# **Final Group Dossier**
# Fontys Module Management System (FMMS)

Nils Nieuwenhuis, Sjoerd Brauer, Loek Ehren, Tobias Derksen

*Fontys School of Technology and Logistics*
*Informatics*
*Module Software Factory (SOFA)*

Venlo, 16th January 2018

# Contents

# Chapter 1

# Project Plan

# Project Plan

Sjoerd Brauer, Tobias Derksen, Loek Ehren, Nils Nieuwenhuis

September 14, 2017

# Contents

# Chapter 1

# Introduction

This chapter is about general information regarding the Sofa Project "Fontys Module Management System". It includes background information and a small description of the customer and the organisation for which this project is done for.

## 1.1 Background information

The project belongs to the module SOFA of the seventh semester in the study course Informatics. The project team consists of one business informatic and three software engineering students. A real customer is involved in this project and the main goal of the project is to satisfy the customer and to act as a real company.

## 1.2 Context

The Project is for the Informatics department of the Fontys University of Applied Sciences in Venlo. The project period is about five months. The overall purpose of this SOFA project is to apply the knowledge and skills gained during the studies in the course Informatics. As students from both study directions business informatics and software engineering are involved in this project, subject-specific knowledge will be shared within the team and applied accordingly.

# Chapter 2

# Description

In this chapter is a description of the assignment for this project followed by an explanation of the problem which is to be solved. Finally the goals of the assignment will be set.

## 2.1 Assignment

The assignment of this project is to develop a system to manage module descriptions within the Informatics department of the Fontys University of Applied Science in Venlo.

## 2.2 Problem

The customer wants a product to easily create, modify and publish module descriptions for the Software Engineering and Business Informatics department of "Fontys Hogeschool Techniek en Logistiek" in the future.

Today there is no standardized way of creating and changing modules and their descriptions.

During the project our company will develop a software product to solve this problem.

## 2.3 Goals

- Delivering a high quality but not necessarily finished product
- Delivering proper documentation

- Delivering an extensible product

## 2.4 Approach

To assure a high quality product we stick to the Agile development framework Scrum during the project. A good documentation is achieved with our "SOFA Quality Guidelines" and "Software Requirements Specification" defined at the beginning of the project. Careful consideration will have to be taking into account when designing parts of the system to ensure the end-product is extensible and easily understandable for possible other project teams who could continue with this project.

# Chapter 3

# Project

In this chapter the project will be described in detail. It starts with a listing of the stakeholders. After that an explanation of the responsibilities will follow. Furthermore a detailed project scope, planning and organisation of this project is described in this chapter.

## 3.1 Roles and Responsibilities

### 3.1.1 Stakeholders

| **Role** | **Name** | **Interest** | **Influence** |
|---|---|---|---|
| Customer | Van den Ham, Richard | High | High |
| Users of the Product | IT Teachers | High | Medium |
| Users of the Product | Students | Medium | Low |
| Students who want to pass | SOFA group | High | High |
| Coach | Jacobs, Jan | High | Medium |

Table 3.1: Stakeholders

### 3.1.2 Team

#### Project Manager: Nils Nieuwenhuis

The project manager is responsible for the planning and will keep in touch with the customer due to the limited time. The project manager will write and keep the project management plan up to date. He will also keep a tight watch on the available resources. The project manager is responsible for the agenda of each meeting.

**Quality Manager: Loek Ehren**

The reviewing process of the project deliverables is managed by the quality manager. He has the responsibility to check that nothing leaves the team without testing and for that he writes a quality management plan and keeps it up to date. He also archives the reports produced in the test processes.

**Configuration Manager: Sjoerd Brauer**

The configuration manager has to document the use and / or modification of any hardware related configuration item as well as the use of the software and document related configuration items. Everything will be documented in the configuration management plan by the configuration manager.

## 3.2 Project scope

This section is about the scope of the project. It focuses on its functions and data, whereas the deliverables are defined in section 3.3.1 as project products.

The target of the project is to create a system that makes the information collection and deployment of the module informations easier. The most important feature of the system is to display the already available Information about modules and the consistency of this information. Especially their learning goals connected to the hoger beroepsonderwijs.

### 3.2.1 Functionality

The functionality is described in the Software Requirements Specification document in detail, for more information please refer to this document.

### 3.2.2 Data

| | |
|---|---|
| **Module** | The name of the module |
| **Semester** | The semster where the module takes place |
| **Credits** | The credits which can be achieved if the module is successfully completed |
| **Valid as of** | The date were the descriptions will be valid |
| **Author** | The creator of the description |
| **Description** | The description of the module |
| **Learning Goals** | The learning goals wich will be achieved during the module |
| **Competence Profile** | The competence profile a student has after finishing the module |
| **Module Assessment** | The assessmentforms of the module |
| **Prior Knowledge** | The modules which have to be completed to participate in the module |
| **Additional Information** | Additional information to the module |

## 3.3 Planning

In this section the planning will take place and in addition to that it begins with the milestones. After that the project products will be defined. The planning of the project will be performed based on the Scrum framework.

### 3.3.1 Milestones

By the end of week 4, the project team is familiar with the project and can begin working on the project in a Scrum fashion.

A Scrum sprint will last one week, after which the completed backlog items will be reviewed and the following sprint will be set up. In this way, each week will be its own milestone.

By the end of the project, the mandatory deliverables must be completed.

### 3.3.2 Project products

- Project plan
- Quality plan
- Configuration management plan
- Documentation
- Prototype
- User manual

## 3.4 Organisation

### 3.4.1 Communication plan

| **Communication medium** | **Stakeholders** | **Frequency** | **Deliverable** |
|---|---|---|---|
| WhatsApp | Project team | If necessary | - |
| Customermeeting | Project team , Customer | Weekly | Meeting minutes |
| Coachmeeting | Project team, Coach | Weekly | Meeting minutes |
| Sprint planning meeting | Project team | At the bebinning of a sprint | Sprint Backlog |
| Daily Scrum | Project team | Daily | - |
| Sprint review | Project team | At the end of a sprint | - |
| Sprint retrospective | Project team | At then end of a sprint | - |

Table 3.2: Communication plan

### 3.4.2 Quality assurance

The quality assurance will be dealt with the ”SOFA Quality Guideline” created and maintained by the Quality Manager, for more information please refer to this document.

### 3.4.3 Risk Management

| # | Risk description | Impact | Owner | Likelihood of occurence | Severity of effects | Status | Mitigation action | Reaction time frame |
|---|---|---|---|---|---|---|---|---|
| 1 | A stakeholder is not present during a meeting | Decisions can't be made | Project manager | Low | High | not yet occured | Veto right is only valid if the stakeholder is present | within 1 day |
| 2 | Changing requirements | Probably changes in already finished Product features | Customer | High | High | not yet occured | Requirements have to be defined in a signed document | within 1 day |
| 3 | At the end of a sprint there are unfinished backlogs | In the next sprint is more work to do | Project team | High | Low | not yet occured | The Backlog items must be graded at the start of a sprint and unfinished Backlogitems will be moved to the next sprint | within 1 day |
| 4 | Ressources are not available | Changes the way of developement | Project team | Medium | High | not yet occured | The Resources have to be defined with the customer. | within 1 day |

Table 3.3: Riskregister

# Chapter 2

# Quality Plan

# SOFA Quality Guideline

Loek Ehren

January 14, 2018

| **Date** | **Version** |
|---|---|
| 11-09-2017 | 1.0 |
| 14-09-2017 | 2.0 |

## Contents

## 1 Documents

This section outlines the quality standards and processes for all documents produced during the SOFA project.

1. All documents shall be written in LaTeX.
2. Every document's source and rendered result shall be committed and versioned.
3. All documents shall be reviewed at least once by another person either inside or outside of the team using the template found at Appendix A Review Template.
4. The results of document reviews shall be stored in the correct folder in the `25reviews/` directory with the following naming convention: **YYDDMM_firstname_documentversion***
5. An improved version of a document shall be reviewed once more to guarantee feedback has been processed and passes standards.
6. If a document does not pass a review, this document shall be improved again.
7. Documents to be reviewed will be added to the week's backlog.

*So the first review of a project plan will have this path:
`25reviews/project_plan/20170911_loek_1.0.pdf`

## 2 Software

This section outlines the quality standards and processes for software development during the SOFA project.

### 2.1 Code

1. All code committed shall be committed alongside relevant testfiles.
2. All code relevant to a new feature, fix or any other change will be developed in a new branch.
3. Any branch that is to be merged into the master branch will be submitted in a pull request and reviewed by at least one person. Code reviews and any CI tool must also pass before merging.
4. All code shall be 100% tested at all time. If it is not, changes will not be accepted into the master branch.
5. Tests shall be useful, concise and descriptive. Don't test only one scenario, but every scenario.
6. All code shall be documented clearly, either by simple self documenting code or written documentation.

### 2.2 Commits

1. Commit messages will have a descriptive header describing the content of the commit.
2. A commit message's body could explain more about the change. Why something is changed, what it does etcetera.
3. Commits will remain small and relevant to only one change. If a commit changes more than one thing, it is too big!

## 3 Others

This section outlines standards for miscellaneous items to be standardized.

- Everyone shall track their Configuration Item in the CI table in `40configuration_management configuration_item_table` folder on GitHub.

## 4 Appendix A Review Template

**Date of Review**
**Document Title**
**Document Date**
**(Optional) Commit ID**
**Reviewer Name**
**Reviewer Comments**

# Chapter 3

# Configuration Management Plan

# Configuration management plan

Sjoerd Brauer

11-9-2017

# Contents

# Chapter 1

# Strategy

This chapter has a high level overview of what can be expected of configuration management.

## 1.1 Scope

The configuration management will include all CI's of the projects. The CI are all things beloning to the project. Documents, hardware etcetera. CM also includes how to deliver what CI's to the stakeholders, the frequency and in what way. CM manager will also document how to use hardware/software related to the project. CI's used but not owned by the project will not be documented.

## 1.2 Responsibility

The CM manager is responsible for maintaining the structure of the CI table(CIT) and this document. Every owner of a CI is responsible for that CI and the documentation of that CI in the CIT.

# Chapter 2

# Procedures

This chapter describes the procedures the group follows for configuration management.

## 2.1 CI management

When new CI's are introduced the owner of the CI must make an entry in the CIT and keep this CI item updated at all times. The configuration manager will pulse group members if they do not update their CI.

When delivering a product the CM manager is responsible for the delivery to the stakeholder. This includes the product and documentation

# Chapter 3

# Deliverables

This chapter mentions the deliverables. More information about the deliverables can be found in the CIT. Like whether it is finished or not. This chapter will be continuously updated.

## 3.1 Artifacts

Project plan. Planning. Quality Management document. Database design. User Storied. Domain model.

# Chapter 4

# Requirements Document

# ”Fontys Module Management System” Software Requirements Specification

Loek Ehren, Nils Nieuwenhuis, Tobias Derksen, Sjoerd Brauer

Fontys Hogeschool Techniek en Logistiek, January 14, 2018

| **Date** | **Version** |
|---|---|
| 11-09-2017 | 1.0 |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This document specifies all the requirements of the SOFA project during the seventh semester of the Software Engineering course at Fontys Hogeschool Techniek en Logistiek. The project is named Fontys Module Management System (FMMS).

## 1.1 Purpose

The purpose of this project is to streamline the process of changing module descriptions for the classes in the Software Engineering course, offering one centralized source of truth for the information contained in the module descriptions and the Onderwijs Examen Regeling (OER), and a controlled way of changing the descriptions if need be.

## 1.2 Scope

The scope of this document contains all the requirements of the system to be created.

## 1.3 Overview

This document is divided into three chapters. Chapter one will introduce the document and its purpose. The second chapter will give an overview of the functionality of the system and other interactions. The third chapter will specify all detailed requirements gathered which shall be implemented.

# Chapter 2

# Overall Description

This chapter will describe the general factors that will influence the product and its requirements.

## 2.1 Product Perspective

The product will be a software system that interacts with a database and a frontend to deliver and receive data.

Input some general diagram or description of the system's architecture

Describe interfaces

## 2.2 Product Functions

The general major functions of this system will be

- CRUD operations on module data
- Generating and releasing module descriptions and the OER
- Logging changes

## 2.3 Users

The users of the system will be

- Teachers
- Department Heads
- Curriculum Owner

## 2.4 External Interfaces

Any external software interfaces are listed here

Table 2.1: User Interaction Interface

| | |
|---|---|
| **Description** | User interface |
| **Purpose** | Display data to the user and allow modification |
| **Source of input** | User interaction |
| **Destination of output** | Backend system |

Table 2.2: Database Interface

| | |
|---|---|
| **Description** | Database connection to a PostgreSQL database |
| **Purpose** | Store and retrieve data from and to the system |
| **Source of input** | Backend system |
| **Destination of output** | Backend system |

Table 2.3: PDF Generator Interface

| | |
|---|---|
| **Description** | PDF Generator |
| **Purpose** | Generate a PDF file |
| **Source of input** | Backend system |
| **Destination of output** | Backend system |

# Chapter 3

# Requirements

This chapter will detail all non functional and functional requirements of the system.

## 3.1 Nonfunctional Requirements

This section will list all nonfunctional requirements of the system.

