Quality Assurance

IT2901 Informatics Project II
Project report, final delivery
May 21, 2012

Group 8:

Anders Palfi, Astrid Kløve-Graue, Daniel Tandberg Abrahamsen
Haakon Sønsteby, Mathilde Oftedal, Thomas Iversen

NTNU – Trondheim
Norwegian University of
Science and Technology

Preface

This document is the project report established in course IT2901 [1], Informatics Project II at Norwegian University of Science and Technology, during the spring of 2012.

The development team would like to take this opportunity to thank the following for their contributions:

- The customer, **Agency of Public Management and eGovernment (Difi)** [2] for the challenging and interesting assignment.
- The customer contact **Erlend Klakegg Bergheim** for being available, explaining questions about the assignment and general guiding along the way.
- The supervisor **Anh Nguyen Duc** for his follow-up of the team and especially the feedback on deliveries, in addition to guidance of the development process.

[1] http://www.ntnu.edu/studies/courses/IT2901/2011
[2] http://www.difi.no/

Abstract

The assigned task was to develop a web based tool to help the Agency of Public Management and eGovernment (Difi) evaluate public web pages. The development team consists of students from NTNU, studying for their Bachelor in Informatics. The application will use MySQL and Java Persistence API to manage and manipulate the database. Drupal and PHP are used to create the graphical user interface, and the REST architecture for handling request and responses between the database and GUI. After requirements from the customer the team will write all back-end code using Java EE6 with Glassfish as the application server.

This report discuss the requirements to the new application, describes the work that has been done and documents the design- and implementation decisions. This report along with the source code presents a solution to the assignment.

Anders Palfi	Astrid Kløve-Graue
Daniel Tandberg Abrahamsen	Haakon Sønsteby
Mathilde Oftedal	Thomas Iversen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem description

Every year the Agency of Public Management and eGovernment (Difi, Appendix A) works to increase the quality of public and government websites [1]. A main part of this work is the annual assessment of quality, which means to go through the process of evaluating over 700 public websites. The criteria set contains 33 indicators in the field's accessibility, user adjustments, and how useful the content is.

Difi would like the team to optimize the process through the course IT2901 Informatics Project II, at NTNU spring 2012. This is a mandatory course in the Bachelor's degree in Informatics, which aims to create a software system for a customer.

The team will develop an application that will use MySQL and Java Persistence API to manage and manipulate the database. Drupal and PHP (Appendix A) are used to create the graphical user interface, and the REST architecture for handling request and responses between the database and GUI. According to the requirements from the customer the team will write all back-end code using Java EE6 with Glassfish as the application server.

[1] http://kvalitet.difi.no/

1.2 Description of requirements, high-level

The two main requirements for this project are to deliver a REST server interface, which must be applicable for further integration with other systems within the domain of Difi's evaluation of websites, and an interface made in Drupal, which use the REST server interface. The finalized product needs to satisfy the functionality needed by an evaluator to both easy and thoroughly document his/her evaluation of a website.

1.3 Motivation

Each team member is highly motivated, and is looking forward to work on this project. Both due to the importance of solving the exercise (developing a system) as a team, the practical experience before individual employment where a real customer is involved, as well as the project will require each member to acquire new knowledge.

It is an important job to evaluate public web pages, almost every inhabitant in Norway visits public web pages several times a year. Therefore it is essential that the sites are quality assured. All the team members are interested in the quality of web pages – for instance that sites are easy to navigate and user friendly. By making a system to improve the efficiency of this process the team feel that they are a part of the quality assurance.

1.4 Customer

The Agency of Public Management and eGovernment (Difi) is a state department who has a central role in development of Norwegian public administration, and ensuring that the public administration is characterized by quality, efficiency, user-orientation, transparency and democracy, and that it is well organized and managed. Difi aim to develop the organization and leadership in the public sector with coordination among public authorities and services.

The customer contact from Difi is Erlend Klakegg Bergheim. He graduated from NTNU, Department of Computer and Information Science, in 2010. He can be contacted through his email: erlend.klakegg.bergheim@difi.no.

Several team members are familiar with the customer through NTNU. Difi has held several courses for the student organization Online and one member of the team has worked at Difi's summer internship. As a result of this the dialog between the team and the customer was good from day one. During the first meeting with

the customer it was decided to have weekly meetings. This was a decision based on the customer's wish, and the team's preference.

More detailed information can be found in the minutes of meeting, Appendix C. The Iterations chapter will also discuss the interaction with the customer.

1.5 The team

The team consists of six students, all studying for their Bachelor's degree in Informatics at NTNU. All members have experience from smaller projects in connection with courses at NTNU: this includes system development, Java, MySQL and the development method Scrum. In addition some of the students have experience from summer internships where they have worked with technologies like Java EE, MS SQL, C#, .NET, C++, Maven, and GlassFish.

1.5.1 Contact

For questions or comments on the report or the product, please contact us:

Anders Palfi	Contact: palfi@stud.ntnu.no
Astrid Kløve-Graue	Contact: klovegra@stud.ntnu.no
Daniel Tandberg Abrahamsen	Contact: danielab@stud.ntnu.no
Haakon Sønsteby	Contact: haakonsv@stud.ntnu.no
Mathilde Oftedal	Contact: mathilof@stud.ntnu.no
Thomas Iversen	Contact: thomaive@stud.ntnu.no

1.6 Supervisor

The team's supervisor is Anh Nguyen Duc. He is a PhD candidate from NTNU, Department of Computer and Information Science. He can be contacted through his email: anhn@idi.ntnu.no.

During the first meeting with the supervisor it was decided to meet every other week. The meetings would mainly be about the process and how the team work together. Anh Nguyen Duc assured that the team could come to him if problems with the customer occurred or if the team had questions about the project report.

More detailed information can be found in the minutes of meeting, Appendix D. The Iterations chapter will also discuss the interaction with the supervisor.

1.7 Project report structure

Chapter 2, Pre-study

Chapter 2 discuss the technologies and development tools the team decided to use during the development of the project. As well as information about alternative development frameworks.

Chapter 3, Project planning

Chapter 3 discuss the team's roles in the project and the chosen development. In addition this chapter shows the project work scope, time scope and the architectural design.

Chapter 4, Requirements

Chapter 4 discuss the functional and non-functional requirements from the customer, how they are prioritized, and textual use cases that reflect the requirements.

Chapter 5, Testing

Chapter 5 discuss the test plan for the project, which involves how the team plan to execute unit-, integration, system-, and usability testing. As well as the test results and the test evaluation.

Chapter 6, Iterations

Chapter 6 discuss a summary of the different iterations that have been completed during the project. This includes the backlog of each sprint, project management, the result of the sprint, and the product backlog at the end of each sprint.

Chapter 7, Closure

Chapter 7 sums up the team's conclusions, lessons learnt and a discussion of the result.

Chapter 2

Pre-study

This chapter will discuss the different technologies and development tools the team decided to use during the development of the project. This includes information about the alternative development frameworks.

2.1 Technical solutions

The customer required Java EE and Drupal for the project. The books "Beginning Java EE 6 Platform with GlassFish 3" (Antonio Goncalves, 2009) [2], and "Pro Drupal 7 Development" (Todd Tomlinson and John VanDyk, 2010) [11] were suggested for pre-study.

"Pro Drupal 7 Development" had a lot of example code for creating Drupal modules with PHP code. The team needed to understand Drupal's structure and syntax to create modules. Figure 2.1 shows the covers of the books.

The books gave the team a lot of tips and tricks, and this knowledge made it easier to look for additional information later on.

2.1.1 Drupal

Drupal[1] is an open-source content-management platform written in PHP that is commonly used to set up and manage websites. It has many helpful features like account registration and maintenance, menu-management,

[1] http://www.drupal.org/

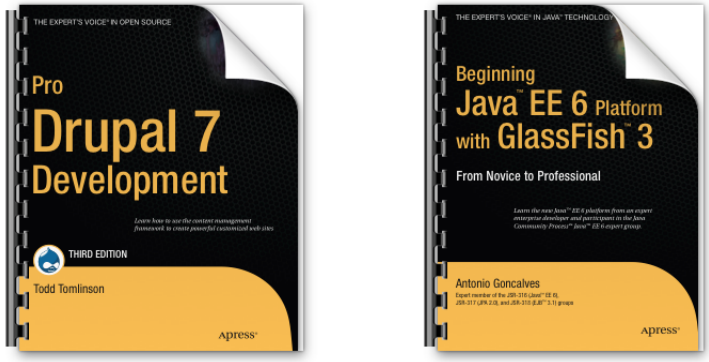


Figure 2.1: Books suggested by the customer

RSS-feeds, page-layout customization and system administration. Additional functionalities can be extended through the many community-contributed add-ons or developers can make their own. Drupal will be used because it is a free and powerful tool that gives the team complete control over the website, in addition to all the features it supports. Drupal was highly recommended by the customer, and also a requirement. Figure 2.2 shows Drupal's logo.

Figure 2.2: Drupal's logo

2.1.2 REST

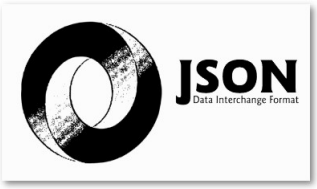
REST (Representational State Transfer)[2] is an architecture style, which uses HTTP methods and URIs as an API for doing different tasks on "resources" on the web. Although the REST architecture uses many standards like HTTP, XML and URL it is not a standard, but a style.

