt de werkelijke waarde die teruggegeven wordt door een test met de verwachte waarde en gooit een `AssertException` indien deze vergelijking mislukt.

Een overzicht van asserts:

`fail(String)` een fail op het toevoegen van de user forceren.

```
@Test
public void testAdduserFail()
{
    System.out.println("addUser fail");
    uc.addUser(user);
    fail("Laat de methode mislukken.");
}
```

`assertTrue(Bericht,boolean voorwaarde)` Controleert of de boolean voorwaarde waar is.

`assertFalse(Bericht, boolean voorwaarde)` Controleert of de boolean voorwaarde onwaar is.

`assertNull(bericht ,object)` Controleert of het object null is

`assertNotNull(bericht ,object)` Controleert of het object niet null is.

```
@Test(expected = UserAlreadyExistsException.class)
public void testAddUserExists() throws UserAlreadyExistsException
{
    testAddUser();
    System.out.println("testAddUserExists");
    uc.addUser(user);
}
```

`AssertEquals(bericht, verwacht ,werkelijk)` Test of twee waarden gelijk zijn.

Indien je met meerdere test klassen werkt kunt u deze combineren in een test suite. Het uitvoeren van een testsuite bevat alle test classes van het project.

### 6.1.5 Gedcom

GEDCOM is een speciaal tekstformaat die ontwikkeld is door de Kerk van Jezus Christus van de Heiligen der Laatste Dagen en is bedoeld als standaard voor communicatie tussen de Kerk en personen die genealogische data aanleveren. Dit formaat heeft zich nu ontwikkeld tot de standaard voor gegevensuitwisseling tussen de meeste genealogische programma's en systemen.

Stel dat we in de applicatie een persoon toevoegen dan zouden deze gegevens in een GEDCOM als volgt voorgesteld worden :

<b>Detail</b>	
Voornaam	Kenzo
Achternaam	Clauw
Geslacht	<input checked="" type="radio"/> Man <input type="radio"/> Vrouw
Geboortedatum	15-mrt-1991
Overlijdingsdatum	
<b>Adres</b>	
Gemeente	Oostende
Postcode	8400
Land	Belgie

Het eerste deel bevat de header met de gebruikte versie,character encoding (ANSI,UNICODE of ASCII) als de belangrijkste informatie.

```
0 HEAD
1 SOUR naam
2 VERS V4.0
1 DEST naam
1 DATE 10 MAY 2014
1 FILE C:\Users\admin\gedcom.ged
1 GEDC
2 VERS 5.5
1 CHAR ANSI
```

Het tweede deel bevat informatie over de personen.

```
0@I1@ INDI
1 NAME Kenzo/Clauw
1 SEX M
1 BIRT
2 DATE 15 MAY 1991
2 PLAC Oostende
1 FAMS @F1@
```

```
0@I2@ INDI
1 NAME Manon/Jonckheere
1 SEX F
1 BIRT
2 DATE 27 FEB 1992
2 PLAC Oostende
1 FAMS @F1@
```

Het laatste deel bevat de relaties tussen personen met TRLR die het bestand afsluit.

```
0 @F1@ FAM
1 HUSB @I1@
1 WIFE @I2@
0 TRLR
```

### 6.1.6 Gedcom4j

Gedcom4j is een gratis open-source Java library voor het laden ( parsing ) en opslaan van gegevens in Gedcom genealogie 5.5 of 5.51 bestanden naar een Java-object hiërarchie.

Om gebruik te maken van Gedcom4j moet je de gedcom4j.jar plaatsen in de classpath van het project.

Om toegang tot gegevens te verkrijgen maak je gebruik van de properties binnen de `gp.gedcom` structure waarbij de personen voorgesteld worden door het object `Individual` en de relaties door `Family`.

Laden van een gedcom bestand :

```
GedcomParser gp = new GedcomParser(); gp.load("sample/TGC551.ged");
```

Om de personen te overlopen :

```
for (Individual i : g.individuals.values()) {  
    System.out.println(i.formattedName());  
}
```

Om de relaties met kinderen te overlopen :

```
for (Family f : g.families.values()) {  
    f.wife.formattedName();  
    f.husband.formattedName();  
    for (Individual c : f.children)  
    {  
        c.formattedName();  
    }  
}
```

### 6.1.7 Sardine

Sardine is een WebDAV-client voor Java die een groot deel van de WebDAV client specificaties bevat.