1. The system should be a web application
2. The system should be able to run in a Docker container
3. The environment should easily connect to different versions of the database
4. The system shall be in English only
5. The system should be available outside office hours
6. The system should be available at normal rates
7. The system should be secure
8. The system should be easily extendable for future developers

## 3.2 User Stories

# User stories

Sjoerd Brauer

11-9-2017

| | |
|---|---|
| Name | Updating the curriculum |
| ID | US_003 |
| Actors | Curriculum owner |
| Story | The curriculum owner changes the curriculum |

| | |
|---|---|
| Name | Publishing the curriculum |
| ID | US_004 |
| Actors | Curriculum owner |
| Story | The curriculum owner releases the Curriculum |

| | |
|---|---|
| Name | Login |
| ID | US_006 |
| Actors | 1. Teacher<br>2. Department head<br>3. Curriculum owner |
| Story | A system user logs in on the users account. |

| | |
|---|---|
| Name | Logout |
| ID | US_007 |
| Actors | 1. Teacher<br>2. Department head<br>3. Curriculum owner |
| Story | A user logs out. |

| | |
|---|---|
| Name | Changing permission |
| ID | US_008 |
| Actors | Department head |
| Story | Department head changes an users permission |

| | |
|---|---|
| Name | Adding users |
| ID | US_009 |
| Actors | Department head |
| Story | Department head adds users to the system. |

| | |
|---|---|
| Name | Assigning courses |
| ID | US_011 |
| Actors | Department head |
| Story | Department head changes teachers to courses |

| | |
|---|---|
| Name | Inputting guidelines |
| ID | US_012 |
| Actors | Department head |
| Story | Department head inputs the guidelines for the module description. |

| | |
|---|---|
| Name | Create a module |
| ID | US_013 |
| Actors | Curriculum owner |
| Story | Curriculum owner creates a module. |

| | |
|---|---|
| Name | Modifying a module |
| ID | US_014 |
| Actors | Teacher |
| Story | Teacher modifies a module. |

| | |
|---|---|
| Name | Checking history |
| ID | US_015 |
| Actors | 1. Teacher<br>2. Department head<br>3. Curriculum owner |
| Story | A system user looks into the history of a module description |

| | |
|---|---|
| Name | Track history p2 |
| ID | US_016 |
| Actors | 1. Teacher<br>2. Department head<br>3. Curriculum owner |
| Story | A system user looks into the history of a module |

| | |
|---|---|
| Name | Display information |
| ID | US_017 |
| Actors | 1. Teacher<br>2. Department head<br>3. Curriculum owner |
| Story | The user displays a module with the corresponding HBO-I matrix. |

| | |
|---|---|
| Name | Insert progress data |
| ID | US_018 |
| Actors | Teacher |
| Story | A teacher inserts a students progress data into the system |

| | |
|---|---|
| Name | Change progress data |
| ID | US_018 |
| Actors | Teacher |
| Story | A teacher changes a students progress data in the system |

| | |
|---|---|
| Name | Update exam information |
| ID | US_020 |
| Actors | Teacher |
| Story | A teacher changes a module's exam information |

| | |
|---|---|
| Name | Create exam information |
| ID | US_021 |
| Actors | Teacher |
| Story | A teacher creates a module's exam information |

| | |
|---|---|
| Name | Add competences |
| ID | US_022 |
| Actors | Teacher |
| Story | A teacher adds competences to the system. |

| | |
|---|---|
| Name | Remove competences |
| ID | US_023 |
| Actors | Teacher |
| Story | A teacher removes competences from the system. |

| | |
|---|---|
| Name | Update competences |
| ID | US_024 |
| Actors | Teacher |
| Story | A teacher updates competences in the system. |

| | |
|---|---|
| Name | Adding learning goals |
| ID | US_025 |
| Actors | Teacher |
| Story | A teacher adds learning goals to a module. |

| | |
|---|---|
| Name | Updating learning goals |
| ID | US_026 |
| Actors | Teacher |
| Story | Teacher updates learning goals in the system. |

| | |
|---|---|
| Name | Creating learning goals |
| ID | US_027 |
| Actors | Teacher |
| Story | A teacher creates learning goals in the system. |

## 3.3 Use Cases

## 3.4 Mockups

# Chapter 5

# User Stories

## 5.1 Document

# User stories

Loek Ehren

11-9-2017

| | |
|---|---|
| Name | View learning goals |
| ID | US_001 |
| Actors | Teacher |
| Story | I want to see the learning goals of a module. |
| Completed in | Sprint 1 |

| | |
|---|---|
| Name | View module information |
| ID | US_002 |
| Actors | Teacher |
| Story | I want to see all the information of a module. |
| Completed in | Sprint 1 |

| | |
|---|---|
| Name | Filter on curricula |
| ID | US_003 |
| Actors | Teacher |
| Story | I want to filter modules on their curricula. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | See all modules of curriculum |
| ID | US_004 |
| Actors | Teacher |
| Story | I want to see all modules of a curriculum. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | Filter on qualifications |
| ID | US_005 |
| Actors | Teacher |
| Story | I want to filter modules on their competences. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | View curriculum full name |
| ID | US_006 |
| Actors | Teacher |
| Story | I want to see the full name of a curriculum. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | View module full name |
| ID | US_007 |
| Actors | Teacher |
| Story | I want to see the real module name in the module overview. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | View semester information |
| ID | US_008 |
| Actors | Teacher |
| Story | I want to see all information of a semester. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | Order projects |
| ID | US_009 |
| Actors | Teacher |
| Story | I want to see projects ordered to the end of the Semester. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | Continuous deployment |
| ID | US_010 |
| Actors | Customer |
| Story | I want an up to date app running. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | Multi line text support |
| ID | US_011 |
| Actors | Teacher |
| Story | I want to see multi line text for introduction. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | Hide empty fields |
| ID | US_012 |
| Actors | Teacher |
| Story | I don't want to see empty fields of a module description. |
| Completed in | Sprint 2 |

| | |
|---|---|
| Name | Save and edit basic information |
| ID | US_013 |
| Actors | Teacher |
| Story | I want to edit and save basic plain text information of a module. |
| Completed in | Sprint 3 |

| | |
|---|---|
| Name | See teaching materials |
| ID | US_014 |
| Actors | Teacher |
| Story | I want to see the teaching materials of a module. |
| Completed in | Sprint 3 |

| | |
|---|---|
| Name | View grading information |
| ID | US_015 |
| Actors | Teacher |
| Story | I want to see a module's grading and assessment information. |
| Completed in | Sprint 3 |

| | |
|---|---|
| Name | Edit prior knowledge |
| ID | US_016 |
| Actors | Teacher |
| Story | I want to edit and save a module's linked modules. |
| Completed in | Sprint 4 |

| | |
|---|---|
| Name | Edit and save grading information |
| ID | US_017 |
| Actors | Teacher |
| Story | I want to edit and save a module's grading information |
| Completed in | Sprint 4 |

| | |
|---|---|
| Name | Edit and save learning goals |
| ID | US_018 |
| Actors | Teacher |
| Story | I want to edit and save a module's learning goals. |
| Completed in | Sprint 4 |

| | |
|---|---|
| Name | Generate PDF |
| ID | US_019 |
| Actors | Teacher |
| Story | As teacher I want to generate a PDF of a module's description. |
| Completed in | Sprint 5 |

**Following are all the user stories that were identified but not completed.**

| | |
|---|---|
| Name | Login |
| ID | US_0020 |
| Actors | User |
| Story | I want to be able to login to the system using Fontys credentials. |

| | |
|---|---|
| Name | Logout |
| ID | US_021 |
| Actors | Logged in user |
| Story | I want to be able to logout. |

| | |
|---|---|
| Name | See logs |
| ID | US_022 |
| Actors | Admin |
| Story | I want to be able to see the changelog of a module. |

| | |
|---|---|
| Name | Approve publish |
| ID | US_023 |
| Actors | Department head |
| Story | I want to be able to approve a module to be published. |

| | |
|---|---|
| Name | Notify changes |
| ID | US_024 |
| Actors | Department head |
| Story | I want to be notified of critical changes in a module. |

| | |
|---|---|
| Name | Prevent modification after publish |
| ID | US_025 |
| Actors | Department head |
| Story | I want to prevent the modification of a module after its start date. |

| | |
|---|---|
| Name | See different module versions |
| ID | US_026 |
| Actors | Teacher |
| Story | I want to see different versions of a curriculum. |

## 5.2 Diagram

# Chapter 6

# Software Architecture Document

# Software architecture

## 1. Interface specifications

This document outlines the specifications of the back-end application. The back-end application has a RESTful interface with which the front-end communicates with.
The data format used will be JSON.
At the beginning of the project we choose that when communicated with the backend the least amount of endpoints should be called. That's why a lot of custom object interfaces can be found in front end.

The JSON objects can be found in the json_objects.ods file in the 60software_architecture\backend_interface directory.

| **Complete overview of back-end endpoints.** | |
|---|---|
| **Uri path with resource** | **description** |
| /curricula | Returns an array of curriculum |
| /curriculum/:curriculum id/module/:module_code | Returns a module object |
| /curriculum/:curriculum id/semesters | Returns an array of semester |
| /curriculum/:curriculum_id/semester/:semester_number | Returns a complete_semester object |
| /qualications | Returns a filter_qualifications object |
| curriculum/:curriculum_id/architecturallayer/:id/activity/:id | returns a complete qualications overview semester object |
| /module/:module_code | Accepts a editable_module_input or returns an editable_module_output. |
| /curriculum/:curriculum id/module/:module_code/pdf | returns a modules PDF |

## 2. Endpoint definitions

This section describes the urls more specically with functionality, the JSON format and HTTP codes. The dataobjects are described in json objects.pdf in this repository.

| **Get data for editing module** | |
|---|---|
| URL | /module/:module_code |
| Method | GET |
| Returns | Returns a filled editable_module_output |
| Returns code | 200 OK<br>404 NOT FOUND |

| **Post editing module** | |
|---|---|
| URL | /module/:module_code |
| Method | POST |
| Returns | Gets a filled editable_module_input |
| Returns code | 200 OK<br>404 NOT FOUND |

| **Complete semester** | |
|---|---|
| URL | /curriculum/:curriculum_id/semester/:semester_number |
| Method | GET |
| Returns | Returns a filled complete_semester object |
| Returns code | 200 OK<br>404 NOT FOUND |

| **Filter of qualifications** | |
|---|---|
| URL | curriculum/:curriculum_id/architecturallayer/:id/activity/:id |
| Method | GET |
| Returns | Returns a filter_qualifications object |
| Returns code | 200 OK<br>404 NOT FOUND |

| complete module | |
|---|---|
| URL | /curriculum/:curriculum id/module/:module_code |
| Method | GET |
| Returns | Returns a filled_module object |
| Returns code | 200 OK<br>404 NOT FOUND |

| Semesters of a curriculum | |
|---|---|
| URL | /curriculum/:curriculum id/semesters |
| Method | GET |
| Returns | Returns an array of semester objects |
| Returns code | 200 OK<br>404 NOT FOUND |

| Curricula of a student program | |
|---|---|
| URL | /curricula |
| Method | GET |
| Returns | Returns an array of curriculum objects |
| Returns code | 200 OK<br>404 NOT FOUND |

# Chapter 7

# Database Documentation

# Documentation of FMMS Database Structure

Software Factory Group 1

Tobias Derksen

*Fontys Venlo Techniek en Logistiek*

*Informatics*

Venlo, January 15, 2018

## Contents

# 1 Introduction

## 1.1 Overview

This document provides an overview over the database structure of the Fontys Module Management System (FMMS). The target group of this document are informatics students of at least semester 7 or persons with a similar or better understanding of information technologies and databases.

The database management system (DBMS) which is used is PostgreSQL, which provides a full set up up-to-date database feature in an open source software product. Initially the database was created by HVD but has been improved during the project.

## 1.2 Conventions

All database names follow a special convention which is described in this section.

**Table Names** All tables are named after the entity they represent. The name is always singular. If the table represents a many-to-many relationship, the table name contains the two tables which are connected separated by an underscores. The order of the tables is not regulated, but normally there is a natural direction inside the relationship.

**Primary Keys** All tables contain a column named "id" with the database type "SERIAL" which implies integer. This column is the primary key of each table. If there are business keys or natural unique combinations, they are created separately keeping "id" as the only primary key column to every table.

**Foreign Keys** All columns which contain a foreign key relationship are named using a special convention. They contain a regular name, which is mostly the table name they reference, followed by an underscore and then the column name of the foreign table which they reference. Therefore, all columns which contain an underscore represent a relationship to another table and hold a foreign key.

**Data Types** For each column the data type has been chosen which fits best to the intended data. There are two exceptions: first, all primary key columns have the data type "SERIAL" which implies an "INTEGER" (as described before), second all columns which contains character have the data type "TEXT". Because there is no performance or data storage difference in PostgreSQL database between "TEXT" and "CHARACTER VARYING" we do not need to keep track of max string length when we use "TEXT" by default.

## 2 Database Structure

This section documents the individual tables and their purpose. The table names aren't always enough to understand the purpose of the table. Also this sections provides important information about relationships between tables and specific constraints.