The Web contains "resources". When a client accesses a resource through a URL it gets a representation of the resource, putting the client in a "state". The result of the client accessing another resource will put the client in another state. The client "transfers" state with each representation. This is where the name Representational State Transfer comes from. REST is the architecture style of the Web, and it is often used in computer-to-computer communication. REST architecture is as mentioned a requirement from

the customer. The flow chart in Chapter 3.11 shows how REST is connected to the different parts of the application.

2.1.3 JSON

JSON (JavaScript Object Notation) [2] is a lightweight data interchange format that easily can represent Java objects. JSON will be used as the interchange format between Drupal and the REST server. Some of the reasons why JSON was chosen over XML are because XML is not as suited for data interchange as JSON. JSON is simpler and much easier for humans to read than XML. Figure 2.1.3 shows JSON's logo 2.3(a) and a JSON object from the project 2.3(b).



(a) JSON's logo

```
{"evaluationSet":
    [{
        "active":"true",
        "created":"1335956991479",
        "criteriaSet":"2",
        "evaluationGroup":"2",
        "id":"2",
        "subject":"peter pan med crew"
    }]
}
```

(b) JSON object from the project

Figure 2.3: JSON logo (a) and an example of a JSON object from the project (b)

2.1.4 Java Persistence API

Java Persistence API (JPA) [2] offers a way to handle the mapping of object relationships in Java. JPA let you assign relational data to persistent Java objects. The relational data is stored in a relational database and can be retrieved at any time. JPA will be used to get information from the database and place it in objects, and also to place new information in the database. JPA supports it's own query language called Java Persistence Query Language (JPQL), and therefore JPQL will be used instead of traditional SQL queries.

[2] http://www.json.org/

2.1.5 MySQL

MySQL [3] is a well-known relational database management system (RDBMS) that is commonly used to keep information within web applications. MySQL is open source and will be used as the database system to manage stored data. As a graphical interface to handle the administration with MySQL the team will use phpMyAdmin. [4]. Figure 2.4 shows MySQL's logo.

Figure 2.4: MySQL's logo

2.2 Development framework

There are many different development frameworks. The key to a successful project is to choose the most suitable development framework for your project. The different development framework have their own strengths and weaknesses. To choose the most suitable framework, a set of factors must be considered. This might be team size, previous experience and customer involvement.

2.2.1 Waterfall method

The waterfall model [12] is a sequential software development process. The process is seen as a waterfall flowing downwards through the different phases. These different phases are described below, and illustrated in figure 2.5.

1. **Requirements**

   This first step consists of gathering information about what the customer want the product to solve. This includes understanding the context and constraints of the customers business, which functions the product must have, and other requirements from the customer. The information in this analysis usually results in a formal requirement specification.

[3] http://mysql.com/
[4] http://www.phpmyadmin.net/

2. **Design**

   This step consists of figuring out technical solutions and development tools the system designers should use. This involves everything from selecting the hardware, defining software architecture, choose database management system, to figure out which programming language to use. User interface design is also a part of this step.

3. **Implementation**

   This step consists of the implementation of the product based on the requirement specification and the design specifications. A development team of programmers, interface designers and other specialists typically implements the system. The result of this step is one or more product components.

4. **Verification/testing**

   This section consists of verification and testing of individual components and the integrated system. Testing is an important tool to check if the components are error free, and meet the requirements from the requirement step. An independent quality assurance team will go through the system, creating "test cases" to check if the system fulfills the requirements. Usually three types of test are executed: unit testing of individual code modules, system testing of the integrated product, and acceptance testing. If there are defects, they will be reported back to the implementation team to be corrected. In this step product documentation, such as a user manual, is published.

5. **Maintenance**

   This step comes after the customer has installed the system, and consists of making modifications to the system, or to a component, to for instance improve the performance. Reasons for modifications can be a request from the customer or defects discovered during use of the system.

This is an example of the simplest waterfall model, but the model can be extended to have additional phases. The model is designed in such a way that the project should only move to the next phase when its preceding phase is completed and perfected. Since every step must be completed, the model promotes discipline for the system developers. Because the requirement- and design step comes before the actual implementation the project waste minimal of time and effort, and reduces the risk of deviations from the project plan. This also improves quality: it is much easier to discover possible flaws at the design phase than the testing phase. The waterfall model is also often criticized because the requirement phase is so early in the process. The customer may not know exactly what they want in the beginning of the project. This also makes the estimating of time and costs difficult. Therefore this model is only recommended for projects that are relatively stable, and where customer requirements can be clearly identified at an early stage.

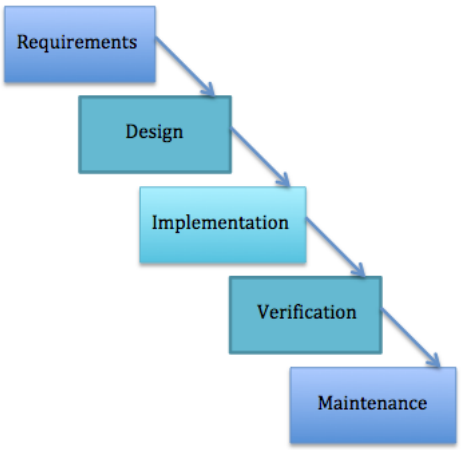


Figure 2.5: The waterfall method's phases

2.2.2 Spiral model

The spiral model [3] is a software development process that combines elements of both design and prototyping-in-stages. The model combines the waterfall method with prototype testing. It is designed as a spiral, as the figure 2.6 [5], where you start in the inner center of the spiral and move around the bigger circle. The spirals represent iterations, and the spiral model was the first development method that described why iterations work. The project can have any number of loops, according to the project. Each round in the spiral the project goes through four phases:

1. **Determine objectives**

   In this phase the development team tries to figure out product goals, alternatives in design and restrictions imposed due to cost, technology and schedule. A number of users are interviewed which represents the internal and external users and other aspects of the project. This results in a typical requirement specification.

2. **Identify and resolve risks**

   In this phase the development team discuss other possible solutions for the implementation that still fulfill the customer's requirements. A risk analysis or a problem analysis is created for each solution, and the evaluation determines future action. Sometimes there is need for a prototype to clarify the risks and problems. This is usually a scaled-down system that represents an approach of the properties of the final product.

3. **Development and test**

[5] http://blog.hydro4ge.com/waterfalltoboehm/

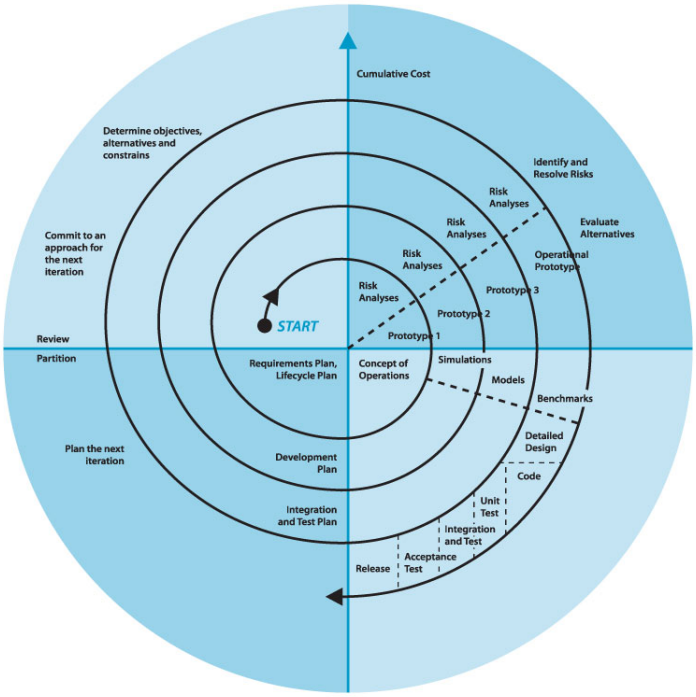


Figure 2.6: The spiral method's phases

In this phase the detailed requirements are determined and the implementation begins. It is possible to implement waterfall or an incremental approach to help the development. Testing is also done in this phase.

4. **Plan the next iteration**

   In this phase the next iteration is planned. Here the customer is given an opportunity to analyze the results from the previous phase and give feedback to the development team.

The spiral model is best suited for large, complicated and high risk projects. It is a flexible model since the project manager can determine the development phases according to the complexity of the project. It is easy to monitor the project because several concerned people, like the customer, must review each phase and loop. If the customer wants a change, it can easily be introduced late in the project. Because each loop involves estimating, the time and cost estimates becomes more and more accurate as the development moves forward. The costs of projects using the spiral model often become very high because the development continues until the customer is satisfied. If the customer wants more requirements the requirements are added in another loop. For project with a clear requirement specification the spiral model often is too complicated and uses unnecessary time during the loops.