WebDAV is een uitbreiding op het HTTP protocol en staat voor Web-based Distributed Authoring and versioning.

Dit protocol wordt gebruikt om op afstand bestanden aan te maken, wijzigen of te verplaatsen op een server via het internet.

De meeste moderne besturingssystemen ondersteunen WebDAV, waardoor een gebruiker bestanden op de server kan plaatsen alsof ze lokaal opgeslagen worden.

Gebruik :

Om gebruik te maken van Sardine plaats je de commons-logging.jar, commons-codec.jar en HttpClient/HttpCore 4.1.1 in je classpath.

Voor onze applicatie hebben we gebruik gemaakt van Sardine in de ImageDAO om images up te loaden naar de Server.

SardineFactory Om gebruik te maken van Sardine interface maak je instantie van de SardineFactory aan om aan de hand van HTTP Authenticatie te verbinden met de WebDAV server.

```
public ImageDao()
{
    try
    {
        setUrlPath();
        sardine = SardineFactory.begin("team12", "RKAxujnJ");
    }
    catch (UnknownHostException ex)
    {
        Logger.getLogger(ImageDao.class.getName()).log(Level.SEVERE, null, ex);
    }
    catch (SardineException ex)
    {
        Logger.getLogger(ImageDao.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Bij het doorgeven van credentials aan de client wordt de authenticatie scope geconfigureerd voor Basic , Digest en NTML verificatie om te reageren op de server.

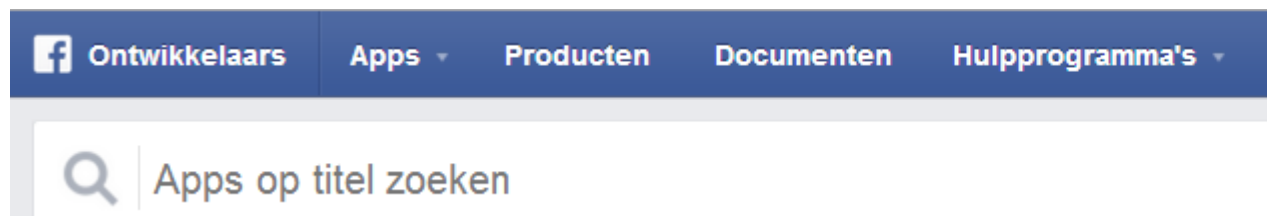
put (String url, byte [] gegevens) Aan de hand van HTTP put wordt een bestand op de server geplaatst.