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | activity |
| **Description** | This table contains all activities as they are defined in the HBO-I qualifications document. |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | architecturallayer |
| **Description** | This table contains all architectural layers as they are defined in the HBO-I qualifications document. |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | curriculum |
| **Description** | This table contains different curricula. Curricula are a set of modules which belongs to a specific study program. Basically, curricula represent a version of a study program which can change over time. A curriculum has a curriculum owner and belongs to a specific department. |
| **Connected Tables** | employee, department |

| Table | |
|---|---|
| **Schema** | study |
| **Name** | department |
| **Description** | This table contain the different departments which manage their modules inside the system. |

| Table | |
|---|---|
| **Schema** | study |
| **Name** | employee |
| **Description** | This table contains the employee, meaning the lecturers of the departments. |

| Table | |
|---|---|
| **Schema** | study |
| **Name** | employee_department |
| **Description** | This table connects employee with certain departments. In theory, an employee can work for multiple departments. |
| **Connected Tables** | employee, department |

| Table | |
|---|---|
| **Schema** | study |
| **Name** | learninggoal |
| **Description** | This table contains all learning goals. Learning goals consists of a sequence number, a weight and a description. Every learning goal is connected to exactly one module. |
| **Connected Tables** | module |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | learninggoal_qualification |
| **Description** | This table connects learning goals with concrete qualifications. A learning goal can be connected with multiple qualifications. These connections shows how the learning goal contributes the to students skills. |
| **Connected Tables** | learninggoal, qualification |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | levelofskill |
| **Description** | This table contains the different level of skills which can be reached during study. A skill is always a combination of an activity and an architectural layer. The possible levels of skills are defined in the HBO-I document. |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | module |
| **Description** | |
| **Connected Tables** | |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | module_employee |
| **Description** | This table contains the many-to-many relationship between modules and employees. A module can be taught by multiple employees (and usually is), and an employee can teach multiple modules. |
| **Connected Tables** | module, employee |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | module_profile |
| **Description** | |
| **Connected Tables** | module, profile |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | moduleassessment |
| **Description** | This table contains information about exams and assessments. Each instance has a globally unique assessment code which can also be found in Progress. |
| **Connected Tables** | module |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | moduledependency |
| **Description** | This table represents dependencies between modules. There are three types of dependencies: PRIOR, CONCURRENT and MANDATORY. Prior module are modules which are logically before the actual module. Concurrent modules are taught in the same semester and share some topics. Mandatory modules have to be passed before you can start with the current module. This table allows to generate a flow chart which displays the dependencies inside a study program. |
| **Connected Tables** | module |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | moduledescription |
| **Description** | This table contains information displayed in module descriptions. |
| **Connected Tables** | module |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | moduletopic |
| **Description** | This table contains the topics for each module. Topics only consists of a simple description and a sequence number. The sequence number is automatically generated and is unique in combination with the module id. |
| **Connected Tables** | module |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | professionaltask |
| **Description** | This table contains the description of the task a student has to fulfill when to reach a certain qualification. The tasks are defined in the HBO-I document. |
| **Connected Tables** | qualification |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | profile |
| **Description** | This table connects a study program with a specific curriculum. A curriculum can belong to multiple study programs. Also a study program can have multiple curricula, but for each point in time only one curriculum can be valid. Therefore this connections can be seen as a kind of curriculum versioning. |
| **Connected Tables** | curriculum, studyprogramme |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | profile_qualification |
| **Description** | This table connects profile with qualifications. The connections show the end qualifications of after a specific study program. **This table is unused at the moment.** |
| **Connected Tables** | profile, qualification |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | qualification |
| **Description** | This table contains all permutations of all reachable skills. A qualification is the combination of an activity, an architectural layer and a certain skill level as defined in "levelofskill". |
| **Connected Tables** | activity, architecturallayer, levelofskill |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | studyprogramme |
| **Description** | This table contains the general information about a study programme, for example software engineering or business informatics. |

| **Table** | |
|---|---|
| **Schema** | study |
| **Name** | teachingmaterial |
| **Description** | This table contains the information about teaching materials. There are different types of teaching materials: books, websites, article and others. The type is stored to allow the export of a book list for each module. The table is connected to module descriptions. |
| **Connected Tables** | moduledescription |

# Chapter 8

# Reviews

## 8.1 Project Plan

**Date of Review**
11-09-2017
**Document Title**
Project Plan
**Document Date**
09-09-2017
**(Optional) Commit ID**
6ef0e1fa27d14ab17f4133610077c6befde22b12
**Reviewer Name**
Loek Ehren
**Reviewer Comments**

- Typo 'infomration' in Introduction page 2 line 2
- Typo 'explaination' in Description page 3 line 2
- Typo 'a explanation' in Project page 4 line 2
- Refer to yourself as 'He' not He/She in 3.1.2 page 4 line 3
- Refer to me as 'He' also please in 3.1.3 page 5
- Sjoerd is configuration manager
- Scope cannot change once its defined. It says what must be done and also what must not be done. It has to be defined clearly in the beginning of the project in 3.2
- I think chapter 3.2.2 is not supposed to be in the project plan but in the requirements document or nowhere at all.
- Chapter 3.3 on a new page please
- Typo 'scurm' in 3.3 line 3
- Try to fix the tables and text going off the side of the page.
- In 3.4.2 just refer to the quality standards document and write how we follow the standards etc.

Keep consistency between 'SoFa' or 'SOFA'.

**11-9-2017**
**Project plan**
**8-11-2017**
**1**
**Sjoerd Brauer**
**Reviewer Comments**

**1. Introduction - 1.1 Background information**
text gives the impression that there are multiple business students. please update.

**2. Introduction - 1.2 Background information**
Text error in first sentence: ot, should be or.

**3. Chapter 2 - Assignment**
It's also about planning, but our focus lies on the modules.

**4. Chapter 3 - Stakeholders**
You forgot the exam commission who is a user of module descriptions as well.

**5. Chapter 3 - Roles**
Sjoerd is the configuration manager.

**6. Chapter 3 - Functionality**
Please refer to SRS the SRS contains all functionality/requirement.

**7. Chapter 3 - Data**
The text goes out of the document.

**8. Chapter 3 - Products**
add configuration management plan.

**9. Chapter 3 - Organisation**
the tables are only partly shown on page.

**10. Chapter 3 - Organisation**
No external communication plan?.

**11. missing**
Missing stakeholder Analysis.

**Date of Review**
14-09-2017
**Document Title**
Project Plan
**Document Date**
14-09-2017
**(Optional) Commit ID**
/
**Reviewer Name**
Loek Ehren
**Reviewer Comments**

- Grammar error in 2.2 Problem ’there’ ’their’
- Describe the problem a bit more. ”The modules for the SEBI course in FHTenL” etc. Now it looks so short and abrupt.
- He/She remains in 3.1.2 Quality Manager section
- Typo ’beginns’ in 3.3 Planning line 1
- In communication plan, ’retroperspective’ should be ’retrospective’

*Adjusted Goal and Approach sections. *Added milestones

## 8.2 Quality Plan

**14-9-2017**
**SOFA Quality Guideline**
**11-9-2017**
**1**
**Sjoerd Brauer**
**Reviewer Comments**

chapter 1 documents 4th bulletpoint: Document refers to wrong directory, should be 25reviews.

chapter 1 documents 7th bulletpoint. Backlog? where is the backlog.

chapter 1 documents 1th asterix. please properly write down how name should be like name_versionnumber

chapter 2.1 bulletpoint 4 100% line test? thats pretty difficult as you know.

chapter 2.1 bulletpoint 5 Every scenario possible? that's too extreme. Just realistic scenarios.

chapter 2.2 last bulletpoint. if more then 2 things are changed it's too big? please don't define size in this document.

general: please use numbering instead of bulletpoints.

## 8.3 Configuration Plan

**Date of Review** 11.09.2017
**Document Title** Configuration Management Plan
**Document Date** 11.09.2017
**(Optional) Commit ID** b5dde48ba02a05d41826e8716a087647795c8e41
**Reviewer Name** Nils Nieuwenhuis
**Reviewer Comments**
3.1 CI management:
TYPO: The configuration manager will pulse group members if they do not update their CI for any reason.

## 8.4 User Stories

**Date of Review**
11-09-2017
**Document Title**
User Stories
**Document Date**
11-09-2017
**(Optional) Commit ID**
2287fb3e22813472e5bd6ad180b02119d0302eb1
**Reviewer Name**
Loek Ehren
**Reviewer Comments**
US_001:

- Random capital at 'Creating'.
- Does the teacher create a module or a module description? They are different things.
- What does retrieve guidelines mean??

US_002:

- Does the teacher create a module or a module description? They are different things.
- What does retrieve guidelines mean??

US_005:

- The teacher makes the module or the module DESCRIPTION public? They are different things.

US_006:

- Saying 'and gets permission according to his role' is already specifying requirements.

US_007:

- 'logout' should start with a capital.
- The story should just be 'A user logs out'.

US_009:

- 'adding users' should start with a capital.

US_012:

- Is this not the same as modifying a module?

US_014:

- Name should start with capital.

**Date of Review**
12-09-2017
**Document Title**
User Stories
**Document Date**
11-09-2017
**(Optional) Commit ID**
38b159a2161fba6859341c0d2b6ba9c268c1c914
**Reviewer Name**
Loek Ehren
**Reviewer Comments**

- Are US14, US12 and US02 all the same? What's the difference?
- Aren't US15 and US16 the same? I thought there was no difference between module and module description? What is the difference?
- US20 / US21: If you're going to specify changing the individual parts of a module as separate user stories you should do all of them. So competences, credits, learning goals, teaching material etc.

**Date of Review**
12-09-2017
**Document Title**
User Stories
**Document Date**
12-09-2017
**(Optional) Commit ID**
37622c1
**Reviewer Name**
Loek Ehren
**Reviewer Comments**

- Random text outside the table at US18
- US20-. Is modifying those things a subset of US14?
- US17 Is there also a display without HBOI matrix? Just displaying it normally?

## 8.5 User Story Diagram

**Date of Review** 14.09.2017
**Document Title** User Story Diagramm
**Document Date** 14.09.2017
**(Optional) Commit ID** 4ed4569162594006fa0a4c9688b572bdb4534c90
**Reviewer Name** Nils Nieuwenhuis
**Reviewer Comments**

1. Department Head: Please change ”including” to ”include”.
2. System user: ”Login” should *include* ”Logout”.
3. Teacher: There are two use cases with ”Modifying a module”.

# Chapter 9

# Handover Document

# Handover Document

Fontys Module Management System

Nils Nieuwenhuis, Loek Ehren, Sjoerd Brauer, Tobias Derksen

*Fontys School of Technology and Logistics*
*Informatics*
*Module Software Factory (SOFA)*

Venlo, 12th January 2018

# Contents

# Chapter 1

# Introduction

This project started during the Software Factory (SOFA) module. The goal of the project was to develop a module management system. The department head of Software Engineering and Business Informatics is the customer of the project. The project, when fully implemented should be able to support all educations of Fontys though.

More information can be found in the 10projectplan at `https://github.com/FSG1/mgmt/tree/master/10projectplan`.

# Chapter 2

# Software Architecture

## 2.1 Frontend

For Frontend a group member did research in multiple technologies for front-end. Then in discussions that followed, the group voted for Angular for front-end. For more information see the research which can be found in the "30analysis/frontend_frameworks" directory. There is also a page breakdown diagram and a page navigation diagram if one needs more information found in "100documentation/frontend".

## 2.2 Backend

As to lower the cost it is decided to use open source programming languages. The deployment environment supports open source as well. The deployment environment is a Linux server provided by the customer. In this light it is chosen that Java is used for back-end development. The group is well experienced with Java, which made a big impact on the choice as well. See the back-end research for more information found in "30analysis/java_frameworks_backend" directory.

Furthermore a research was done on which frameworks to use in this project. From this research it was decided to use JDBC because complex query's are required. These are not possible with for example Hibernate. Furthermore it was decided that Jersey was to be used for the endpoints. The group preferred small frameworks over big frameworks that do more then one needs.

## 2.3 Database

To store the relational data, a PostgreSQL database is used. This decision was made by the customer because he already provided an engineered database when the project started. Nevertheless PostgreSQL is a reasonable choice, it provides all necessary feature important for the project.

## 2.4 Architectural decisions

One of the big decisions made is that the littlest amount of endpoints on a page should be called as possible. The consequence of this is that we have few endpoints but many interfaces (object definitions). The communication between front-end and back-end is thus faster as result. If one is to look through the front-end interface one will see that we almost exclusively use interfaces instead of classes. This is done because creating an object adhering to an interface has a lower memory footprint.

Chapter 2: Hardware Document 84

For more information about the endpoints see the Software "architecture endpoints aesthetic.pdf" document in the 100documentation directory.

# Chapter 3

# Setup Frontend

1. Clone the GitHub repository of the Frontend application `https://github.com/FSG1/frontend.git`
2. Install the Angular CLI as it is described in the angular "Get Started" tutorial `https://angular.io/guide/quickstart`.
3. Open the cloned frontend repository with a terminal and execute "npm install" to install all necessary node packages for the frontend application.
4. Look at the "page breakdown diagramm.png" to understand the structure and the pages of the frontend. You can find it in the "mgmt" repository on GitHub `https://github.com/FSG1/mgmt/tree/master/100documentation/frontend`.
5. Read the "Software architecture endpoints asthetic.pdf" to understand the connections to the backend application. You can find it also at the "mgmt" repository on GitHub `https://github.com/FSG1/mgmt/tree/master/100documentation`.
6. To understand the source code read the documentation. *RTFM*
7. To start the application open the cloned frontend repository with a terminal and execute "ng serve". Point your browser to `http://localhost:4200` to view the frontend application.
8. If you want to test the frontend application make yourself familiar with the Testing documentation of Angular, you can find it here `https://angular.io/guide/testing`. To start the test, open the cloned frontend repository with a terminal and execute "npm test". A browser window opens which shows you the results of your tests.