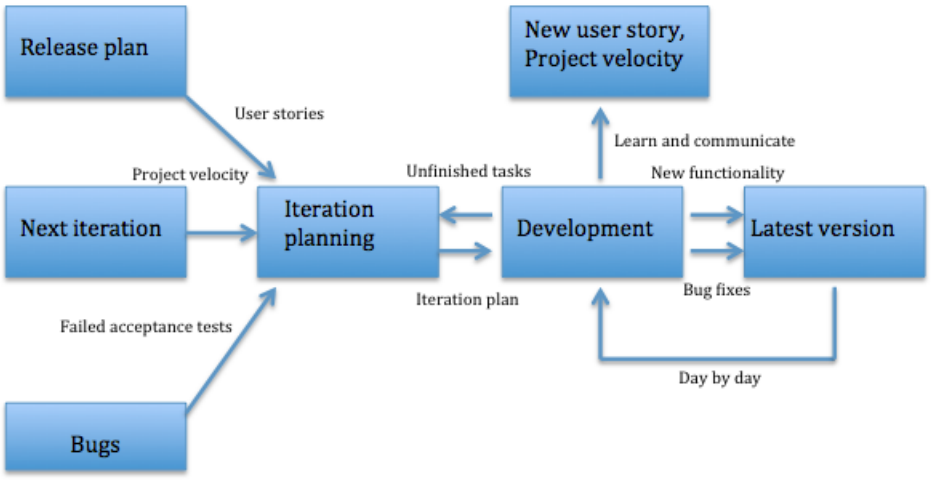


Figure 2.7: An iteration using extreme programming

2.2.3 Extreme programming (XP)

Extreme programming [4] is a software development methodology with goal to improve software quality and responsiveness to changing customer requirements. Extreme programming is a type of agile software development, therefore it has small development cycles. One cycle may last for two to three weeks depending on the size of the project. Figure 2.7 represent an iteration using Extreme programming. It is normal in extreme programming to start with the simplest solution and then add extra features and functionality later. This methodology does not use a lot of time on future requirements before they become relevant. Testing is a big part of extreme programming: If a little testing can eliminate a few flaws, a lot of testing can eliminate many flaws. All the team members are equal partners in a collaborate team, this includes the customer and managers as well as the developers. It is also very common to work in pairs when programming.

Extreme programming [7] use five different ways to improve a project:

1. **Communication:** constantly communicating with the customer and the development team.
2. **Simplicity:** the development team always keeps their code and design simple and clean.
3. **Feedback:** through testing the different software components from day one, the development team gets constant feedback. The system is delivered to the customer as soon as possible in case the customer wants changes. This way the requirements can be implemented continuously.
4. **Respect:** every small success deepens the respect for each team member.

5. **Courage:** this way the development team courageously can handle changes in the requirements and the technology.

Extreme programming consists of simple rules that works like a jigsaw puzzle. There are many small pieces that individually do not make sense, but when combined together you can see the whole picture. The rules are:

- **Planning:** consist of writing user stories, planning and scheduling the iterations and releases.
- **Managing:** consist of setting up the team with a workspace, stand up meetings every day, measure the project velocity, and fix Extreme Programming if it breaks.
- **Designing:** the key word in this rule is simplicity. Here the team must choose a system metaphor, and refactor whenever and wherever it is possible.
- **Coding:** it is important that the customer always is available in case of questions about the implementation. The code must be written as the agreed standards say, through pair programming. Only one pair can integrate their code at a time, and this should happen often. Create the unit test before the actual coding.
- **Testing:** all the code must have unit tests, and all code must pass the unit tests before it can be released.

Some of the advantages of extreme programming are that it focuses on the code, and not unnecessary paperwork and meetings. It provides a social atmosphere and more opportunities to learn new skills through pair programming. Extreme programming creates working code fast, with few defects, and to a low cost. Unfortunately extreme programming is hard to carry out. It is often difficult to get all the programmers to agree in the best practices, and the customer may not like to be as involved as the method implies.

2.2.4 Scrum

Scrum [9] is an agile software development method for managing software projects and developing software applications. This method encourages to work incremental and iterative, and that interdisciplinary teams execute the development job. Scrum is mainly used to develop software systems and is often used in combination with extreme programming. The requirements from the customer are placed in the product backlog. A project is typically divided into multiple iterations, called sprints, lasting from one week to a month. Before each sprint, there is a sprint planning meeting. Here the planning of the next sprint is done. It consists of two parts where the questions below are answered:

- What will be delivered as a result of the next sprint?
- How will this work be achieved?

It is normal to have small scrum meetings, called daily sprints or stand-up, every day where every team member informs the others about what they have done since last meeting, what they shall do to next meeting and talk about any problems along the way. It is normal to use a Scrum board to keep track of the tickets/tasks (Appendix A) during the sprint. The board is usually divided into three columns: to-do, in progress, and done. All the tickets are estimated through the process called scrum poker (how the Scrum team solves this varies). A development team member uses the Scrum board by selecting the ticket with the highest priority, and moves it from "to-do", till "in progress". When a ticket is completed it is moved to the "done" column. After each sprint a sprint review meeting is held to inspect the tickets that are done. This is an informational meeting with focus on feedback and will foster collaboration. Sprint retrospective is held to give the development team an opportunity to evaluate the previous sprint, and focuses on improvements for the next sprint. Figure 2.8 represent the ticket flow using Scrum.

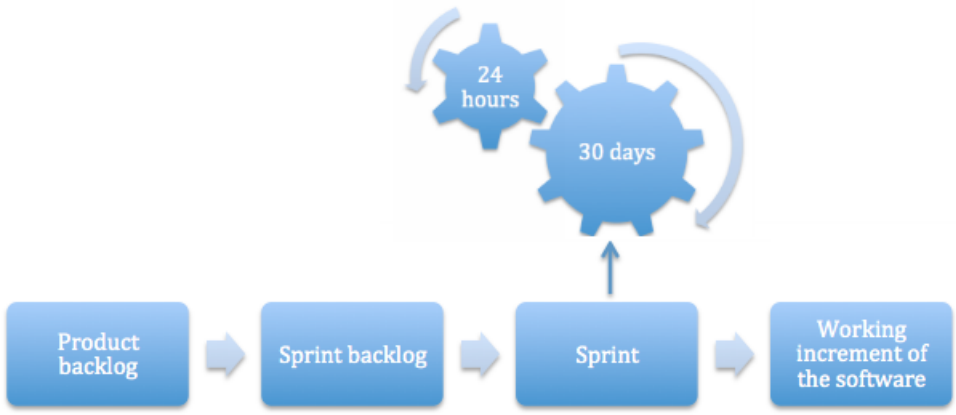


Figure 2.8: The Scrum method

The scrum team is cross functional and consists of:
Product owner is responsible for maximizing the value of the product and the work of the development team. This person is the only one responsible for handling the product backlog. He/she represents the customer's decisions when the customer himself is not present. The product backlog management consists of:

- Make sure that the product backlog items are clearly expressed.
- Prioritize the items to best achieve goals and missions.

- Ensure that the work the development team does is of best value.
- Ensure that the product backlog is visible, transparent, and clear to all, and will show what the development team should work on next.
- Ensure that the development team understands the items in the product backlog at the level needed.

The development team consists of professionals on the subject. The team must be small enough to remain nimble and large enough to complete significant work. Preferably a size between 3-5 people, larger than 9 people requires too much coordination.

The Scrum master is the leader of the Scrum team, and responsible for making sure Scrum is understood and implemented. This is conducted through the Scrum theory, practices and rules.

Scrum provides open communication, everyone in the development team knows what everyone is working on. It is a method that can save time and money, team efficiency can increase with within 20% when using Scrum [6]. Because of small iterations, Scrum requires constant feedback from the customer and the user, and it becomes easier to cope with changes. Disadvantages with Scrum are that decision-making is entirely in the hands of the development team. It only works on small teams, and Scrum requires that the product owner is involved in the whole cycle. The sprints are planned according to requirements, and for each sprint it is a risk that the customer wants to add new features.

2.2.5 Kanban

Kanban [8] literally means signboard (in Japanese), and is another method for developing software products. The Kanban method is designed to use just in time production, and is based on three basic principles:

Start with what you do now

Kanban has no specific roles or steps, and there is no such thing as the Kanban software development process or project management method. Kanban does not impose the development team to change their progress. In Kanban the development start with the roles and processes the team have and stimulates continuously, gradual and evolutionary changes to the current system.

Agree to pursue gradual, evolutionary changes

To get the implementation flowing as desired, the development team must agree that continuously, gradual and evolutionary changes is the way to make the changes in the system.

[6] http://www.brighthub.com/office/project-management/articles/2042.aspx

Respect the current process, roles, responsibilities and titles

Often the development team currently has some elements that work acceptably, and are worth preserving. By agreeing to respect current roles, responsibilities and job titles the team eliminate initial fears.

It has been identified five core properties that have been observed in successful use of Kanban, known as the "Principles of Kanban" [8]:

1. **Visualize the workflow**

   It is important to understand how the workflow functions to make the right decisions. Therefore it is a good tool to visualize the workflow, often conducted a wall of cards divided in different columns: Split the work into pieces, write each item on a card and put it on the wall. The wall evolves into the Kanban task board.

2. **Limit work-in-progress**

   It is important to limit the tasks created in the principle above. This is often done by limit the amount of tasks that can be in one column at the time. New tasks cannot be "pulled", started, until the tasks in progress are moved to another column. See figure 2.9 [7].

3. **Manage flow**

   It is important to monitor and report when each task flow through the task board. The speed and smoothness of the task movement is interesting, and fast smooth movement is ideal.

4. **Make process policies explicit**

   It is difficult to have a discussion about improving a process if the process is not explicit. Therefore it is important to have an explicit understanding to achieve more rational, empirical and objective discussion of issues.

5. **Improve collaboratively (using models & the scientific method)**

   If the team have a shared understanding of the theories about work, workflow, process, and risks, the team is more likely to have a shared understanding of the problem. Then a product can be created, where everybody agrees on is the correct solution. This can be achieved by using different models, for example "The Theory of Constraints" [8].