```
private void uploadImage(String url, BufferedImage bufferedImage)
{
    try
    {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ImageIO.write(bufferedImage, "jpg", baos);
        baos.flush();
        byte[] imageInByte = baos.toByteArray();
        baos.close();
        sardine.put(url, imageInByte);
    }
}
```

### 6.1.8 RestFB

RestFB is een efficiënte Java implementatie van een Facebook Graph Api. Het voordeel van RestFB is dat er ook ondersteuning is voor oudere versies van de API omdat sommige features nog niet geïmplementeerd zijn in de Graph API.

Registeren van een developer app : Ga naar <https://developers.facebook.com/apps> om een nieuwe app te maken.



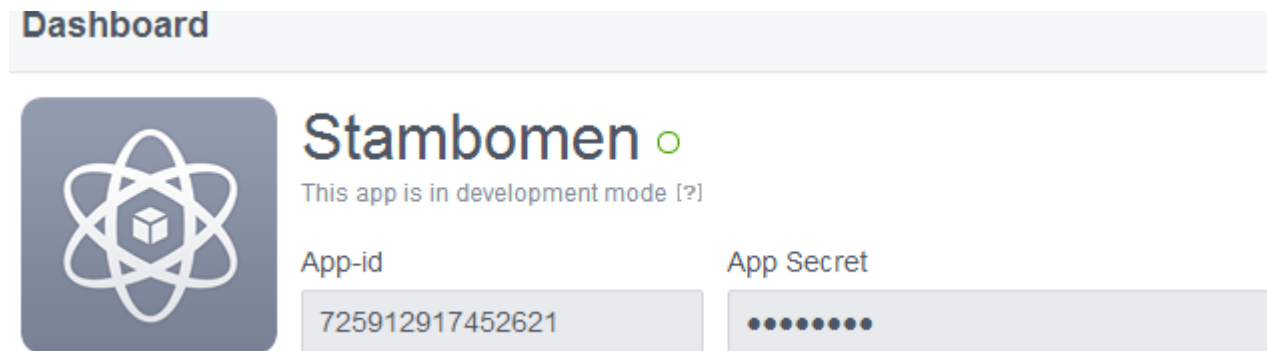
Een aanvraag bevestigen :



Geef een naam op voor de nieuwe App Stambomen.

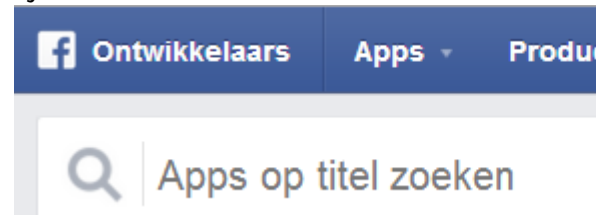
A screenshot of the Facebook 'Create a New App' form. The title is 'Create a New App' with the subtitle 'Get started integrating Facebook into your app or website'. The form has three input fields: 'Display Name' with the value 'Stambomen', 'Namespace' with the placeholder 'A unique identifier for your app (optional)', and 'Categorie' with a dropdown menu showing 'Hulpprogramma's'. At the bottom, there is a text line 'By proceeding, you agree to the Facebook Platform Policies' and two buttons: 'Annuleren' and 'App maken'.

Op je dashboard verschijnt er een overzicht van je App met App-id en geheime code die gebruikt worden in de applicatie om een verbinding te maken.



Om in de applicatie in te loggen met facebook maak je een classFacebookClient

aan waarin de secret code van de API vermeld wordt.



Om een user op te vragen roep je in de FacebookClient de methode fetchObject op.

### 6.1.9 JCalendar

JCalendar is een Java-date chooser om grafisch een datum te selecteren en bestaat uit een combinatie van de volgende componenten :

JDateChooser

Om een datum te kiezen kan je ofwel de datum rechtstreeks invullen of klikken op de afbeelding om de datum te kiezen. De default date editor vangt de nodige fouten op met een respectievelijke rode kleur.

JDayChooser

Om een dag te kiezen klik je op het dag nummer die hoort bij de weeknummers



en dagen die afhankelijk zijn van de locale.

JMonthChooser

Een JComboBox waarbij een maand kan geselecteerd worden uit een keuzelijst of door de knoppen te verhogen / verlagen. De taal waarin de namen worden vermeld zijn afhankelijk van de Locale.

JYearChooser

Een JSpinField die het toelaat om een jaar te selecteren door de knoppen te verhogen / verlagen of deze manueel in het tekstveld invoeren.

### 6.1.10 Abego TreeLayout

ABEGO TreeLayout is een Efficiënt en aanpasbare Boom Layout algoritme voor Java. De TreeLayout creëert boom lay-outs voor willekeurige bomen. Het is niet beperkt tot een bepaalde productie of formaat, maar kan worden gebruikt voor elke vorm van tweedimensionale tekening. Voorbeelden zijn Swing gebaseerde componenten, SVG-bestanden, en nog veel meer.

Om de Treelayout te gebruiken moet je een instantie van de klasse TreeLayout voorzien van de knooppunten van de boom inclusief zijn kinderen samen met de hoogte tussen verschillende niveaus.

Eigenschappen : Op basis van deze informatie zorgt de TreeLayout voor een compacte layout met een overzichtelijke boom.

De indeling toont de hiërarchische structuur van de boom, namelijk de y-coördinaat van een knooppunt wordt bepaald door het niveau.

De randen kruisen elkaar niet en de nodes op hetzelfde niveau hebben een minimale horizontale afstand. De volgorde van de kinderen van een knooppunt wordt weergegeven in de tekening.

Het algoritme werkt symmetrisch, dat wil zeggen de tekening van de weerspiegeling van een boom is het gereflecteerde tekening van de oorspronkelijke boom

Om gebruik te maken van de Abego TreeLayout moet je de org.abego.treelayout.core.jar(de TreeLayout algoritme kern) en org.abego.treelayout.netbeans.jar( gebruikt van de