# Chapter 4

# Setup Backend

The backend of FMMS consists of a REST API connected to a PostgreSQL database. The REST API is written in Java using Jersey. Jersey is an open source framework that supports JAX-RS, which is a simple API spec for creating REST APIs. The Jersey documentation can be seen at `https://jersey.github.io/`.

To checkout the project:

1. First install Maven if you do not have it already.
   `https://maven.apache.org/install.html`
2. Clone the GitHub repository with `git clone git@github.com:FSG1/backend.git`

The backend can be run in Docker or standalone from the jar.

- To build and run the backend in Docker, refer to the chapter on Docker deployment.
- To build and run the backend standalone, run `mvn build` followed by `mvn exec:java` in the root of the project, or use your IDEs built in build-and-run functionality.

# Chapter 5

# Setup Database

The database needs an already installed version of PostgreSQL. To import the database schema, first a new user and a new database have to be created. After that, the SQL files from the "scripts" directory can be imported to the database. Please take care of the order as it is indicated by the numbers.

Listing 5.1 shows how to create a user, a database and import all SQL files. The script is written in bash and only works in Linux systems. For Windows or MacOS please use a database client like PgAdmin or DataGrip.

**Please be aware that the database repository does contain only the database schema. To import the data there is a SQL file inside the subversion repository which can be imported the same way as the other files.**

```bash
        # Create user "module"
        sudo -u postgres \
                psql --command "CREATE USER module WITH PASSWORD 'fmms';"

        # Create database "modulemanagement"
        createdb -O module modulemanagement

        for f in scripts/*.sql; do
                sudo -u postgres psql -U module -d modulemanagement -f "$f";
        done
```

Listing 5.1: Import database schema

# Chapter 6

# Run with Docker

For each part of the project there is a docker[1] file which can be used to run the software. The docker file automates the build process and encapsulates it into a container. These containers runs on every operating system and does not need any external dependencies besides the installed docker daemon. You can compose the separate containers to a full service which includes all parts of the project.

## 6.1 Database

To setup a database server and create the proper users and databases can be very error-prone. Therefore a docker container can be used which automatically sets up the user, the database and tables.

During the creation process, the SQL files in the "scripts" directory are executed, sorted by name (that's why there are numbers in front of the filenames). The database repository only contains the table structure and does not contain any data. Nevertheless, data can be automatically imported by copying a proper SQL file into the "scripts" directory before building the container.

When running the database as a docker container, please make sure that there are no other databases running on port 5432 or change the port mapping.

```
        cd database
        # Build container
        docker build -t fmms-database .
        # Run container
        docker run -d --name fmms-database -p 5432:5432 fmms-database
```

Listing 6.1: Build and run Database Container

## 6.2 Backend

The configuration of the Backend API can be done without any changes to the source code. During initialization, environment variables [2] are read and the values will be used as configuration. There are basically three important parts to configure:

- The Backend URI containing port and base url
- The database connection

[1] https://www.docker.com/get-docker
[2] https://en.wikipedia.org/wiki/Environment_variable

- Username and password for restricted actions

Restricted actions are all actions which can change the data in the database. The backend uses HTTP Basic authentication to authenticate users who wants to perform restricted actions. The credentials are currently hard-coded into the configuration and can be set via environment variables.

The default values has been chosen to allow the software to be run locally. For server deployment other values need to be entered. The default database URL contains the default docker host ip address, which implies that a PostgreSQL server is bound to the port 5432 of the host.

The environment configuration can be given into the docker containers using the docker environment functionality (see the Docker documentation [3]).

| **Name** | **Default Value** | |
|---|---|---|
| HOST | 0.0.0.0 | IP Address to bind server socket to. Usually the default value will do the job. |
| PORT | 8080 | Server port to listen on |
| BASE | /fmms | API Base URI |
| DB | 172.17.0.1:5432/modulemanagement | DB URL for JDBC postgres driver format: <IP>:<PORT>/<databasename> |
| DB_USER | fmms | Username to access the database |
| DB_PASSWD | fmms | Password to access the database |
| AUTH_USER | fmms | Username for restricted actions |
| AUTH_PASSWORD | modulemanagement | Password for restricted actions |

Table 6.1: Environment Configuration for Backend

```
        cd backend
        # Build container
        docker build -t fmms-backend .
        # Run container
        docker run -d --name fmms-backend -p 8080:8080 fmms-backend
```

Listing 6.2: Build Backend Container

## 6.3 Frontend

The frontend configuration needs to be done inside the source code before building the container. The default configuration works only for local use and is not suitable for server deployment. The configuration is done via environment files which are loaded based on cli arguments. [4]

```
        cd frontend
        # Build container
        docker build -t fmms-frontend .
        # Run container
        docker run -d --name fmms-frontend -p 4200:4200 fmms-frontend
```

Listing 6.3: Build Frontend Container

[3] https://docs.docker.com/engine/reference/run/#env-environment-variables
[4] http://tattoocoder.com/angular-cli-using-the-environment-option/

Chapter 0: Handover Document 99

## 6.4 Compose

To run all parts of the software inside docker containers, Docker Compose[5] can be used to run and supervise the docker containers. Therefore a docker compose file is needed which defines the structure of the application and the needed parameters. The following listing shows a docker compose file which contains all needed configuration to run the project on your local machine.
To use docker compose perform the following steps:

1. Install Docker and Docker-Compose
2. Build Database, Backend and Frontend as explained in sections 6.1, 6.2 and 6.3
3. Put the content of listing 6.4 into a file named „docker-compose.yml“
4. Run shell command „docker-compose up -d“ in the directory with the file created in the previous step

[5]`https://docs.docker.com/compose/install/`

```
version: '2'

networks:
  fmms:
    driver: bridge

services:
  database:
    restart: always
    image: fmms-database
    ports:
      - 5432:5432
    networks:
      - fmms

  backend:
    restart: always
    image: fmms-backend
    ports:
      - 8080:8080
    environment:
      - HOST=0.0.0.0
      - PORT=8080
      - BASE=/fmms
      - DB=database:5432/modulemanagement
      - DB_USER=module
      - DB_PASSWD=fmms
      - AUTH_USER=fmms
      - AUTH_PASSWORD=fmms
    volumes:
      - maven:/root/.m2
    networks:
      - fmms

  frontend:
    restart: always
    image: fmms-frontend
    command: ["--no-live-reload", "--no-watch"]
    depends_on:
      - backend
    ports:
      - 4200:4200
    networks:
      - fmms

volumes:
  maven:
    driver: local
```

Listing 6.4: Docker Compose File

# Chapter 7

# Staging Area

There is a running instance of the project available at `https://modulemanagement.fontysvenlo.org/`. This instance is mainly a staging and testing instance for the customer and does not guarantee any data persistence. The instance is automatically built and deployed with a Jenkins CI at `https://jenkins.fontysvenlo.org`. The Jenkins installation contains three projects:

- backend
- frontend
- deployment

The backend project is responsible to build the backend code, run unit tests, code style analysis and test coverage checks and, if all checks passes, build a docker container and push the container to a local docker registry.

The frontend project is responsible to build the frontend code, run unit tests and, if the tests run successfully, build a docker container and push the container to a local docker registry.

The deployment project first stops the running frontend and backend containers. After that it starts the build of the frontend and backend projects. When they finished successfully the deployment project pulls the new containers from the registry and starts the frontend and backend with the new versions.

The build configurations are stored in files named "Jenkinsfile". There is exactly one Jenkinsfile for each project which lives in to root of each repository. It uses a special Jenkins pipeline syntax written in Groovy.

**To deploy a new version of the project, the build of the deployment project needs to be started manually.**

# Chapter 10

# Minutes

## 10.1 Daily Scrum Meetings

Campus Venlo FHTenL
Informatica

**Minutes SOFA – Van Aarsen Machinefabriek 31.08.2017**

| | |
|---|---|
| Date: | 31.08.2017, 10:30 – 11:00 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-08-31 Meeting.doc |

| ***Name*** | ***Short*** | ***Minutes*** | ***Participant*** | ***To be informed*** |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. Get in contact with company
2. Move one SoFa Day to Monday

1. SB tried to call the company supervisor. He is currently in a meeting. We will call later.

2.
SAP and DAMI are on mondays now, so there is the need to move one SoFa day to monday. The group decided to move friday to monday. This will take effect after the next monday. First monday will be 11.09.2017 .

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **07.09.2017**

Date: 07.09.2017, 10:45 – 12:00
Location: Campus Venlo, Ambient Lab
File: 2017-09-07 Meeting.doc

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. Group Name & Product Name
2. Usage of Github
3. Group roles
4. Product License
5. Further procedure

1. Group Name: FSG1 – Fontys SoFa Group 1
   Product Name: FMMS – Fontys Module Management System

2. The group decided to use Github to host the central repository. The repository will be publicly available. Maybe there are some project documents which contains sensible data. The customer needs to be asked which data is sensible.
   https://github.com/FSG1

3. Project Manager: Nils Nieuwenhuis
   Configuration Manager: Sjoerd Brauer
   Quality Manager: Loek Ehren
   Software Architect: Tobias Derksen

4. Discuss with Client at Client Meeting 7/9 - 15:00

5. Further procedure this week
   - Analyze domain information acquired during customer meeting on 05/09.
   - Define basic project mgmt stuff: Stakeholders, communication plan, etc.
   - Setup Github organization & repositories
   - Analyze domain → Domain Model

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **08.09.2017**

| | |
|---|---|
| Date: | 08.09.2017, 10:15 – 11:00 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-08 Meeting.doc |

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. Saving sensible data

1. All sensible data is stored inside a private SVN repository.
   https://www.fontysvenlo.org/svnp/2427362/FGS1

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **11.09.2017**

| | |
|---|---|
| Date: | 11.09.2017, 10:15 – 11:45 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-11 Meeting.doc |

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time (Friday 08/09)
2. Agenda for today

1. **Nils**: Wrote project plan, problems with chapter content as well as latex
   **Sjoerd**: Update domain model, wrote user stories, review project plan, start configuration management plan, problems with loek ehren
   **Tobias**: Created database from provided dump, wrote down some questions for client and to discuss with group
   **Loek**: Reviewed project plan, first draft of SRS
2. **Sjoerd**: Configuration management plan, document with user stories
   **Tobias**: Improve database design, create document with constraints, develop proposals for technology stack
   **Loek**: Look into user stories
   **Nils**: SAP

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD) 12.09.2017**

| | |
|---|---|
| Date: | 12.09.2017, 09:30 – 10:00 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-12 Meeting.doc |

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:

1. Agenda for today

1. **Nils**: Create Trello Board for Scrum
   **Sjoerd**: Write letter to van Aarsen
   **Tobias**: Improve database design
   **Loek**: Refine SRS

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD) 14.09.2017**

| | |
|---|---|
| Date: | 14.09.2017, 10:40 – 11:00 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-14 Meeting.doc |

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today
3. Todo project plan

1. **Loek**: Reviewed user stories, added user stories to SRS and created questions for non functional requirements
   **Nils**: Worked on project plan and fixed structure.
   **Sjoerd**: Changed user stories according to client‘s input. Defined learning goals. Updated configuration mgmt document. Wrote letter to van Aarsen.
   **Tobias**: Worked on the database and created a lot of entities and structure.
2. **Nils**: Improve project plan after discussion about (3)
   **Loek**: Improve SRS and help Nils with project plan
   **Sjoerd**: Improve configuration mgmt plan. Create user story diagram
   **Tobias**: Keep on improving database, document database by creating a report containing database structure and constraints. Think about non-functional requirements.

Goals Discussion:
Deliver high quality product
Deliver of a working product but maybe not finished
Deliver good documentation
Extendable product

Approach Discussion:
Scrum
Quality mgmt. & config mgmt.
Communication plan
Patterns

3. Discuss project goals, project approach

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **18.09.2017**

| | |
|---|---|
| Date: | 18.09.2017, 10:20 – 10:30 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-18 Meeting.doc |

| ***Name*** | ***Short*** | ***Minutes*** | ***Participant*** | ***To be informed*** |
|---|---|---|---|---|
| S. Brauer | SB | | | X |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Reviewed project plan, improved learning goals after coach feedback.
   **Nils**: Finished first draft of project plan, create trello content for scrum
   **Tobias**: Finished database structure. Started working on database documentation
2. **Nils**: Improve project plan after customer feedback, revise learning goals
   **Loek**: Review project plan after customer feedback; create questions to get non-functional requirements
   **Tobias**: Improve database after customer feedback, keep working on database documentation, improve learning goals after coach feedback.