Some of the benefits of Kanban are that it reduces overproduction and costs. Bottlenecks become clearly visible, and the whole team must collaborate to optimize the flow. Kanban provide an agile solution without necessary having to commit to time-boxed iterations, like sprints in Scrum. The principles around Kanban

[7] http://blog.crisp.se/2009/06/26/henrikkniberg/1246053060000

[8] http://en.wikipedia.org/wiki/Theory_of_Constraints

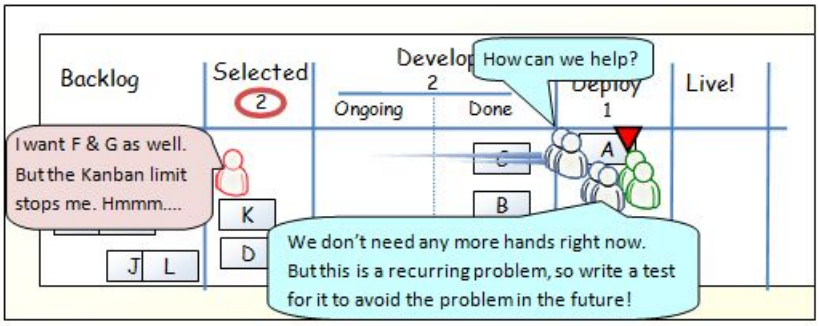


Figure 2.9: A figure of the Kanban board

tend to spread to other departments, which increase visibility of everything that is going on in the company. Kanban does not focus on estimation, which can make it hard to estimate delivery dates and the pace of the development. Kanban requires more discipline from the development team since there are no clear iterations.

2.3 Development tools

2.3.1 Git

The customer wanted the team to use Git [9] as the version control system. None of the team members had any experience with Git, and information had to be gathered. The team got started with git by reading the book "Getting Good With Git" (Burgess, 2010) [1]. The book holds a lot of basic information about Git, and gave a good introduction. The team also arranged a personal course held by a fellow student with Git experience.

Git will be used as the version control system, and the repository will be connected to the customers account at GitHub [10]. Git will ensure that a copy of the project always will be saved locally by every team member (Git is distributed) and it opens the possibility to go back in versions if anything should go wrong at any time.

[9] http://git-scm.com/
[10] http://github.com/

(a) Git's logo (b) GitHub's logo

Figure 2.10: Git- and GitHub's logo (a) and (b)

2.3.2 Eclipse

All team members were free to choose their own preferred IDE. Eclipse [11] was the natural choice for the whole team, due to several courses and projects on NTNU where this IDE were used. Eclipse is a powerful multi-language environment for software development. It consists of an integrated development environment and an extensible plug-in system. The Eclipse Java development tool (JDT) is included in the integrated development environment.

Figure 2.11: Eclipse's logo

2.3.3 Maven

Maven [12] will be used to build and manage the project. This gives several possibilities when compiling, and ensures a fast build. Maven also makes it easier to include all the necessary jars and libraries, which makes

[11] http://www.eclipse.org/
[12] http://maven.apache.org/

it more convenient for each team member to implement new components to the existing system. To include the project and generate Eclipse IDE files the team will use the Maven Eclipce Plugin [13]. Maven was highly recommended by the customer.

Figure 2.12: Maven's logo

2.3.4 Java Platform, Enterprise Edition (Java EE6)

Java EE6 [14] is one of the most widely spread technology to write enterprise-class applications. It consists of Enterprise Java Beans 3.1, Java Persistence API 2.0 and the Java API for RESTful web services, JAX-RS. The customer requested the use of Java EE6. Java EE6 is well suited for developing enterprise applications. The team has used the book 'Beginning Java EE Platform with Glassfish 3' (Antonio Goncalves) [2] to get an introduction to Java EE, RESTful web services and JPA.

Figure 2.13: Java's logo

[13] http://maven.apache.org/plugins/maven-eclipse-plugin/
[14] http://www.oracle.com/technetwork/java/javaee/overview/index.html

2.3.5 Glassfish

Glassfish [15] is an open source application server that was recommended by the customer. Glassfish is the reference implementation of Java EE and supports all the Java application programming interfaces used in this project, such as Java Persistence API 2.0, Enterprise JavaBeans 3.1 and RESTful web services. This makes it possible to create enterprise applications that are both portable and scalable. The team will use the GlassFish v3.

Figure 2.14: Glassfish's logo

2.3.6 JIRA

JIRA [16] is a system for tracking issues in a project. It keeps a good overview of the tasks that are in the backlog, which are in progress, and which are finished. This web-based program will be used in connection with Scrum and the plugin Greenhopper. The team chose JIRA for tracking issues because of a team member's good experiences with this system.

Figure 2.15: JIRA's logo

[15] http://glassfish.java.net/
[16] http://www.atlassian.com/software/jira/overview

2.3.7 Cacoo

Cacoo [17] is a user friendly and useful web based drawing tool to create wireframes, UML diagrams, flowcharts, site maps, mind maps, network diagrams and others. The team selected Cacoo because of its real-time collaboration possibilities and easy sharing between team members and will use this to create various UML diagrams such as class diagram, entity relationship diagram and flow charts. One team member has used Cacoo in an earlier project with good experiences.

Figure 2.16: Cacoo's logo

2.3.8 Dropbox

Dropbox [18] is a free service that allows the user to upload different files into their own space on Dropbox. Any file saved on Dropbox will automatically be available to all the collaborating users accounts. Through Dropbox the team can easily share documents and pictures through a shared folder for the project report.

Figure 2.17: Dropbox's logo

2.3.9 LaTeX

LaTeX [19] is a document preparation system and document markup language for the TeX. LaTeX provides a high-level language that uses the power of TeX. TeX is a typesetting system that produces high-quality text

[17] http://cacoo.com/
[18] http://www.dropbox.com/
[19] http://www.latex-project.org/

documents using a reasonable amount of effort, and to provide a system that would give the same results on every computer. Since TeX is low-level, LaTeX have gotten popular because it provides a system that is easier to use.

Figure 2.18: LaTex's logo

Chapter 3

Project planning

This chapter will discuss how the team plans to conduct the application. This includes the team organization, the chosen development method, the project work- and time scope and architectural design. Screen shots of the prototype are also included.

3.1 Team organization

Haakon: Leader
Mathilde: Vice chairman, Customer relations
Daniel: Scrum-master
Anders: Development manager
Thomas: Test manager
Astrid: Product owner, Report manager

3.1.1 The roles

Here follows is a description of the team members roles and responsibilities. Figure 3.1 show the organization chart for the team hierarchy.

Leader: The leader's main task is to ensure overall progress and making sure deadlines are kept. The leader will also be the main contact point with the supervisor and convene meetings with the team. Haakon got

the leader position because of his focus on early initiating structure in the project and his ability to create a good working environment for the rest of the team.

Vice chairman: The tasks of the vice chairman will be to take the leader's responsibility when the leader is not present. Mathilde is the vice chairman because she is responsible, and likes to have an overview of the project.

Customer relations: The task of the customer relations is to be responsible for keeping contact with the customer and schedule meetings. Because Mathilde worked for the customer, it is natural that she has this responsibility.

Scrum-master: The tasks consist of leading scrum meetings prior to and after each sprint. New tasks will be evaluated and the team will try to estimate the time needed to complete them. The scrum master will in addition have an overview of the scrum process. Daniel got the role of scrum-master in this project due to his experience with scrum the previous summer, his leader role in a committee in the student association and his knowledge about the chosen scrum process tool: JIRA. In addition to the technical foundation that this roles requires he had the positive attitude towards doing scrum in a rightful way.

Development manager: The tasks of the development manager is to keep track of the development process and take necessary actions if needed to maintain the planned progression throughout the project. Anders got chosen as development manager because his ability to quickly acquire new knowledge, especially within technologies.

Test manager: The tasks of the test manager are to make sure that tests will be performed during the development process. This is done to uncover errors and fix them as soon as possible. Thomas has experience working with JUnit 4.0 and has the structured mind needed to ensure a good final product.

Report manager: The report manager will have an overview of the content of the report, make sure the content is updated and consistent. Astrid was chosen as the report manager because of her good English skills and ability to delegate.

Product owner: The product owner has the responsibilities to make decisions on behalf of the customer when he is not reachable. In normal projects this person also has the responsibility to ensure that the team delivers value to the business. During this project the team has mainly collaborated on the decisions made without the customer, but Astrid was appointed official product owner to make sure someone could make hard decisions if necessary.

Secretary: Consists of writing notes during meetings and if necessary a whole summary. The team will share this role.