NetBeans visuele API ) plaatsen in de classpath van het project.

### 6.1.11 Java FX

JavaFX is een platform die gebruikt wordt om Rich Internet Applications te ontwikkelen die audio, video en webservices aanbieden en kan gebruikt worden op computers of smartphones. Rich Internet Applications zijn interactieve internet-applicaties die het gevoel van een desktopprogramma geven.

Omdat JavaFX gebaseerd is op Java is het platformafhankelijk met als voordeel dat alle Java-bibliotheken kunnen worden gebruikt bij het ontwikkelen van applicaties. Applicaties worden geschreven in JavaFX Script, een declaratieve scripttaal die speciaal is ontwikkeld voor webontwikkelaars en designers die visueel willen programmeren. De applicaties in JavaFX kunnen worden uitgevoerd in een webbrowser of kunnen worden geïnstalleerd door ze naar het bureaublad te slepen;

### 6.1.12 JQuery

JQuery is een krachtig javascript-bibliotheek die het gebruiken van javascript-evenementen, javascript-functies en, html en css modificaties vereenvoudigd. Het is een gratis, licht en krachtig framework die ingeburgerd is in de wereld van webdevelopment.

Aangezien de meeste van het werk in de WebApplicatie op de achtergrond in Java gebeurt, waren er slechts een aantal specifieke toepassingen voor JQuery in onze applicatie. Zo werd JQuery voornamelijk gebruikt bij de teletijdschine die een interactieve interface vereiste.

### 6.1.13 JQuery UI

JQuery UI is een uitbreiding op JQuery die extra interactieve evenementen toevoegt, een aantrekkelijke UI voor bepaalde componenten en meer.

Ook deze bibliotheek werd voornamelijk gebruikt in de WebApplicatie voor de teletijdschine.

### **6.1.14 JSTL**

JSTL is een afkorting voor JSP - Standard Tag Library wat een bibliotheek is die je op een eenvoudige manier objecten van de backend laat verwerken in de JSP-pagina's. Alle algemeen gekende operaties zoals if-then-else, for-loops, while-loops, enz. zijn aanwezig. De bibliotheek maakt gebruik van standaard html-tags waardoor de code mooi geïntegreerd worden in de html-code.

## Chapter 7

# Kwaliteitscontrole

Om de kwaliteit van onze applicatie te garanderen wordt er gebruik gemaakt van Test-driven development die een onderdeel is van agile softwareontwikkeling.

Hoe werkt Test-driven development?

Voor dat er code geschreven wordt maak je eerst een geautomatiseerde test waarbij je rekening moet houden met alle mogelijkheden van invoer , errors en uitvoer. Op deze manier moet je nog geen rekening houden met code. De eerste keer dat een test uitgevoerd wordt moet deze een error produceren omdat er nog geen code aanwezig is. Na het afwerken van de test is het de bedoeling om op basis van de tests code te schrijven. Eens de code met succes de verschillende testen kan doorstaan kun je bepaalde bugs uit je applicatie al uitsluiten.

Voor onze applicatie hebben we gekozen voor JUNIT die het mogelijk maakt om aan de hand van functionele of integratie testen te debuggen volgens de opgestelde use cases.

Wat moet er getest worden?

In het algemeen is het veilig om bepaalde methoden zoals getters en setters die gewoon waarden oproepen of toewijzen te negeren. Het schrijven van tests hiervoor is tijdrovend en zinloos omdat je op deze manier de Java Virtual Machine zou testen waarvoor er al testen voorzien zijn.

Het is aangeraden om voor de kritische en complexe onderdelen van uw een applicatie uitgebreide testen te voorzien. Een stevige test suite beschermt u ook tegen regressie in de bestaande code als je nieuwe features introduceert.

# Chapter 8

## Gekende problemen

### 8.1 WebApp

1. Facebook share button (onderdeel van de facebook social plugins) toevoegen gaat enorm traag wanneer er enorm veel share button elementen geïnjecteerd moeten worden. Om deze plug-in te kunnen gebruiken wordt er gebruik gemaakt van facebook hun javascript SDK, deze wordt asynchroon geladen en zal de rendering van de pagina niet in de weg lopen. Doordat de share button telkens in de page load alle elementen moet overlopen om te zoeken waar er een share button moet toegevoegd worden, bevriest de GUI een beetje.