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD) 19.09.2017**

| | |
|---|---|
| Date: | 19.09.2017, 09:45 – 11:00 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-19 Meeting.doc |

---

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today
3. Choice of technology stack

1. **Loek**: Fixed learning goals
   **Nils**: Changed project plan
   **Tobias**: Fixed learning goals
   **Sjoerd**: Nothing, simple (physical) presence, professional breathing

2. **Nils**: Change project plan according to customer feedback. Work on learning goals.
   **Loek**: Fix SRS according to customer feedback. Review updated project plan.
   **Tobias**: Working on database and database documentation
   **Sjoerd**: Work on learning goals, Research Java frameworks to use in backend

3. As a group we choose a technology stack.
   There were three choices to make because our architecture consists of 3 different layers: Database, Backend, Frontend.

   **Database**: For the database layer we had already gotten a sample database provided by the customer and written in PostgreSQL. So the choice of using PostgreSQL was logical. Furthermore Postgres provides some features which allows to ensure business constraints on database level which are not provided by other open-source databases.

   **Backend**: The backend should be able to handle HTTP requests and using JSON as datatype for input and output data. We decided to use Java. Java fulfills all requirements mentioned, furthermore due to the pre-knowledge of all group members, Java is a reasonable choice so no one needs to learn the language as a whole. Nevertheless, there are some Java frameworks which makes life a lot of easier. There needs to be some research to decide whether and which framework we should use.

   **Frontend**: The decision about a frontend technology were quite hard. The main requirement of the customer was, that it run inside a browser. Further requirements were: communication with an JSON API; framework allow good application structure; performance.

All the requirements are met by a few different JavaScript frameworks. We picked out two of them to further analysis: Angular and React.
After analysis, we found out that both frameworks meet all requirements perfectly, so we had a look at some example projects provided by Loek and Tobias.
After that we carried out a simple vote. The result was Angular 2 votes, React 1 vote. Loek refused to vote.
So the final decision is using Angular in its newest version for the frontend layer.

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD) 21.09.2017**

| | |
|---|---|
| Date: | 21.09.2017, 11:20 – 11:40 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-09-21 Meeting.doc |

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today
3. Split up development roles

1. **Loek**: Non-functional requirements added to SRS. Worked in personal development plan.
   **Nils**: Changed project plan according to the customer feedback. Started with personal development plan
   **Tobias**: Worked on personal development plan.
   **Sjoerd**: Worked in research Java frameworks and improved personal development plan. Studied Angular 4

2. **Nils**: Finish personal development plan.
   **Loek**: Setup basic code quality standards.
   **Tobias**: Work on development plan. Setup angular project.
   **Sjoerd**: Try to finished Java framework research, write proposal which framework to use in the backend.

3. Roles: Scrum Master, Software Architect, Software Developers, DevOps Engineer

   Role Assignments:
   Nils: Project Leader, Software Developer
   Sjoerd: Configuration Manager, Software Architect
   Loek: Quality Manager, Scrum Master, Software Developer
   Tobias: DevOps Engineer, Software Developer

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **25.09.2017**

Date: 25.09.2017, 10:30 – 10:40
Location: Campus Venlo, Ambient Lab
File: 2017-09-25 Meeting.doc

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Worked on development plan, create backend project with maven
   **Nils**: Worked on personal development plan.
   **Tobias**: Worked on personal development plan, Setup basic angular project, setup angular template project
   **Sjoerd**: Finished java backend research, wrote backend proposal, made personal development plan, started learning angular

2. **Nils**: Create user stories in Trello
   **Loek**: Create jersey project skeleton
   **Tobias**: Improve database design based on customer's feedback
   **Sjoerd**: Study angular, take angular heroes tutorial, Create backend ↔ Frontend interface

**Minutes SOFA – FH TenL Informatics (HVD) 02.10.2017**

| | |
|---|---|
| Date: | 02.10.2017, 11:00 – 11:15 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-10-02 Meeting.doc |

---

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | | X |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Implemented backend endpoint for all semesters in a curriculum and created database query for it. Started testing the backend. Problems with testing database layer of backend.
   **Tobias**: Working database deployment using Docker. Problems with persistence of database during docker restart.
   **Sjoerd**: Created backend architecture document. Updated JSON object table for this sprint. Updated research according to JAC feedback. Updated personal development plan according to JAC feedback. Created mockups for user stories. Problems: Research plan is not document according to the APPL guideline.

2. **Loek**: Research how to test the database layer of the backend and how to test the endpoints.
   **Tobias**: Working on database persistence inside docker container. Create dockerfiles for backend and frontend and compose this with docker-compose to o
   **Sjoerd**: Start implementing the frontend. Improve mockups after customer meeting. Improve software architecture document after customer meeting.

**Minutes SOFA – FH TenL Informatics (HVD)** **03.10.2017**

| | |
|---|---|
| Date: | 03.10.2017, 10:00 – 10:15 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-10-03 Meeting.doc |

---

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Figured out how to do dependency injection in jersey, wrote algorithm to send correct result to client
   **Tobias**: Finished persistent database docker container.
   **Sjoerd**: Improved backend architecture, experiments on in-memory-database, Problems with angular default implementation.
   **Nils**: Worked on frontend design, worked on the project plan

2. **Loek**: Research how to test a backend endpoint using mocks
   **Tobias**: Revise personal development plan, create docker container for frontend and backend to run in
   **Sjoerd**: Research in testing angular application using in-memory-database or mocks. Improve research about backend framework
   **Nils:** Work on frontend, finish project plan

3. Sjoerd brought motivational nuts which will help the group to make progress. Everytime a task has been finished, the responsible person may get one nut.
4. Tobias proposed to perform unit testing on database layer. Our quality plan only defines automated tests for frontend and backend but the software architecture consists of three layers. Databases are usually not tested because most developers do not see the amount of logic inside the database which should be tested as well as the business or the presentation logic.

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **05.10.2017**

Date: 05.10.2017, 11:00 – 11:15
Location: Campus Venlo, Ambient Lab
File: 2017-10-05 Meeting.doc

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Figured out how to test the jersey-application using the DI system.
   **Tobias**: Improved personal development due to the individual deadline by JAC
   **Sjoerd**: Fixed the in-memory-database implementation for frontend testing. Looked up how to test angular applications.
   **Nils**: Worked on frontend: how to display the semesters, Worked on project plan

2. **Loek**: Create tests for backend endpoints and rework backend application structure.
   **Tobias**: Improve database docker file to run on windows. Setup Dockerfile to run the backend.
   **Sjoerd**: Studying testing angular applications, creating tests for frontend application.
   **Nils:** Merge code into master, Learn about testing angular application

3. After the daily scrum meeting, Sjoerd wants to give a motivational speech.
4. Sjoerd adviced Nils to update the configration table

Campus Venlo / Informatics FH TenL

**Minutes SOFA – FH TenL Informatics (HVD)** **09.10.2017**

| | |
|---|---|
| Date: | 09.10.2017, 10:20 – 10:30 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-10-09 Meeting.doc |

---

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | | X |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Wrote tests for backend. Finished backend
   **Tobias**: Worked docker deployment, creating docker database on windows
   **Nils**: Finished project plan. Learned something about testing angular

2. **Loek**: Prepare customer meeting, Maintain scrum backlog
   **Tobias**: Deploy whole service using docker compose
   **Nils:** Understand what Sjoerd is doing. Learn more about testing angular applications

**Minutes SOFA – FH TenL Informatics (HVD) 10.10.2017**

| | |
|---|---|
| Date: | 10.10.2017, 10:10 – 10:15 |
| Location: | Campus Venlo, Ambient Lab |
| File: | 2017-10-10 Meeting.doc |

---

| *Name* | *Short* | *Minutes* | *Participant* | *To be informed* |
|---|---|---|---|---|
| S. Brauer | SB | | X | |
| L. Ehren | LE | | X | |
| N. Nieuwenhuis | NN | | X | |
| T. Derksen | TD | X | | |

Agenda:
1. What did we do last time
2. What we do today

1. **Loek**: Completed test coverage on backend. Changed database query. Add CORS headers to responses. Tobias helped with setup of backend in docker.
   **Tobias**: Finished running all services with docker composed. Helped on backend to get it running; Helped on frontend to get it running,
   **Nils**: Got frontend running, fixed some issues
   **Sjoerd**: Wrote test for frontend application; Learned about reading; Updated personal development plan

2. **Loek**: Create the endpoint to get the curriculums defined by Sjoerd.
   **Tobias**: Refine Dockerfile for frontend; Enable automatic tests for frontend.
   **Nils:** Merge modules of BI and SE in one view
   **Sjoerd**: Get docker running, Discuss with Nils about frontend development. Change test data to use in-memory-database. Today’s: Add endpoint for curriculums

## Fontys Venlo SoFa Group 1

Minutes for October 24, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today
3. Motivational Speech

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Improved mockups after kickoff meeting, Improved software architecture document, wrote tests for frontend, start implementing skill matrix component |
| **Nils** | Worked on frontend design (module overview), Tackled a bug with the dropdown menu |
| **Loek** | Created abstract class for endpoints to inherit common stuff. |
| **Tobias** | Worked on SQL queries to get postgresql data out in the right format. Created latex template for minutes absed on feedback from JAC |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Continue working on the skill matrix component, slight modification on software architecture document, improve frontend tests. |
| **Nils** | Work on overview of learning goals, write tests for frontend. |
| **Loek** | Work on abstract class, create new endpoints for this sprint, extend tests to red sample json data from files. |
| **Tobias** | Work on latex minutes, start creating database tests. |

### Motivational Speech (S. Brauer)

Sjoerd gives a motivational speech about how short is life, and a part of our short lifetime goes into the project. When you probably meet god, you can reflect on what you did to this project.

## Fontys Venlo SoFa Group 1

Minutes for October 25, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Edited Mockups for second user story, defined endpoint for second user story, created matrix and navigation component, fixed bugs. |
| **Nils** | Finished the new design and merged to master, merged new tests and components to master, worked on learning goals overview |
| **Loek** | Created base class for service endpoints, create tests for new endpoints |
| **Tobias** | Created latex template for minutes, Started working on database documentation. Reviewed pull requests of loek |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Refine frontend tests because of reasons, Create tests for user story 3, implement frontend <-> backend communication, implement components for user story 2 |
| **Nils** | Finish learning goals overview, Implement new components for user story 2 |
| **Loek** | fix broken branches and tests, finish backend endpoint for user story 2 |
| **Tobias** | Work in database documentation, review new pull requests and merge branches to master |

## Fontys Venlo SoFa Group 1

Minutes for October 30, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Removed redundant information from software architecture and mockups; Improved design of skill matrix; Created tests for skill matrix components |
| **Nils** | Finished overview of learning goals; Fixed issues with the backend; Helped Sjoerd with Docker; Added travis ci file to execute tests on frontend |
| **Loek** | Finished learning goals endpoint; fixed tests on backend; rewrite tests to match a similar structure |
| **Tobias** | Improved SQL queries to get json structure out of the database; Reviewed git pull requests, worked on database documentation |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Change software architecture depending on customers answers to questions; Implement the exam learning goal component |
| **Nils** | Finished test for module component |
| **Loek** | Expand learning goal endpoint to deliver whole module information; Create tests for extended endpoint |
| **Tobias** | Work on database documentation; Help Loek to improve SQL query for extended endpoint |

**Fontys Venlo SoFa Group 1**

Minutes for November 02, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Finished exam learning goal component with unit tests; Looked into module component tests; Assisted Loek with frontend backend connection; Update Software Architecture Document |
| **Nils** | Tried to write a test for module component, did not worked due to unknown reason; Merged frontend pull requests |
| **Loek** | Removed separate module endpoint and merged it into curriculum endpoint; Rewrote backend tests to meet a common structure and test the right think; Fixed frontend backend connection; Backend bugfixes |
| **Tobias** | Worked on SQL query to get all module information out of the database in the right structure. |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Discuss with Nils what will be done next; Look into module component tests; Rethink focus of frontend tests |
| **Nils** | Discuss with Sjoerd the next steps in frontend development and how to deal with tests. |
| **Loek** | Help with the frontend development |
| **Tobias** | Extend SQL query to load skill matrix after modules; Make frontend more fancy |

## Fontys Venlo SoFa Group 1

Minutes for November 9, 2017

**Present:** L. Ehren (Chair), S. Brauer, T. Derksen (Secretary) N. Nieuwenhuis *(via Skype)*

### Agenda

1. What did we do last time
2. What will we do today
3. Motivational Speech

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Assisted Loek with frontend implementation; made mockups for new sprint; developed new endpoint for sprint; Developed new component for semester overview. |
| **Nils** | Helped sjoerd with docker; reviewed some pull requests; merged pull requests after successful review; worked on design of new component |
| **Loek** | Created new endpoints for sprint 3; created tests for new endpoint; QUery still missing -> Tobias |
| **Tobias** | Implemented user stories 2,3,4 |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Review frontend |
| **Nils** | Looked at US5 |
| **Loek** | Look in frontend code and see what is done there; Implement some caching strategies inside the server |
| **Tobias** | Write query for new endpoint; Implement US5 |

### Motivational Speech (S. Brauer)

Sjoerd gives his motivational speech: We are quite ahead of our planning, our group now works like a natural working chain. Now we are fixed on positions and this works well together. If you do something out of the ordinary it will explode.