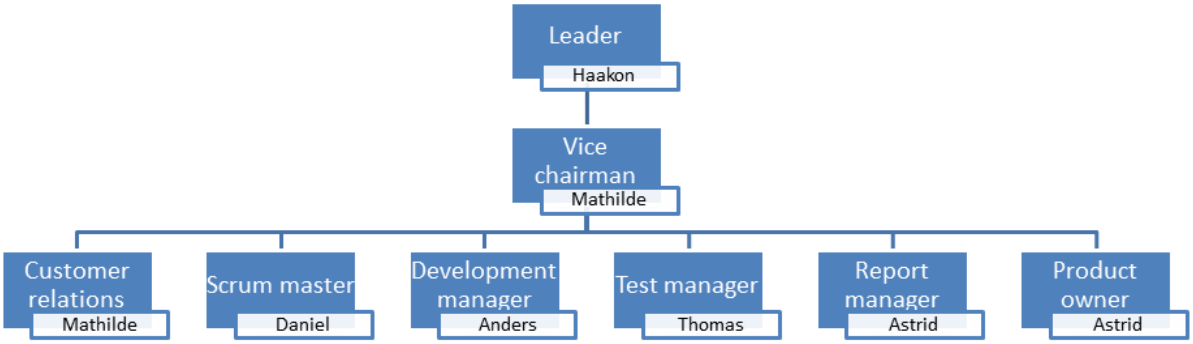


Figure 3.1: Organization chart for the team hierarchy

3.2 The team's development model

The chosen development method will be based on agile development, mainly Scrum (section 2.2.4) with small parts of extreme programming (section 2.2.3) such as pair programming. Several of the team members have practical experience with this method through school projects and summer internships with positive experiences. Several weekly meetings are required to ensure that everyone does their job, and to make sure that it is achievable to complete the project. Frequent customer contact is important to satisfy the customer and remove any uncertainty in the software's design and functionality. Small iterations with duration of two weeks will minimize risks and give the team members deadlines to complete tasks for each iteration. The figure 3.2 illustrate the team's backlog, the team's sprint backlog, and the team's iterations. The sprint usually last two weeks, and there is usually 48 hours between every stand-up. It is important that the project is finished on time, and estimation though the sprint planning is an important feature from Scrum. The report is essential in this course and it is important to work with the report throughout the whole semester to meet the courses deadlines. To solve difficult tasks a good solution is to work in pairs or ask the other team members for assistance. To keep track of all the tasks, the web based project management solution JIRA will be used.

Some of the development technology in this project are unknown for several of the team members, and much time will be used read and learn how to do tasks correctly. Due to unknown development technology, the need for breaks and a lot of meetings, the focus factor for this project will be set to 0.65. This means that 65 % of the 20 hours a week per team member is meant to be efficiently used to do specific tasks (from the agile task board).

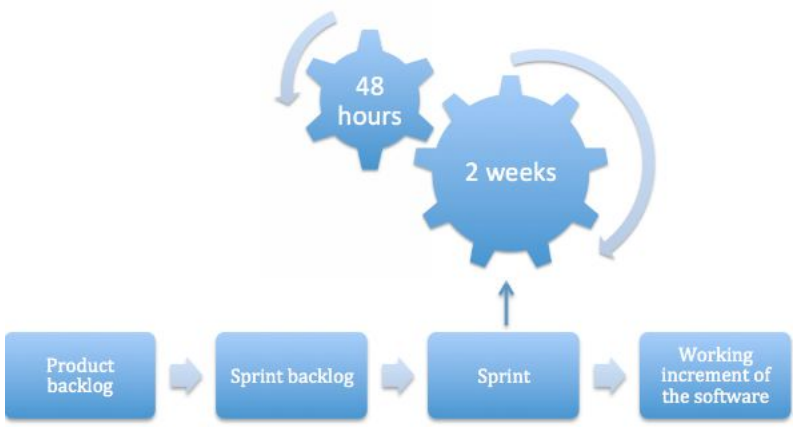


Figure 3.2: The team's development model

3.3 Work scope

In this section the work scope of the project will be discussed.

3.3.1 Risk analysis

Creating a risk analysis is a good way for the team members to be prepared for eventual problems during the project. In the first meeting as a team problems that may occur were discussed. The result is shown in the table below 3.1. Probability and consequence are numbered in the range of one to five, where five is the most serious.

Discovery of unknown tasks

Throughout the project the team is likely to find new tasks that previously were not contemplated. Therefore the probability of this problem is set to 5. To prevent this from having an effect on the finished product the team should plan the work in a way that would give some extra time at the end. The team could use this extra time to complete the unpredicted tasks. The consequence is set to 3, and this gives a sum of 15, which is the highest sum in this risk analysis.

Wrong time estimate

It is not easy to estimate the correct use of time when problems not encountered before were to be solved. Therefore probability of this risk is set to 5. With good planning and good discussions during planning, hopefully the estimates will not turn out completely wrong. To avoid a catastrophic outcome the team need to take precautions by being flexible and agile minded. The consequence is set to 3 which gives a sum of 10.

Description	Probability	Consequence	Sum (P*C)	Prevention	Measures
Discovery of unknown tasks	5	3	15	Good planning, margin for errors while estimating.	Structure new tasks and put them into sprint.
Wrong time estimate	5	2	10	Good planning, discuss every task with the team.	Work hard when a problem occurs. Allocate tasks.
Loss of data	2	5	10	Backup data.	Rewrite lost code.
Customer/ supervisor not reachable	3	3	9	Make appointments in advance.	Continue with existing tasks.
Short term sickness	4	2	8		Keep a good dialog. Share knowledge.
Internal strife	3	2	6	Talk about problems at an early stage.	Internal meeting.
Motivational problems	2	3	6	Team building, and let team members work with what they like as much as possible.	Work with smaller and simpler tasks for a while. Increased sense of achievement leads to increased motivation.
Miss the deadline	1	5	5	Good planning.	Communicate with the customer and supervisor.
Prolonged sickness	1	4	4		Keep a good dialog. Share knowledge.

Table 3.1: Risk analysis

Loss of data

Losing data will have huge consequences but is easy to prevent. The probability is set to 2 and the consequence is set to 5. This gives a sum of 10. Using tools like GIT or SVN would almost completely abolish the possibility of losing data.

Customer/supervisor not reachable

Customer or supervisor not being reachable could delay the development if the team is dependent on asking the customer/supervisor questions when encounters a crossroad. Both the probability and the consequence are set to 3, which gives a sum of 9. Figuring out problems at an early stage and not postponing hard tasks would help the team to prevent this to happen.

Short term sickness

The consequence of sickness over a time period of just some days are not severe, but can influence the work flow if it occurs regularly. But through having a good dialog and by sharing knowledge the team hopes to reduce the consequence of this risk, causing the probability to be set to 4 and the consequence to 2. This gives a sum of 8.

Internal strife

Some disagreements always occur, but it is not a problem if it is handled the right way. By internal strife the team means problems within that may have an effect on the team member's ability to work together. The probability is set to 3 and the consequence is set to 2. This gives a sum of 6.

Motivational problems

Having a low motivation is going to affect the team's ability to deliver a good product to the customer. It is important that everyone in the team feels included and is working on something interesting. Getting to know each other would make the overall motivation of the team increase. If a team member is stuck on one problem he/she could take a break, and switch tasks with other team members. The probability of this risk is set to 2, and the consequence is set to 3. This gives a sum of 6.

Miss the deadline

There are three ways that could lead to missing the deadline: Bad planning on the team's part, the customer changing the requirements at the last minute, or serveral of the risks described occurs. Good planning and continuous communication with the customer throughout the project are necessary countermeasures. The probability of this risk is set to 1, and the consequence is set to 5. This gives a sum of 5.

Prolonged sickness

It is impossible to prevent sickness from happening, but the team can keep the overall damage low by sharing knowledge and not having only one person working on each part of the system. The probability of this risk is set to 1 and the consequence is set to 4. This gives a sum of 4.

3.3.2 Work breakdown structure

The work structure is represented by a work breakdown structure. This gives a better overview of the total work of the project, and will be helpful in organizing and structuring the rest of the project. Figure 3.3 show an overall work breakdown structure, based on the Gantt diagram 3.5. The work packages in this WBS is explained in more detail in figure 3.4. Here the work packages are divided according to how the team envisioned the development of the project. This is based on how the planning, development and implementation, testing and how the report should be handled.