2. Wanneer een rechtstreekse link ingegeven wordt naar een pagina van de stambomenWebApp (bvb. naar een specifieke stamboom) zonder ingelogd te zijn zal deze redirecten naar de loginpagina. Wanneer er dan ingelogd wordt zal de pagina van de ingegeven link niet verkregen worden, maar de main pagina.
3. Geen paginering bij trees, vrienden, user profiles, wanneer deze veel elementen bevatten zal de pagina minder overzichtelijk worden.

## 8.2 DesktopApp

1. Wanneer een nieuwe tree aangemaakt is met een gebruiker en een image aan deze gebruiker wordt toegevoegd wordt er geen nieuwe map (met de naam van de tree) aangemaakt op de server. Daardoor zal er geen image toegevoegd worden aan de persoon.

# Chapter 9

## Verbeterpunten en uitbreidingsmogelijkheden

Er zijn nog verschillende uitbreidingsmogelijkheden voor de applicatie die kunnen onderverdeeld worden volgens :

### 9.1 Gedcom

1. Exporteren van gedcom bestanden Op de documentatie pagina van Gedcom4j staat er een perfect voorbeeld van hoe een GEDCOM bestand kan opgesteld worden met de nodige controle door een gedcomvalidator vanaf nul. <http://gedcom4j.org/main/example-creating-gedcom-scratch>
2. Meerdere stambomen uit een gedcom bestand importeren Bij het importeren van een gedcom bestand in de huidige applicatie is het enkel mogelijk om een stamboom te importeren waarvan alle relaties aan elkaar geknoopt zijn. Een uitbreiding zou ervoor kunnen zorgen dat meerdere stambomen opgesteld worden uit eenzelfde bestand.
3. Toevoegen van een gedcom bestand aan een stamboom

## 9.2 Stambomen

1. Printen van een stamboom
2. Personen verplaatsen tussen stambomen
3. Samenvoegen van stambomen
4. In 3D wandelen door een stamboom
5. Exporteren en importeren van stambomen in bepaalde formaten zoals pdf , word , excel ,scv ..

## 9.3 Admin

In de huidige applicatie bestaat het admin gedeelte enkel uit een Useroverviewpanel die een overzicht bevat van alle gebruikers. Deze gebruikers kunnen door de admin geblokkeerd of gewijzigd worden en is er een mogelijkheid om in te loggen als een geselecteerde gebruiker om de gegevens van hun stamboom te wijzigen. Het inloggen als een gebruiker door de admin heeft als voordeel dat de administrator een duidelijk beeld krijgt van zijn stambomen waarbij er niet te veel gegevens uit de databank opgehaald moeten worden. Een uitbreidingsmogelijkheid zou zijn om een personoverviewpanel te voorzien om direct gegevens te wijzigingen in een lijst van personen. Tijdens sprint 2 is er een prototype ontwikkelt die een lijst bevat van alle personen maar qua performantie was dit ver van ideaal. Om het personoverviewpanel te verbeteren zouden de personen moeten verspreid worden door eerst een lijst te voorzien van de gebruikers , gevolgd door hun stambomen en dan pas een lijst van personen die aan de hand van paginerings opgehaald worden uit de databank.

Naast het personoverviewpanel zou de administrator de mogelijkheid moeten krijgen om volgende opties uit te voeren :

1. Het importeren van gegevens uit andere databanken die gebruik maken van de applicatie
2. Een statistisch overzicht van de verschillende acties in de applicatie

Naast uitbreidingsmogelijkheden zijn er ook nog enkele verbeteringspunten :



## CHAPTER 9. VERBETERPUNTEN EN UITBREIDINGSMOGELIJKHEIDEN

1. Bij het wijzigen van de user in de useroverviewpanel worden alle users terug opgehaald na een wijziging , het zou efficiënter zijn om enkel de gewijzigde persoon terug op te halen.

# Chapter 10

## Nabeschouwing en besluit

Dit vakoverschrijdend project is na drie sprints een afgewerkt project dat alle features bevat die met de klant werden besproken. Het project bestaat uit verschillende onderdelen die elk hun taak uitvoeren binnen het geheel.