## Fontys Venlo SoFa Group 1

Minutes for November 13, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Designed new endpoint for current sprint; Made mockups for new view; Started developing new view qualification overview |
| **Nils** | *Sick* |
| **Loek** | Added endpoints for qualifications and semester information; Working exception handling in backend; |
| **Tobias** | Implemented user story to order projects at the end of a semester; Wrote query for getting semester information from the database; Synced with HOM to get access to the fontysvenlo server |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Get in sync with Nils and divide the work which has to be done |
| **Nils** | Get in sync with Sjoerd for current status of frontend development |
| **Loek** | Improve exception handling in backend to display real error information; Think about some caching in backend. |
| **Tobias** | Setup first deployment setup on fontysvenlo server; Perform database tasks for this sprint; Setup CI and CD pipelines on server |

## Fontys Venlo SoFa Group 1

Minutes for November 14, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Changes to the software architecture; Worked on qualification overview view |
| **Nils** | Fixed skill overview matrix; Added backend service for semester overview |
| **Loek** | Changed exception handling in backend; Working on splitting up services in different classes |
| **Tobias** | Setup up deployment on fontysvenlo server; Draw deployment strategy and architecture on server with HOM; Setup Jenkins and create projects to build the backend and the frontend |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Write tests for qualifications overview; Refactoring of code, remove unused code |
| **Nils** | Assists Sjoerd in writing tests for qualification; Finish frontend for deployment |
| **Loek** | Still work on splitting up classes; Finish endpoint when query is ready |
| **Tobias** | Fix small frontend issues; Create query for qualification |

## Fontys Venlo SoFa Group 1

Minutes for November 16, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Wrote tests for frontend (us6); Found big bug -> repaired; Fixed dependencies for tests |
| **Nils** | Worked on frontend design for qualification overview table; Started using child-routes |
| **Loek** | Added database connection pool; Fixed bug introduced with connection pool; Completed endpoint for qualification overview; Wrote tests for backend |
| **Tobias** | Wrote database query to get qualifications from database; Deployed application on fontysvenlo server; Reviewed pull requests |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Refactor API endpoints; Discussion about frontend |
| **Nils** | Work on using child-routes |
| **Loek** | Refactor enpoints and tests |
| **Tobias** | Incorporate the new database data got from HVD; Work on deployment to get a subdomain and SSL running (with HOM) |

Sjoerd reminded the group to regularly update the configuration item table.

**Fontys Venlo SoFa Group 1**

Minutes for November 23, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

## Agenda

1. What did we do last time
2. What will we do today

## What did we do last time

| | |
|---|---|
| **Sjoerd** | Made mockups for new sprint; Refactored endpoints; Refactored software architectural document; Implemented refactored endpoints to frontend |
| **Nils** | Created table for exam information; Changed and added tests for frontend |
| **Loek** | Edited backend to fit refactored endpoints; Improved tests |
| **Tobias** | Updated queries for refactored endpoints; Improved database for new requirements (teaching material, module assessments); Fixed issues on frontend |

## What will we do today

| | |
|---|---|
| **Sjoerd** | Work on User Story 3 (editing plaintext information) |
| **Nils** | Fix some CSS problems and astethics; Sync with Sjoerd to divide work |
| **Loek** | Create endpoint for User Story 3 with tests |
| **Tobias** | Create concept for basic authentication to secure application; Start researching how to set up a continuous delivery pipeline |

## Fontys Venlo SoFa Group 1

Minutes for November 27, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Worked on software architecture for user story 3; Added endpoints and json objects to document |
| **Nils** | Designed new edit functionality |
| **Loek** | Implemented new endpoint for editing data; Troubles with unit tests for this |
| **Tobias** | Created concept for basic authentication to the system; Implemented backend HTTP authentication |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Form validation for user story 3; Write unit tests for new functionality |
| **Nils** | Sync with Sjoerd for new edit functionality and finish design |
| **Loek** | Implement database transaction; Improve exception handling; Unit tests for new editing endpoint |
| **Tobias** | Implement frontend authentication |

## Fontys Venlo SoFa Group 1

Minutes for November 28, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | I am not ready for this |
| **Nils** | Worked on frontend design for user story 4 |
| **Loek** | Working on endpoint to edit a module; Implemented transactional database queries |
| **Tobias** | Implemented authentication and login on frontend |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Nothing |
| **Nils** | Fix dropdown issues; Sync with Sjoerd what is left to do |
| **Loek** | Finish endpoint and fix some issues with the queries |
| **Tobias** | Change backend to automatically secure all request which try to change data; Work on continuous delivery research |

## Fontys Venlo SoFa Group 1

Minutes for December 7, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Designed endpoints for editing complex module data; Created mockups for editing complex module data; Frontend implementation of those mockups; Added form validation to module edit; Wrote unit test for implemented frontend functions |
| **Nils** | Designed exam information form and created frontend logic for that; Secure the edit module form with input validation |
| **Loek** | Added sql statements for editing module information; Added test for the endpoint and queries for editing module information |
| **Tobias** | Create continuous integration and delivery pipeline using jenkins; Changed database according to the feedback of HVD (prior knowledge references); Performed interview with group and customer to collect requirements for CI/CD |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Sync with Nils and split up the remaining work for frontend |
| **Nils** | Sync with Sjoerd |
| **Loek** | Fixing build error (unknown yet); Improve unit tests for module editing |
| **Tobias** | Write down the collected requirements in a professional way; Add a small improvement to the database according to the customer feedback (sequencenumber of learninggoals); Perform research on continuous integration and delivery |

## Fontys Venlo SoFa Group 1

Minutes for December 12, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today
3. Discussion about PDF generation
4. Customer discussion about PDF generation

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Researched possible solution to generate PDF files for module descriptions; Improved module edit component and updated software architecture |
| **Nils** | Worked in edit learning goals and edit exam information |
| **Loek** | Finished endpoints for editing a module |
| **Tobias** | Created template for the poster (for MON); Research different options to generate PDFs for module description |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Work on frontend aesthetics; Discuss solutions for PDF generation problem. |
| **Nils** | Finish and test edit learning goals and edit exam information parts of frontend |
| **Loek** | Write documentation on how to use the project; Work on personal development plan |
| **Tobias** | Work on personal research; Discuss solutions to the PDF generation problem |

### Discussion about PDF generation

There are different options for creating PDF inside the browser and download it to the client:

- JS PDF
- Pdf Make
- PhantomJS
- Browser's native print support

All of them have several serious disadvantages. JS PDF and PdfMake can either print an image of the shown module description into a pdf or we have to create the whole description manually by calculating all positions.
PhantomJS can render any HTML including Javascript to different output types. But we need to create a proper HTML with CSS styling to get this running properly.
The browser's native print support is highly dependent on the concrete browser and the operating system which is running. It produces different output on different browsers and platforms.

### Customer discussion about PDF generation

The option we research and all the disadvantages we found (as described above) has been presented to our customer. The customer was not happy which one of the given options. We discussed using LaTex to generate a proper PDF. Fontys does not yet have a proper LaTex template for module descriptions, the descriptions are currently managed in a document format. Therefore we need to create a proper LaTex template first, which we then can fill with concrete data and give to a proper compiler.
We discussed, that the time remaining for this module is limited and we probably are not able to create a full LaTex template. We will now divide the module description into parts. We will implement these parts one after each other, so we get a working generation at the end of the sprint, but possibly not all parts of the module description are in there.

## Fontys Venlo SoFa Group 1

Minutes for December 14, 2017

**Present:** L. Ehren (Chair), N. Nieuwenhuis, S. Brauer, T. Derksen (Secretary)

### Agenda

1. What did we do last time
2. What will we do today
3. Poster for SoFa event

### What did we do last time

| | |
|---|---|
| **Sjoerd** | Writing a report about PDF technologies; Looked into how to make the frontend more aesthetic; Helped Nils fix a problem with asynchronous communication |
| **Nils** | Worked on adding learning goals to a module |
| **Loek** | Wrote documentation for backend; Wrote on personal development document according to the feedback of the coach |
| **Tobias** | Researched latex template generation and how to compile latex files using javascript; Started working on poster for Sofa event |

### What will we do today

| | |
|---|---|
| **Sjoerd** | Finish report on PDF technologies; Assists Nils on completing the last parts of this sprint. |
| **Nils** | Finish adding and editing learning goals part of frontend |
| **Loek** | Check backend frontend connection for editing module information and fix arising problems; Work on personal research document |
| **Tobias** | Work on poster for Sofa event and finish a first draft; Create a latex template for module descriptions |

**Poster for SoFa event**

Tobias created the first draft of the poster and divided the space into separate topics. The concrete contents of these topics now have to be written by the group members. The work is divided into parts as followed:

1. About FMMS ((L. Ehren))
2. Problem Description (S. Brauer)
3. Method & Process
   Analysis (N. Nieuwenhuis)
   Scrum (L. Ehren)
4. CI / CD (T. Derksen)
5. Results
   Past (N. Nieuwenhuis)
   Future (S. Brauer)
6. Conclusions (T. Derksen)

# Chapter 11

# Research Documents

## 11.1 Nils

# Angular Research

Nils Nieuwenhuis

Fontys of Hoghschool Techniek en Logistic, Dezember 7, 2017

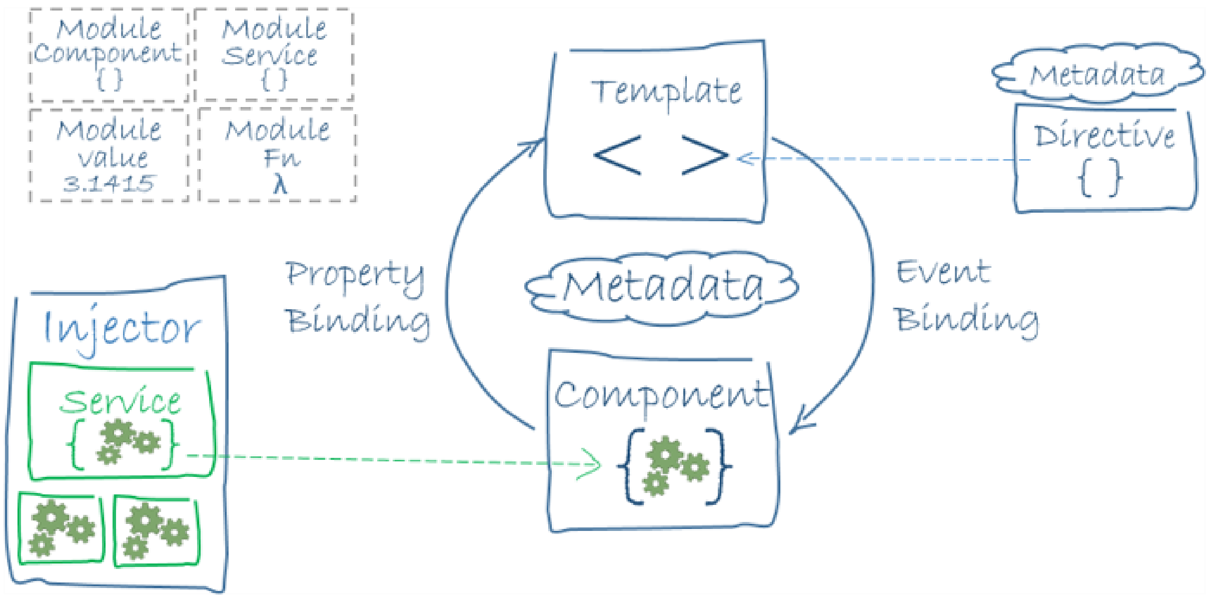
## 1. Introduction

This research is part of the SOFA module in Semester seven at the Fontys University of Applied Science. Our group of four students, three from the Software department and one from the business informatics department creating the “Fontys Module Managementsystem” for the Fontys University of Applied Science to view and edit Module informations. to accomplish that, we decided to use the Angular Framework for the Frontend. This research is about the Angular technics and common best practices we use to build the application.

## 2. Inhaltsverzeichnis

## 3. General Architecture

Angular is a framework for building client applications, it consists of several libraries, some of them core and some optional. You write Angular applications by combining HTML templates with Angularized markup, writing component classes to manage those templates, adding application logic in services, and boxing components and services in modules. The app is launched by bootstrapping the root module. Angular takes over, presenting your application content in a browser and responding to user interactions according to the instructions provided.



## 4. Modules

Angular apps are modular which is called “NgModules” in Angular. Every Angular app has at least one “NgModule” class the root module. An “NgModule” is a decorator, a decorator is function that modifys JavaScript classes, more about decorators later.

NgModule is a decorator function that has a single metadata object whose properties describe the module:

- **declarations:** the *view classes* that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
- **exports:** the subset of declarations that should be visible and usable in the component templates of other modules.
- **imports:** other modules whose exported classes are needed by component templates declared in *this* module.
- **providers**: creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
- **bootstrap:** the main application view, called the *root component*, that hosts all other app views. Only the *root module* should set this bootstrap property.

Here is an example of a typical root module:

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Libaries

Angular comes with a collection of javascript libraries, installed with the npm package manager. They are imported with the "import" statement and every angular library start with the "@angular" prefix. An import in angular looks like this:

```
import { Component } from '@angular/core';
```

## 5. Components

Components are the main way to build and specify elements and logic on the page, through both custom elements and attributes that add functionality to existing components. Everything in Angular is a component for example a List or a form.