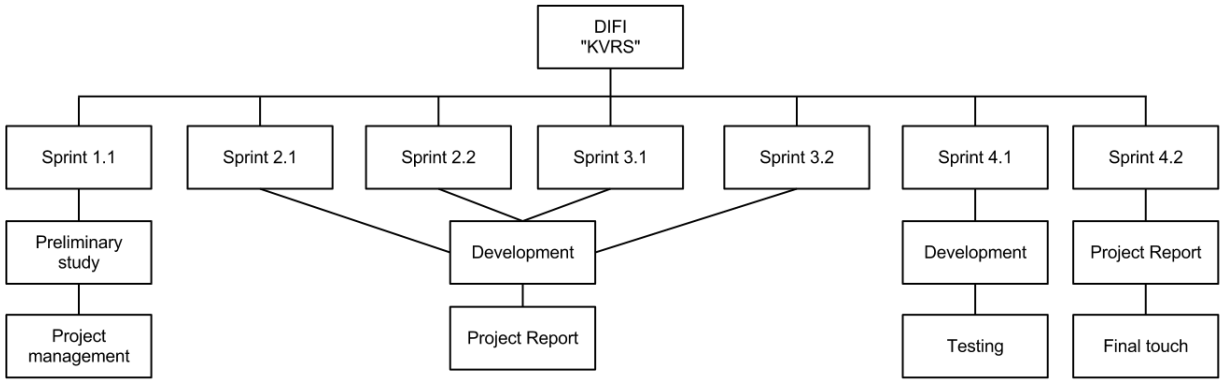


Figure 3.3: Overall work breakdown structure

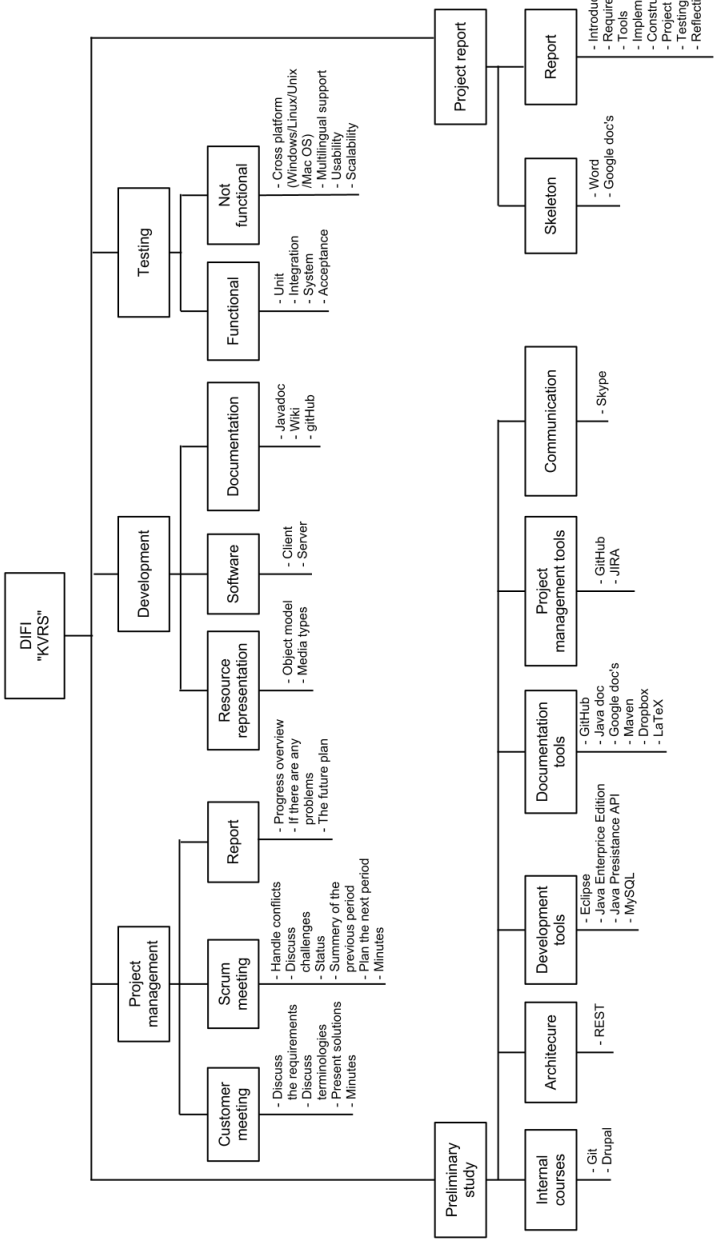


Figure 3.4: Detailed work breakdown structure

3.4 Time scope

In this section the time scope of the project will be discussed.

3.4.1 Project milestones

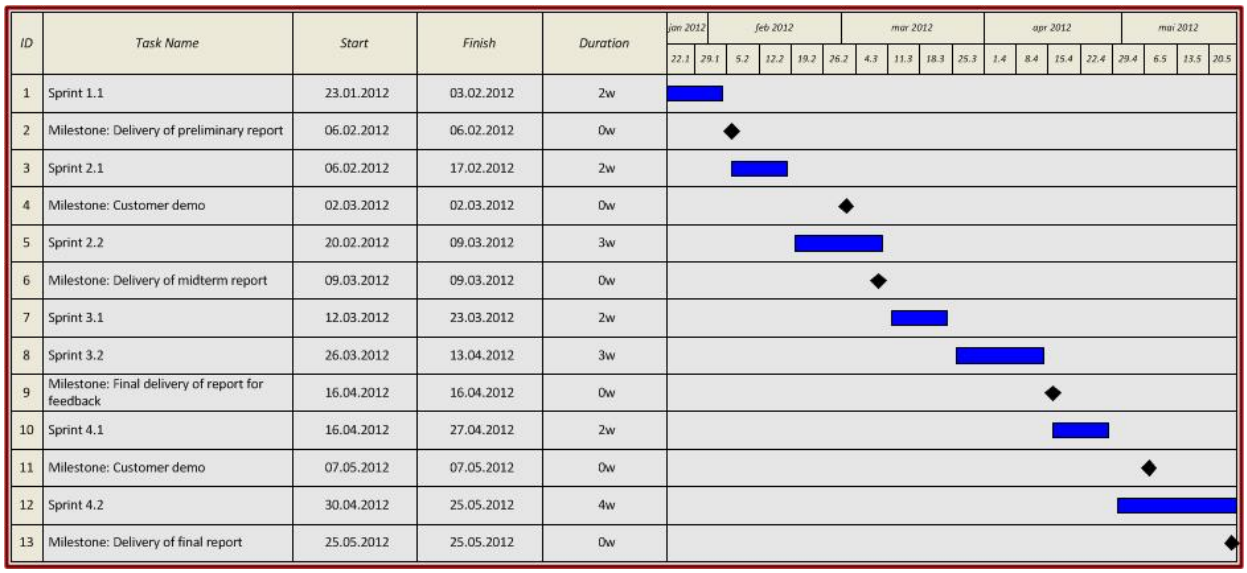
It is important to have several milestones to work against during the project. This is a general overview of the milestones in this project.

Date	**Description**
6. February	Delivery of preliminary report
2. March	Customer demo
9. March	Delivery of midterm report
16. April	Final delivery of report for feedback
7. May	Customer demo
25. May	Delivery of final report

Table 3.2: Milestones

3.4.2 Gantt chart

The Gantt chart represents the project plan (see figure 3.5). This gives a good overview over the project planning, milestones and the projects duration. The Gantt chart is divided into sprints, with start and stop dates, and important milestones such as report deliveries and customer demos.



ID	Task Name	Start	Finish	Duration
1	Sprint 1.1	23.01.2012	03.02.2012	2w
2	Milestone: Delivery of preliminary report	06.02.2012	06.02.2012	0w
3	Sprint 2.1	06.02.2012	17.02.2012	2w
4	Milestone: Customer demo	02.03.2012	02.03.2012	0w
5	Sprint 2.2	20.02.2012	09.03.2012	3w
6	Milestone: Delivery of midterm report	09.03.2012	09.03.2012	0w
7	Sprint 3.1	12.03.2012	23.03.2012	2w
8	Sprint 3.2	26.03.2012	13.04.2012	3w
9	Milestone: Final delivery of report for feedback	16.04.2012	16.04.2012	0w
10	Sprint 4.1	16.04.2012	27.04.2012	2w
11	Milestone: Customer demo	07.05.2012	07.05.2012	0w
12	Sprint 4.2	30.04.2012	25.05.2012	4w
13	Milestone: Delivery of final report	25.05.2012	25.05.2012	0w

Figure 3.5: Gantt chart

3.5 Architectural design

3.5.1 Drupal

There are two ways of implementing the wanted functionality in Drupal. The next two chapters explain the different methods roughly.

Content pages

One is to create a content page module for each of the entities (see figure 3.12) and use the 'hooks' provided by Drupal. Hooks are automatically called methods (by following a name convention) by the Drupal application when specific actions take place (for example push the save button). One of the advantages are that it is easy to install a module specified based on needs, and combine a module with others (both self made and made by others). A content page can be compared to an object that may contain different values and properties. However, in the given assignment a disadvantage appear. Drupal automatically stores information in a local (local to the server running Drupal) relation database when any changes are made. As the assignment describes, the team was to create an own server which would store the information. Even though duplication is messy, it is not catastrophic. In addition to the duplication it was harder to find good documentation for how to avoid getting the duplicated (and not guaranteed identical) information from the local Drupal database instead of the created REST server. Creating the installation file (belonging to each module) could

as well be a real time-consuming task. Figure 3.6 shows an example of using the hook structure.

```
$node->title = $result->name;
drupal_set_title($result->name);
$node->e_group_id['und']['0']['value'] = intval($result->id);
$node->e_group_active['und']['0']['value'] = $result->active;
$node->e_group_created['und']['0']['value'] = createdInMillisToText($result->created);
```

Figure 3.6: Example of setting values using the hook structure

Drupal as a view

Another way to use the Drupal framework is by creating one module (divided in different files though), and use Drupal as a pure view for representing and saving data to the external (REST) server. This avoid the problem of duplication and two databases. This solution consists in a larger scale of PHP and lesser use of the Drupal framework. Even though the use of Drupal's framework is smaller there are still forms and fields custom to Drupal with own hooks to use. The disadvantage is that all functionality is in one module, and extracting only part of the functionality may be hard. Figure 3.7 shows an example of using PHP.

```
$form['name'] = array(
    '#title' => 'Tittel',
    '#type' => 'textfield',
    '#default_value' => isset($evaluationgroup['name']) ? $evaluationgroup['name'] : '',
);
```

Figure 3.7: Example of creating a form not using the hook structure

3.5.2 Architectural diagrams

This section will discuss how the team envisioned the implementation to be in the beginning of the project.

Class diagram

The class diagram is roughly divided into three parts (see figure 3.8). At the bottom of the system is the model. This model consists of six entities which is generated in the MySQL database by JPA. The entities describe how data should be stored in the database, which attribute that is a primary key, and relations to other entities. In true JPA spirit the named queries are defined in the entities. Above the model is the logic part. Here are business logic that handle queries and synchronizes data between the

database and the REST server. Since the logic classes have some common methods and attributes, an interface called AbstractJPAHandler, was created. Resource is the final part in the back-end of the system. Resource contains the six resource classes, which mainly manages the communication between Drupal and the database. To do this, every resource class needs to handle encoding and decoding of JSON strings to its respective Java object. The front end of the system is not yet in the class diagram.