De MySQL-database is de backbone van de gehele applicatie die alle data bevat. Met de perl-scripts kan deze worden opgevuld met dummy-data om zo een goede testomgeving te creëren. De Api zorgt voor de verwerking van data en een goede communicatie tussen de database en StambomenClient. De StambomenClient is de basis voor zowel de Desktop- als de WebApplicatie. Deze staat in voor alles wat de Desktop- en de WebApplicatie gemeenschappelijk hebben om zo een efficiëntere en performantere oplossing aan te bieden. De Desktop- en de WebApplicatie zijn de twee applicatie waarmee de gebruiker in contact komt. De DesktopApplicatie stelt de gebruiker in staat stambomen aan te maken en deze te beheren. De stambomen bevatten personen die elk hun eigen gegevens hebben die kunnen worden aangepast. De WebApplicatie biedt de gebruiker een eenvoudig en sociaal platform aan waarop de gemaakte stambomen kunnen worden weergegeven en gedeeld onder vrienden. De facebook-integratie maakt bepaalde taken zoals inloggen en vrienden zoeken nog eenvoudiger en helpt de gebruiker bij het delen van zijn stambomen.

Het vakoverschrijdend project is een uitgebreid project geworden dat alle aspecten bevat die van ons werden verwacht. Er zijn zoals besproken uiteraard nog enkele kleine foutjes die weggewerkt zouden kunnen worden of puntjes die voor verbetering vatbaar zijn. Verder kan het project nog op bepaalde vlakken worden uitgebreid om de user-experience te verbeteren.

In de huidige staat zou ons project geïmplementeerd en gebruikt kunnen worden op grote schaal. Dit in combinatie met het feit dat ons project voldoet aan de eisen van de klant kan er worden gezegd dat ons vakoverschrijdend project een afgewerkt project is.

# List of Figures

6.1 SLF4J Logging . . . . .	41
-----------------------------	----

## List of Tables

## **Appendix A**

## **Appendix**

## A.1 Opdracht sprint 1

# STAMBOMEN

### DOEL

Door van elke persoon de (biologische) vader en moeder bij te houden, worden boomstructuren opgebouwd. Hierbij kan het voorkomen dat deze ouders onbekend zijn, hetzij omdat niet geweten is wie zij zijn, hetzij omdat de gebruiker de gegevens nog niet heeft ingevoerd.

Verder houdt het systeem van elke persoon de voor- en familienaam, het geslacht, de geboortedatum en de geboorteplaats bij. De geboortedatum en -plaats kunnen daarbij ontbreken.

Elke gebruiker kan verschillende stambomen beheren. Dit betekent dat hij/zij als enige de structuur en persoonsgegevens kan wijzigen; dit is het *bewerken* van de stamboom. Bij het bewerken worden enkel de in de huidige stamboom aanwezige personen weergegeven.

Bij het *consulteren* daarentegen voegt het systeem alle beschikbare stamboomgegevens samen, om zo tot een zo volledig mogelijk overzicht te komen. Indien een persoon met identieke gegevens in meerdere bomen voorkomt, wordt daarbij aangenomen dat het om dezelfde persoon gaat. Er wordt gestart van de stamboom van de huidige gebruiker; dit noemen we de *referentiepersoon*. Men kan een andere referentiepersoon selecteren door deze aan te klikken in de stamboom.

Een gebruiker kan bij elke stamboom aangeven of hij deze wil *delen* met andere gebruikers van de applicatie. Is dit niet het geval, dan wordt de inhoud ervan verborgen bij het consulteren. Er kan worden gedeeld met iedereen, met niemand, of enkel met de vrienden van de gebruiker. Voor deze laatste mogelijkheid kunnen gebruikers vriendenlijsten aanleggen.

Naast de klassieke boomweergave is er ook de *teletijdmachine*, een wereldkaart met tijdsaanduiding. Hierop worden de geboorteplaatsen gevisualiseerd van de voorouders en het nageslacht van een persoon naar keuze. Men kiest bovendien een datum; enkel de familieleden die op die datum in leven waren, worden opgenomen in de visualisatie.

De applicatie wordt deels als webapplicatie, deels als desktopapplicatie aangeboden. Overleg daarom met de klant onder andere over welke componenten zich waar bevinden.