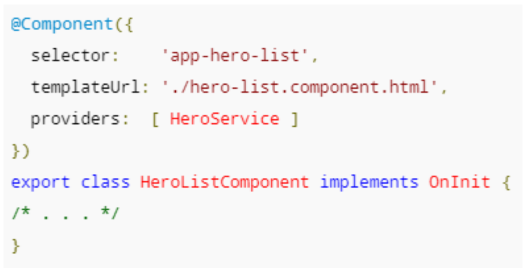
```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

To change a normal class into a component there is a "@Component" decorator, which adds some metadata to the class. These metadata tells Angular what is part of the Component, the most used configurations options are:

- **selector:** The selector is used to insert the component inside of the HTML code of a template.
- **templateUrl:** specifies the path to the template html file for the view.

- **providers:** An array of dependency injections for services that the component requires.

Here is an example of a component declaration:



```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

## 6. Templates

A Template is basically a view of a component and is built with regular HTML code. In addition to the regular HTML code there are special angular HTML tags. More of the angular HTML tags will follow in this research.

## 7. Data binding

Angular supports data binding with the possibility to combine parts of a template with parts of a component. Add binding markup to the template HTML to tell Angular how to connect both sides.

Angular supports four form of data binding:

- **Interpolation:** With this data binding method you can display a component property. To use it curly brackets are used, like this {{myComponent}}
- **Property Binding:** This method is used to fill a html tag with data from a component, for example: <img [src]=”myComponentURL”>
- **Event Binding:** To add an event handler to a template user action, for example a click this method is used: <button (click)=”myClickFunction()”>Click me!</button>
- **Two-way data binding:** With this method an asynchronous dataflow is realized. For example, the value of a component property is used for an input field and displayed in the template. Now if the user changes the value of the input field, the component property is changed too.

## 8. Directives

A directive is basically a class with a @Directive decorator or a like a component with a @Component decorator for the metadata. There are two important decorator types, the structural and attribute directives. Structural directives alter the layout by adding, removing and replacing elements in the DOM. Attribute directives changing the appearance or behaviour of existing elements.

## 11.2 Loek

# SOFA Research

Angular vs React

Loek Ehren

January 9, 2018

# Contents

# Chapter 1

# Introduction

For our SOFA project we have to build a frontend part which interacts with a backend. Several choices of technology are available to use in making a frontend but in the end it came down to two: Angular or React. To make a proper educated choice between the two, the pros and cons of both will need to be listed and compared to eachother, as well as be related to the business needs of the project and the customer.
In this document Angular and React and their characteristics will be described and related to the requirements of the project. After that a conclusion will be drawn and a recommendation will be given on which technology to use.

# Chapter 2

# Requirements

Several requirements have been identified that will help choose what frontend technology to use. These will be listed here:

1. The learning curve should not be too high, since learning a new technology will also take a lot of time.
2. The technology should be able to be testable easily.
3. The technology should be able to provide a good way of having dynamic pages with lots of changing data.
4. A project using this technology should be able to be set up quickly, and be able to be extended quickly.
5. Performance is not that important.

# Chapter 3

# Angular

Angular, formerly called AngularJS, is a JavaScript based open source framework to develop frontend applications. It's created and maintained by Google. As a framework, Angular offers developers tools to make HTTP requests, enable page routing and do component testing among others. It's goal is to enable developing of frontends using the MVC (Model, View, Controller) pattern to separate presentation from data from logic.

Angular versions above 2 use TypeScript, which is a more enhanced language of JavaScript which enables static typing, but largely is the same as JavaScript.

Angular uses templates to render pages using components and directives. These directives come from Angular's own defined attributes on HTML tags and can manipulate data in some way. Such as 'ng-for' which enables you to do a for loop and iterate over some data, maybe to create a row of table cells or other elements in a page.

## 3.1 Pros and Cons

In this section and the next the characteristics of Angular will be tested against the requirements defined in chapter 2.

**The learning curve should not be too high, since learning a new technology will also take a lot of time.**
The learning curve will be pretty high, since Angular will require learning a whole new framework from data binding to testing, becoming familiar with using the MVC pattern as well as learning TypeScript. Angular is a huge library with a lot to learn. What is an advantage though is that every essential thing such as routing and HTTP requests is built in and plenty of documentation is available to read.
However this can also be a negative since there is so much documentation, which might not necessarily be well written, which will inevitably raise even more questions and slow down development time.

**The technology should be able to be testable easily.**
Along with its built in unit testing functionality, third party libraries such as Enzyme can also be used to extend and improve tests.

**The technology should be able to provide a good way of having dynamic pages with lots of changing data.**
Angular has one-way data binding which means the UI can only be updated from the model it represents. If something has to change in the UI, the model has to be updated first, and the UI will follow.
But Angular also enables two-way data binding which means that data in the UI and model can be updated from both sources.
Building a one way data binding can be a lot of work, since event handlers will be required to handle for instance a user typing something in the input field, and updating that value in the model. With two-way data binding the input field can update the model automatically, so time is saved implementing such a trivial feature. This proves a significant advantage over React as React enforces one way data binding.

There are counterarguments to this though. A rule in programming says that explicit is better than implicit, and that doing it in the way of React is better than cleaner than the under-the-hood implementation of Angular's two-way binding. It can also slow down your program as tons of data handling functions will be called once something changes in either the model or the view. And conflicts can arise due to that as well. Creating complex, large and rich UIs can then quickly turn into spaghetti code that slows down and where you will have no idea what is going on, as everything is abstracted behind Angular bindings and directives.

**A project using this technology should be able to be set up quickly, and be able to be extended quickly.**
Angular also offers a CLI (Command Line Interface) that provides a ton of functionality to quickly setup projects, run tests and builds, and add new components with a single command. It can also create a scaffold of whatever new component you need, along with its stylesheets, associated test files, and any other required files.

# Chapter 4

# React

React is a JavaScript library for frontend development. It is not a framework such as Angular which gives a whole MVC architecture to you, but essentially is only the View layer of MVC, allowing you to fill in the gaps yourself with other technology. It is developed and maintained by Facebook.

The thing that sets React apart from Angular is its simplicity and it being lightweight. It also uses JSX which is a syntax that combines HTML and JavaScript to provide databinding and manipulation. In essence a component is now one JSX file in React, instead of a separate HTML and TypeScript file in Angular.

## 4.1 Pros and Cons

In this section and the next the characteristics of React will be tested against the requirements defined in chapter 2.

**The learning curve should not be too high, since learning a new technology will also take a lot of time.**
Because React only defines the View part of a traditional MVC pattern the learning curve will be easier for newcomers. Users can define their own models and controllers in JavaScript and make them as simple as they want, giving users a lot of freedom. Users can also use different libraries or frameworks to do things like HTTP requests, dependency injection and data flow. The syntax for React is also fairly simple. It uses JSX which looks like a mixture between HTML and JavaScript. It does not use a lot of abstractions and creating components is easily done in a single file. React can be expanded upon by adding libraries such as React-router for routing, Flow for type checking and Redux for more complex data handling. This has the added risk of some libraries and frameworks being more complex than expected.

**The technology should be able to be testable easily.**
React claims to be more testable than Angular. As with Angular third party testing libraries can be used.

**The technology should be able to provide a good way of having dynamic pages with lots of changing data.**
As previously explained React enforces one-way data binding. There can be added complexity as React does not define controllers or any fixed way to transmit data between components. Libraries like Redux can be used to create state containers and implement more complex state changing.

**A project using this technology should be able to be set up quickly, and be able to be extended quickly.**
Like Angular, there is a command line tool for React to quickly setup projects. 'Create-react-app' is that tool. It offers a modern setup with no configuration. All the build options are abstracted and cannot be seen unless you explicitly remove the abstraction with a simple command. A package can be added called react-cli that allows the functionality to add more components similar to angular-cli.

# Chapter 5

# Conclusion

In conclusion, Angular offers a complete package with a huge amount of functionality and included packages. It has a steep learning curve but once mastered there is little need to add anything extra to the core MVC functionality.

React on the other hand has a more minimalistic approach, both in code complexity and functionality. This allows for a more modular and simple approach to coding. But will require learning extra libraries to enable routing and complex state management. This has the added risk of some libraries and frameworks being more complex than expected.

Taking the requirements defined in Chapter 2 into account, the recommended approach would be to invest time into learning the Angular framework and use this for the frontend part of the project.

# Chapter 6

# References

https://www.accelebrate.com/blog/two-way-data-binding-angular-2-and-react/
https://www.upwork.com/hiring/development/angularjs-vs-react/
https://react-etc.net/entry/react-gets-official-boilerplate-scaffolding-through-react-cli
https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176
https://medium.com/@mnemon1ck/why-you-should-not-use-angularjs-1df5ddf6fc99
https://reactjs.org/blog/2016/07/22/create-apps-with-no-configuration.html
https://medium.com/@chriscordle/why-angular-2-4-is-too-little-too-late-ea86d7fa0bae
https://www.sitepoint.com/react-vs-angular/
https://hackernoon.com/angular-vs-react-the-deal-breaker-7d76c04496bc
https://reactjs.org/docs/introducing-jsx.html

## 11.3 Sjoerd

### 11.3.1 Java Backend Frameworks

# Java backend research

19-9-2017

**Sjoerd Brauer**
**Fontys Venlo**

Dear FSG1,

In this document 10 Java frameworks will be described. This document will outline the pros and cons of each framework. Based on the criteria and pro's and cons a decision will be made.

Criteria for our project are:

1. Framework must work with PostgreSQL.
2. Framework must support HTTP protocol.
3. Framework must have documentation.
4. Framework must provide tutorials on how to use the framework.

Sincerely,

Sjoerd Brauer
E: s.brauer@student.fontys.nl

**Problem Description**
There are many frameworks one can use for the back-end. Which one is best suited for this project and why?

**Subquestion one** Which framework is best suited for the HTTP protocol.

**Subquestion two** Which framework is best suited for working with back-end communication for this project.

**Spring MVC**
Spring is a full blown MVC java framework.

Supports all criteria?

1. Spring needs an additional framework to connect with the database.
2. Spring has an http communication package: org.springframework.http
3. Spring is well documented, see website in sources.
4. Many tutorials can be found: https://www.tutorialspoint.com/spring/

**Why use it?**
Spring is a big time endured. It's very powerful and can do many many things.

Pros:

1. Simplified injection of test data through the use of Plain old Java object .
2. Enhanced modularity, resulting in better code readability.
3. Loose coupling between different modules.
4. Dependency Injection (DI) flexible use.
5. You can also create a restful interface which can be extended later on by other programmers.
6. You could also use spring security, so that only certain people have access to the rest interface.
7. Easy to set up OATH2.

Cons:

1. Steep learning curve.
2. Very big.

**Sources**
https://spring.io/

**Strut 2**
Struts 2 is a pull-MVC framework. i.e. the data that is to be displayed to user has to be pulled from the Action.

Supports all criteria?

1. Strut2 needs an additional framework to connect with the database.
2. Strut2 uses servlets for http conenction: org.apache.struts2.ServletActionContext
3. Strut2 is well documented, see website in sources.
4. tutorials can easily be found: https://www.tutorialspoint.com/struts_2

**Why use it?**
Strut 2 is a time endured framework. It's very powerful and can do many many things.

Pros:

1. Configurable MVC components, which are stored in struts.xml file. If you want to change anything, you can easily do it in the xml file.
2. POJO based actions. Struts 2 action class is Plain Old Java Object, which prevents developers to implement any interface or inherit any class.
3. Support for Ajax, which is used to make asynchronous request. It only sends needed field data rather than providing unnecessary information, which at the end improves the performance.
4. Whether you want to use JSP, freemarker, velocity or anything else, you can use different kinds of result types in Struts 2.

Cons:

1. Compatibility
2. Limited Documentation.
3. UI driven framework.

**Sources**
https://struts.apache.org/

**akka**
Akka, a set of open-source libraries for designing scalable, resilient systems that span processor cores and networks. Akka allows you to focus on meeting business needs instead of writing low-level code to provide reliable behavior, fault tolerance, and high performance.

Supports all criteria?

1. Akka supports PostgreSQL https://github.com/AkkaNetContrib/Akka.Persistence.PostgreSql
2. Akka supports HTTP protocol: http://doc.akka.io/docs/akka-http/current/scala/http/
3. Akka is well documented, see sources and previous point.
4. Tutorials can easily be found: http://doc.akka.io/docs/akka/2.0.1/intro/getting-started-first-java.html

**Why use it?**
Compared to the other frameworks akka really has a benefit over them with their asynchronous ways. Most useful in a multi threaded environment.

Pros:

1. Multi-threaded behavior without the use of low-level concurrency constructs like atomics or locks, relieving you from even thinking about memory visibility issues.
2. Transparent remote communication between systems and their components, relieving you from writing and maintaining difficult networking code.
3. A clustered, high-availability architecture that is elastic, scales in or out, on demand, enabling you to deliver a truly reactive system.
4. Does not run on Java JEE server.

Cons:

**Sources**
http://akka.io/

**Vert.x**
Eclipse Vert.x is a polyglot event-driven application framework that runs on the Java Virtual Machine.

Supports all criteria?