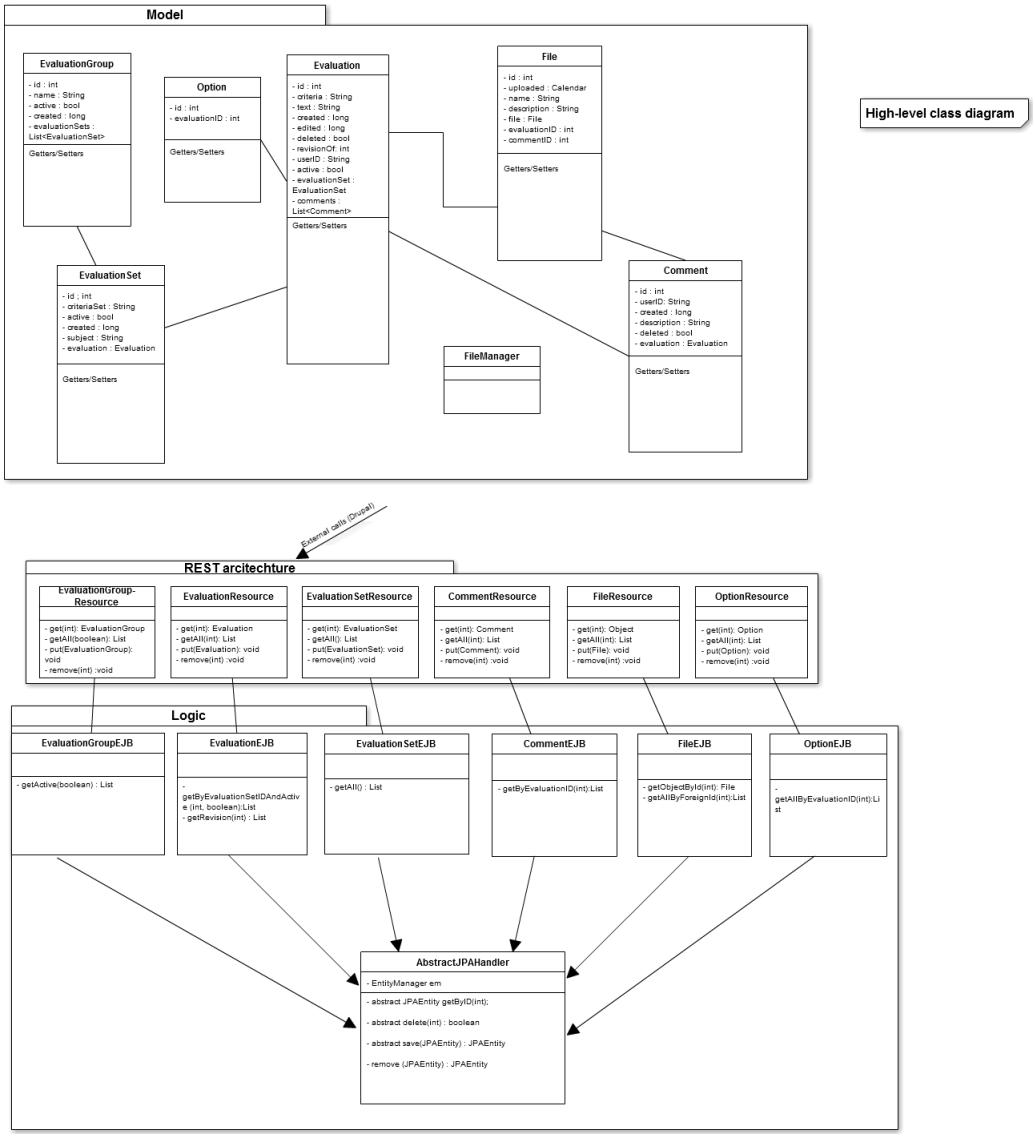


Figure 3.8: Class diagram

Entity relation diagram

Entity relation diagram shows the tables in the database and how they are connected (see figure 3.10). This ER diagram is based on the domain model received by the customer (see figure 3.9). Every table has their own identifier to easily keep track of all the different rows in the tables. The table "evaluation group" has none or many evaluation groups (Appendix A) This means that in the database, the evaluation set (Appendix A) table has an evaluationgroupid field for the relation. Evaluation set has none or many evaluations (Appendix A). This is because an evaluation is done on basis of a criteria set from evaluation set. Evaluation can have one or many options. These options are created on the basis of the criteria set from evaluation set. The evaluation table has none or many comments, so that it is possible to write a comment for every evaluation. Both evaluation and comment can have none or many files. This makes it possible for anyone who either has the ability to comment or create an evaluation, to upload a relevant file.

Because of the functional requirements the application needs to save each evaluation ever made. A new evaluation is a revision (Appendix A)of itself, else when a user update an evaluation, the application creates a new evaluation and sets the 'RevisionOf' field to the same value as the 'RevisionOf' field of the old evaluation. This way the application can group the original evaluation and all its new versions together based on the 'RevisionOf' value. The auto incremented id and the created attribute of each evaluation with the same 'RevisionOf' value, will tell in what order the evaluations were made.

Flow chart

The flow chart shows how all the different parts in the application are connected and how the data flows between them (see figure 3.11). In the bottom of the application is the MySQL database to store information. Using SQL queries from the Java Persistence API solves communication with the database. The Java Persistence API converts database information into Java objects. This makes it easy to manipulate or pass the information to the REST component. The REST component retrieve or send information with simple http URLs. The REST component converts Java objects to and from simple JSON strings. The actual website which users interact with, uses these JSON strings from the REST component to show and save data. The website is based on the open-source content management system and content management framework Drupal.