De lijst met features is een voorstel. Ken er story points aan toe en maak een planning op om met de klant tot een consensus te komen over het contract voor sprint 1. Implementeer enkel het gevraagde.

## **A.2 Opdracht sprint 1**

### **SPRINT 1**

1. De gebruiker maakt een account aan.
2. De gebruiker logt in op de webapplicatie.
3. De gebruiker logt in op de desktopapplicatie.
4. De gebruiker maakt een nieuwe stamboom aan.
5. De gebruiker opent een bestaande stamboom.
6. De gebruiker voegt een persoon toe aan de huidige stamboom.
7. De gebruiker wijzigt de gegevens van een persoon in de huidige stamboom.
8. De gebruiker verplaatst een persoon in de huidige stamboom.
9. De gebruiker verwijdert een persoon uit de huidige stamboom.
10. De gebruiker importeert een GEDCOM-bestand.
11. De gebruiker verstuurt een vriendschapsverzoek.
12. De gebruiker aanvaardt een vriendschapsverzoek.
13. De gebruiker weigert een vriendschapsverzoek.
14. De gebruiker bekijkt zijn vriendenlijst.
15. De gebruiker verwijdert iemand uit zijn vriendenlijst.
16. De gebruiker consulteert zijn samengestelde stamboom.
17. De gebruiker bekijkt de gegevens van een persoon in een stamboom.
18. De gebruiker wijzigt de referentiepersoon van een stamboom.
19. De gebruiker kiest de referentiepersoon voor de teletijdmachine.
20. De gebruiker kiest de datum voor de teletijdmachine.
21. De teletijdmachine markeert de geboorteplaatsen van de familieleden die op de ingevoerde datum in leven waren.
22. Het systeem bevat een grote hoeveelheid stamboomgegevens.



## A.3 Opdracht sprint 2

# STAMBOMEN

### DOEL

Net als bij sprint 1 is de lijst met features een voorstel. Voeg deze samen met de eventuele features die in sprint 1 nog niet (volledig) werden gerealiseerd. Deel opnieuw de features op in taken en ken er story points aan toe. Stem af met de klant om zo snel mogelijk tot een definitief contract te komen.

### SPRINT 2

1. De gebruiker wijzigt de weergavetaal.
2. Het systeem houdt een overzicht bij van relevante gebeurtenissen.
3. De gebruiker voegt een foto toe aan een persoon.
4. De gebruiker verwijdert een foto van een persoon.
5. De gebruiker zoomt in op de stamboom. Naarmate meer ruimte beschikbaar is, worden meer gegevens weergegeven bij een persoon.
6. De gebruiker koppelt een persoon aan zijn/haar Facebookaccount.
7. De gebruiker doorzoekt de voor hem beschikbare stambomen.
8. De gebruiker geeft aan dat twee personen uit verschillende stambomen aan elkaar gelijk zijn.
9. Het systeem detecteert gelijkaardige personen uit verschillende stambomen en meldt de gelijkenissen aan de betrokken gebruikers.
10. De gebruiker bevestigt dat twee personen aan elkaar gelijk zijn.
11. De gebruiker geeft aan dat twee personen niet aan elkaar gelijk zijn.
12. De gebruiker doorzoekt de openbare gebruikerslijst.
13. De gebruiker bekijkt een openbaar profiel.
14. De gebruiker maakt zijn/haar profiel openbaar.

## **A.4 Opdracht sprint 2**

15. De gebruiker geeft aan of een stamboom wordt gedeeld met alle gebruikers van de applicatie, enkel met zijn/haar vrienden, of met niemand.
16. De gebruiker registreert zich met zijn/haar Facebookaccount.
17. De gebruiker koppelt zijn/haar account aan zijn/haar persoonlijke Facebookaccount.
18. De gebruiker logt in met Facebook.
19. De gebruiker deelt stamboomgegevens op Facebook.
20. De gebruiker voegt een Facebookvriend toe aan zijn/haar vriendenlijst.
21. De gebruiker bekijkt een overzicht van de activiteiten van zijn/haar vrienden.
22. De moderator wijzigt de gegevens van een persoon.
23. De moderator wijzigt het profiel van een gebruiker.
24. De moderator blokkeert tijdelijk het account van een gebruiker.
25. De gebruiker bekijkt een animatie van de teletijdmachine.
26. De gebruiker vertraagt of versnelt de teletijdmachine.
27. De gebruiker pauzeert de teletijdmachine.
28. De gebruiker navigeert naar een tijdstip met de teletijdmachine.
29. De gebruiker wijzigt het thema van de applicatie.
30. De ontwikkelaar wijzigt de huisstijl van de applicatie.