1. Vert.x supports PostgreSQL connection: http://vertx.io/docs/vertx-sql-common/java/
2. Vert.x supports HTTP protocol: http://vertx.io/blog/some-rest-with-vert-x/
3. Vert.x is well documented see sources.
4. Many tutorials can be found: http://vertx.io/blog/posts/introduction-to-vertx.html

**Why use it?**
Like Akka it's scalable, concurrent, non-blocking but with the exception that it runs on the JVM

1. Non-blocking, event driven runtime
2. Simple to use concurrency and scalability
3. Polyglot (multiple languages can use vert.x)

Cons:

1. less well known.

**Sources**
http://vertx.io/blog/posts/introduction-to-vertx.html

**playframework**
Java Database Connectivity (JDBC) is an application program interface (API) specification for connecting programs written in Java to the data in popular databases.

Supports all criteria?

1. Play framework supports PostgreSQL: https://stackoverflow.com/questions/24124789/install-postgresql-with-play-framework-driver-not-found-org-postgresql-drive
2. Play framework supports HTTP: https://www.playframework.com/documentation/2.5.x/HTTPServer
3. Play framework has good documentation: https://www.playframework.com/documentation
4. There are good tutorials, see url above.

**Why use it?**
Fast multithreaded play work that covers a large area.

1. Reactive. Play is built on Netty, so it supports non-blocking I/O. This means it's very easy and inexpensive to make remote calls in parallel, which is important for high performance apps in a service oriented architecture. It also makes it possible to use server push technologies such as Comet and WebSockets. More info here: Play Framework: async I/O without the thread pool and callback hell
2. Amazing error handling. Play has beautiful error handling in dev mode: for both compile and runtime errors, it shows the error message, the file path, line number, and relevant code snippet right in the browser. No more digging through random log files (Tomcat) and far fewer incomprehensible, gigantic stacktraces (Spring).
3. Java (and Scala). Use reliable, type-safe languages and leverage JVM performance to scale to many users and many developers. Also, you can leverage the huge Java community, strong IDE/tooling support, and tons of open source libraries.
4.

Cons:

1. not well known.
2. Java + Async. Play is built around async I/O, which means writing code that "executes later". Unlike Scala, Java lacks key language features, such as closures, to keep async code clean. There are patterns and tools that make it tolerable, but you end up with lots of anonymous inner classes. For streaming functionality (iteratees, enumeratees), you can't really use Java at all. Also worth mentioning is that lots of existing Java libraries are synchronous/blocking, so you have to be careful with which ones you use in a an async/non-blocking environment like Netty. However, if necessary, you can always configure Play's thread pool to use lots of threads and behave just like any other blocking server.
3. Memory Hog

**Sources**
https://www.playframework.com/ https://www.quora.com/What-are-the-pros-and-cons-of-the-Play-Framework-2-for-a-Java-developer

**Jersey**
Jersey framework is more than the JAX-RS Reference Implementation. Jersey provides it's own API that extend the JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development. Jersey also exposes numerous extension SPIs so that developers may extend Jersey to best suit their needs.

Goals of Jersey project can be summarized in the following points:

Track the JAX-RS API and provide regular releases of production quality Reference Implementations that ships with GlassFish; Provide APIs to extend Jersey & Build a community of users and developers; and finally Make it easy to build RESTful Web services utilising Java and the Java Virtual Machine.

Supports all criteria?

1. Jersey can communicate with PostgreSQL but best in combination with another technology.
2. Jersey supports HTTP. see source.
3. Jersey has proper documentation see source.
4. Jersey has tutorials see source.

**Why use it?**
Jersey implements the JAX-RS api with extra features. Jersey makes it easier to create a restful service on any java application server.

1. Fast and cutting edge
2. Smooth JUnit integration
3. Supports true asynchronous connections

Cons:

1. Jersey 2.0+ uses a somewhat messy dependency injection implemetation
2. A large amount of online resources (3rd party) are related to Jersey 1.X making them unsuitable for Jersey 2.X

**Sources**
https://jersey.github.io/ https://stackoverflow.com/questions/7052152/why-use-jax-rs-jersey http://www.gajotres.net/best-available-java-restful-micro-frameworks/

**spark framework**
Spark - A micro framework for creating web applications in Kotlin and Java 8 with minimal effort

Supports all criteria?

1. Spark has a function to communicate with a PostgreSQL: https://sparktutorials.github.io/2015/04/29/spa and-sql2o.html
2. Spark supports HTTP see source.
3. Spark has documentation see source.
4. Spark has tutorials see source.

**Why use it?**
A small lightweight framework with one purpose, to communicate with front-end

1. Fast and lightweight
2. Excellent for rapid prototyping
3. Easy to setup
4. Most commonly used with AngularJS
5. Real micro-framework

Cons:

1. Documentation could be better, it is not intended for beginners
2. Not suitable for large projects
3. Small community

**Sources**
http://sparkjava.com/ http://www.gajotres.net/best-available-java-restful-micro-frameworks/

**Hibernate**
Hibernate ORM (Hibernate in short) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Supports all criteria?

1. Hibernate is specially made to connect with database, see source for more information.
2. Hibernate does not support HTTP protocol, another framework needs to be used.
3. Hibernate is well documented, see website in sources.
4. Tutorials can easily be found: https://www.tutorialspoint.com/hibernate/

**Why use it?**
Hibernate is used to make database connections easier. For simple use it's perfect.

Pros:

1. Caching mechanism to bug database with similar queries.
2. N+1 or Lazy loading support.
3. Inheritance, encapsulation

Cons:

1. Does not permit multiple inserts
2. Supports less queries then JDBC.
3. Not a good choice for small projects.

textbfSources
http://hibernate.org/

**JDBC**
Java Database Connectivity (JDBC) is an application program interface (API) specification for connecting programs written in Java to the data in popular databases.

Supports all criteria?

1. JDBC is specifically designed to communicate with databases.
2. JDBC does not support HTTP protocol.
3. JDBC is well documented: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
4. Many tutorials can be found, see url of previous link.

**Why use it?**
JDBC is used to connect to databases. It has everything to do what you need.

Pros:

1. You have complete control.
2. Can manipulate large data sets.
3. Great for handling large data transferring.

Cons:

1. Hard to implement MVC concept.
2. Large programming overhead.

**Sources**
https://www.slideshare.net/rajkrrsingh/proscons-jdbc-hibernate-ejb
https://stackoverflow.com/questions/35955020/hibernate-orm-framework-vs-jdbc-pros-and-cons

**OrmLite**
Object Relational Mapping Lite (ORM Lite) provides some simple, lightweight functionality for persisting Java objects to SQL databases while avoiding the complexity and overhead of more standard ORM packages.

Supports all criteria?

1. ORMLite is specifically designed to communicate with databases.
2. ORMLite does not support HTTP.
3. ORMLite has documentation see sources.
4. ORMLite has tutorials see sources.

**Why use it?**
ORMLite is a lightweight ORM framework. Easy and accessible to use.

Supports all criteria?
**Doesn't support criteria 2**
You need to use something else to handle HTTP communication.

Pros:

1. Supports complicated database operations
2. no need to remember to SQL queries
3. prefer for big size application

Cons:

1. unnecessarily increase size of application

**Sources**
http://ormlite.com/ https://stackoverflow.com/questions/25424679/ormlite-or-sqlite-in-android

For additional details see corresponding framework in the research

| Framework | PostgreSQL | HTTP | Documentation | Tutorial |
|---|---|---|---|---|
| Spring MVC | | x | x | x |
| Strut 2 | | x | x | x |
| Jersey | | x | x | x |
| Akka | x | x | x | x |
| Vert.x | x | x | x | x |
| Play Framework | x | x | x | x |
| Spark | x | x | x | x |
| Hibernate | x | | x | x |
| JDBC | x | | x | x |
| ORMLite | x | | x | x |

### 11.3.2 PDF Generation

# PDF Research

## Goal

The goal of this research is to find a solution to generating a module's PDF for frontend. The research should point out if the solution is adequate to the customers requirements or not.

## Customer requirements

1. Should be possible to generate a module as PDF
2. Should be possible to generate a shortened module description
3. Should be possible to generate all modules at once
4. Should be possible to generate shortened versions of a module description at once
5. Generating PDF's is browser independent

## Frontend PDF technologies researched

- jsPDF
- PDF.js
- PDFMake
- Native
- texlive.js

## General Findings

Criteria 3 and 4 make things much harder on the frontend side. It's possible to generate all PDF's frontend. That would require retrieving a lot of data and calculations. All this effort which needs to be done frontend will increase stress on the frontend server which in turn might react slower.

Two approaches are found in creating a solution to the problem;
1. What you see is what you get
2. Template driven

What you see is what you get means that what is displayed as module is what will become the PDF. Template driven means that a template is used for creating a module.

There was also another problem found using these technologies which added complexity to create a PDF frontend. It is not possible to directly translate the html/css to pdf because angular code is not executed with these technologies.

## Technology findings

### jsPDF

jsPDF is a open source javascript framework created by James Hall. This library is ideal in many situations and is quite flexible. It meets all requirement. It could even use a what you see is what you get approach. However it has problems with translating angular code. You could still use this framework for a template driven approach.

The template driven approach needs to be calculated. Things like variable text sizes increases the complexity of this solution a lot. So altough it meets al requirements it takes an considerable amount of time to get a proper PDF.

Met Criteria

| | |
|---|---|
| 1. Should be possible to generate a module as PDF | X |
| 2. Should be possible to generate a shortened module description | X |
| 3. Should be possible to generate all modules at once | X |
| 4. Should be possible to generate shortened versions of a module description at once | X |
| 5. Generating PDF's is browser independent | X |
| What you see is what you get approach | |
| Template driven approach | X |

### PDF.js

This technology does not fulfill any of the requirements. PDF.js is only usefull for rendering PDF's properly on a browser. So altough this technology is not useful for this scenario. It could still be used in the future depending on the needs.

## PDFMake

PDFMake Is another open source PDF library for javascript founded by Bartek Pampuch. This framework is very similair to jsPDF and faces the same difficulties as jsPDF. They use abit of a different approach. The main difference is that PDFMake is better documented.

| | |
|---|---|
| 1. Should be possible to generate a module as PDF | X |
| 2. Should be possible to generate a shortened module description | X |
| 3. Should be possible to generate all modules at once | X |
| 4. Should be possible to generate shortened versions of a module description at once | X |
| 5. Generating PDF’s is browser independent | X |
| What you see is what you get approach | |
| Template driven approach | X |

## Native

Unlike the other technologies. Native is not a javascript framework but the capabilities of ones browser. With native technologies (css). It is possible to a PDF looking like a web page. It’s possible to hide extra information and using checkboxes makes it easy to generate a shorthened module description. The problem of using this approach is that it is heavily browser dependant. Chrome and Edge seem to properly support this approach. Mozilla and safari have less options and can ignore certain parts of the CSS. Solving criteria 3 and 4 is also very difficult using this approach.

| | |
|---|---|
| 1. Should be possible to generate a module as PDF | X |
| 2. Should be possible to generate a shortened module description | X |
| 3. Should be possible to generate all modules at once | |
| 4. Should be possible to generate shortened versions of a module description at once | |
| 5. Generating PDF’s is browser independent | |
| What you see is what you get approach | X |
| Template driven approach | |

### texlive.js

Texlive.js is yet another javascript framework that is capable of creating PDF documents frontend. Author of this framework is Manuel Scholling. Using this technology you don't need to do the calculations to get everything at the right position as is the case with jsPDF and PDFMake. Using texlive.js you take on a latex approach of creating a PDF frontend.

| | |
|---|---|
| 1. Should be possible to generate a module as PDF | X |
| 2. Should be possible to generate a shortened module description | X |
| 3. Should be possible to generate all modules at once | X |
| 4. Should be possible to generate shortened versions of a module description at once | X |
| 5. Generating PDF's is browser independent | X |
| What you see is what you get approach | |
| Template driven approach | X |

## Conclusion

As one might notice there are a lot of options to create PDF's frontend. For the current problem however generating PDF's frontend might not be the ideal solution with the given input of the customer. The most suitable technoloy for our use case will be texlive.js We started this research without having the knowledge of criteria 3, 4 and 5.

For proof see log: 2017-12-12 Daily Meeting in which is implied the research already started (which it did on 11-12-2017). Later that day we asked the customer to discuss possible solutions and we wrote this down in the 2017-12-12 Daily Meeting log as well.

## 11.4 Tobias

# Document title
Fancy subtitle

Firstname Lastname

*Fontys Venlo Techniek en Logistiek*
*Informatics*
*Module Applied Research (APPL)*

Venlo, 11th October 2015

# Summary

Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# Contents

# Chapter 1

# Introduction

Sample citation:
(Stroustrup, 2015)
(Wolf, 2017)
(Stroustrup, 2015)
(Einstein, 1905)
(Goossens et al., 1993)
(Knuth, 2017)

# Acronyms

| | |
|---|---|
| **JSF** | Java Server Faces |
| **JEE** | Java Enterprise Edition |
| **MVC** | Model-View-Controller |

# References

Einstein, A. (1905). Zur elektrodynamik bewegter koerper. (German). *Annalen der Physik*, 322(10):891–921.

Goossens, M., Mittelbach, F., and Samarin, A. (1993). *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts.

Knuth, D. (2017). Knuth: Computers and typesetting.

Stroustrup, B. (2015). *The C++ programming language*. Addison-Wesley, 20 edition.

Wolf, M. (2017).