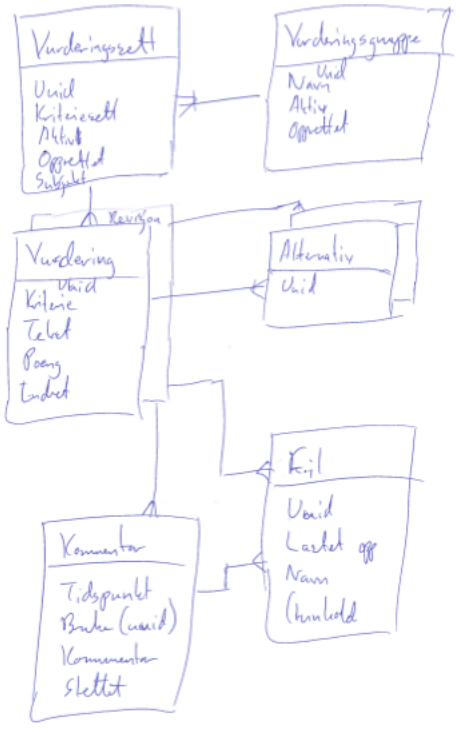


Figure 3.9: Domain model from the customer

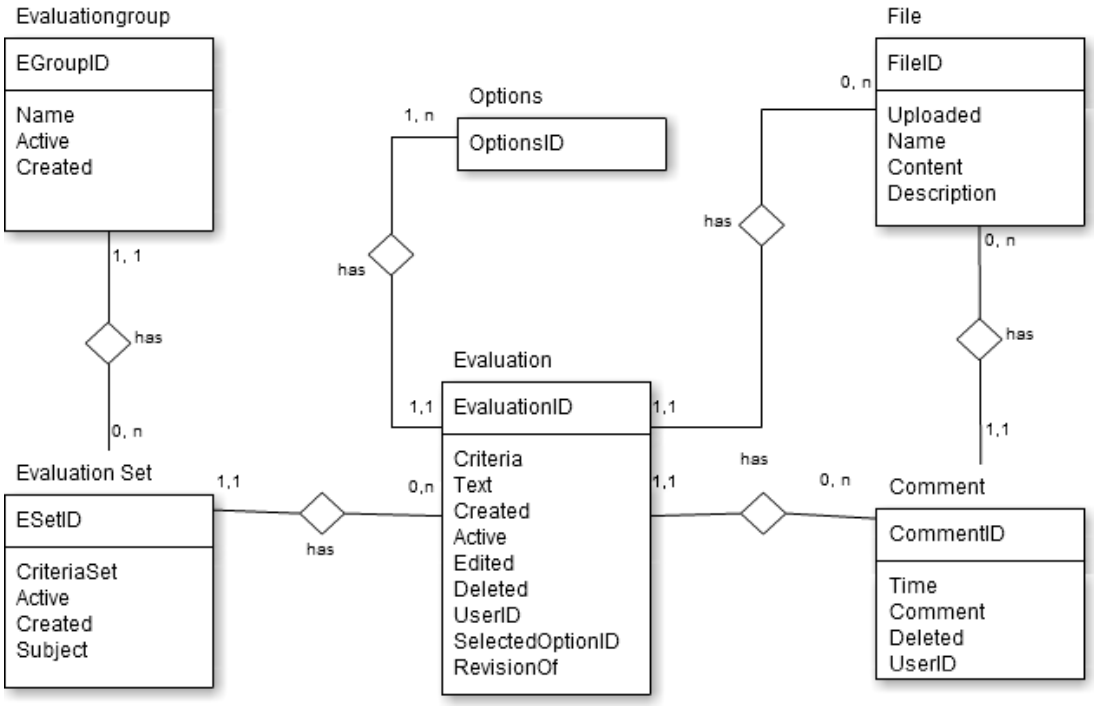


Figure 3.10: Entity relation diagram

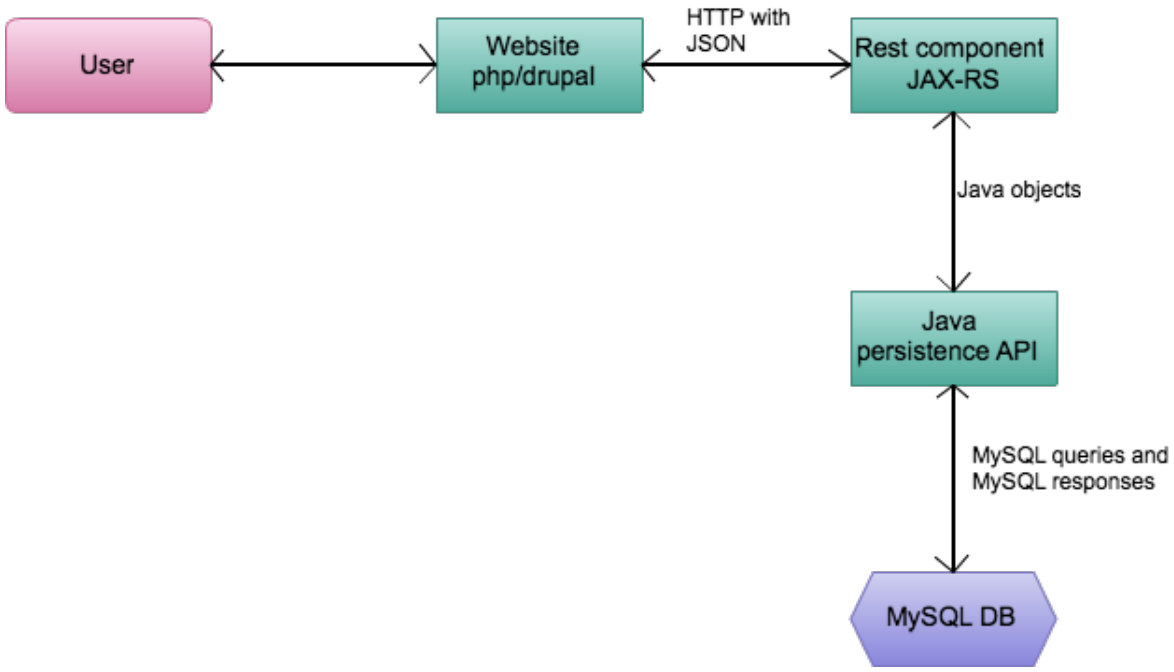


Figure 3.11: Flow chart

Sequence diagram

Sequence diagrams easily shows how the user interacts with the application on a logic level. The diagrams shown below describe how the team envisioned the different sequences.

Figure 3.12 describes the sequence when a user add a new file to an evaluation. First the user needs to select the evaluation. Then the application will display this evaluation and the user can click on the newFile button. The user then sees the file uploader view and selects the file to upload. The system stores the file in the database and goes back to the evaluation view, with the recently uploaded file below.

Figure 3.13 describes the sequence when the user displays all evaluations groups. The user then clicks on the button "Display all evaluation groups". Drupal then calls the REST server's getAll method, which returns all the evaluation groups as a JSON string. The evaluation group view in Drupal will then show all the evaluation groups in a nice GUI.

Figure 3.14 describes the sequence when a user creates a new evaluation group. The user then clicks on the newGroup-button, and the form for creating a new evaluation group appears. Then the user fills out the form and saves the group. Then the system will show the evaluation group just created.

Figure 3.15 describes the sequence when a user creates a new revision of an evaluation. The user clicks on

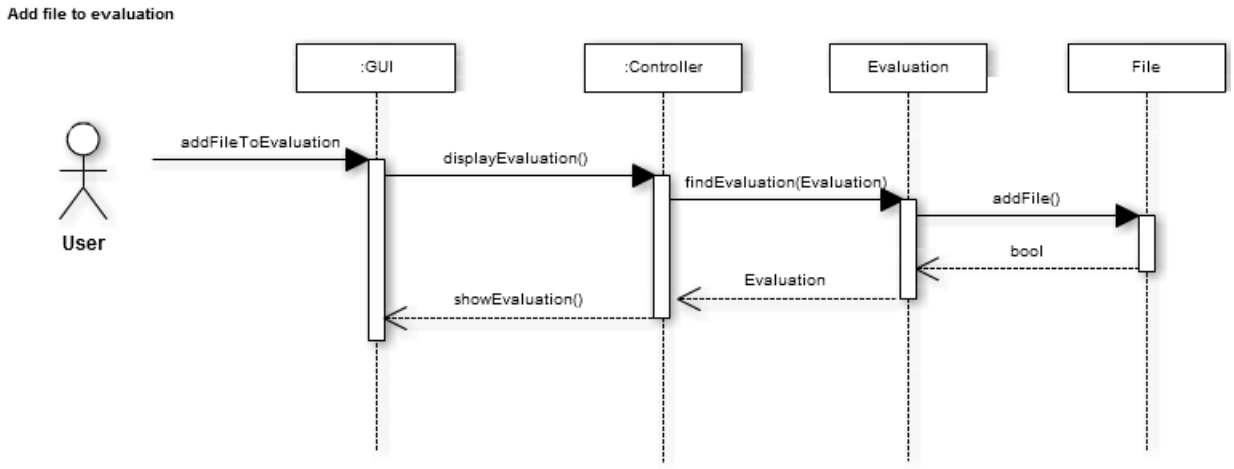


Figure 3.12: Sequence diagram, Add file to evaluation

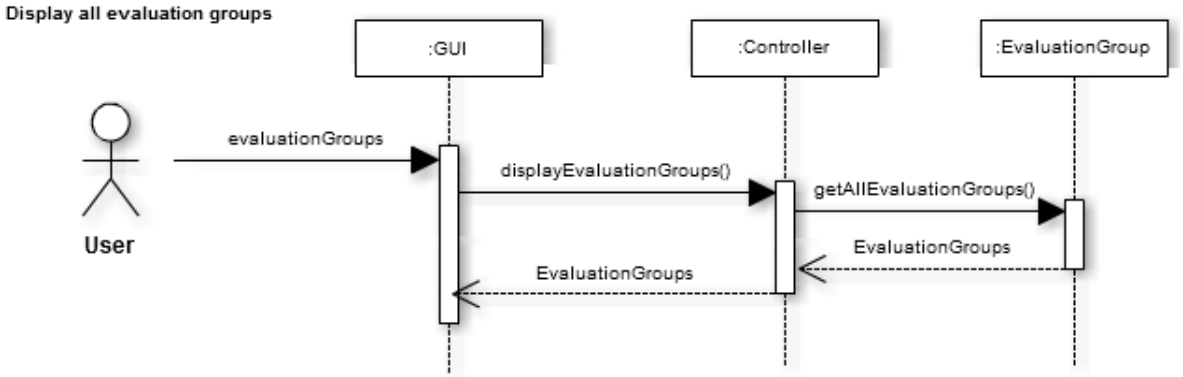


Figure 3.13: Sequence diagram, Display all evaluation groups

the newEvaluation-button, and the form to create a new evaluation appears. This trigger the setCurrentEvaluationInactive() method, which set the marked evaluation as inactive. The new information about the evaluation is saved, and the system shows the evaluation.

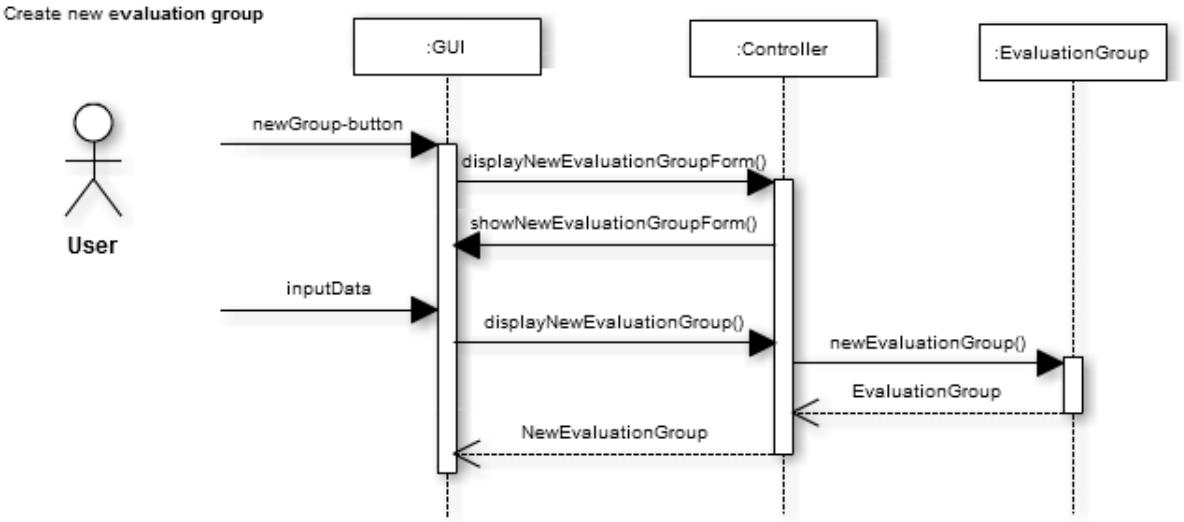


Figure 3.14: Sequence diagram, Create new evaluation group

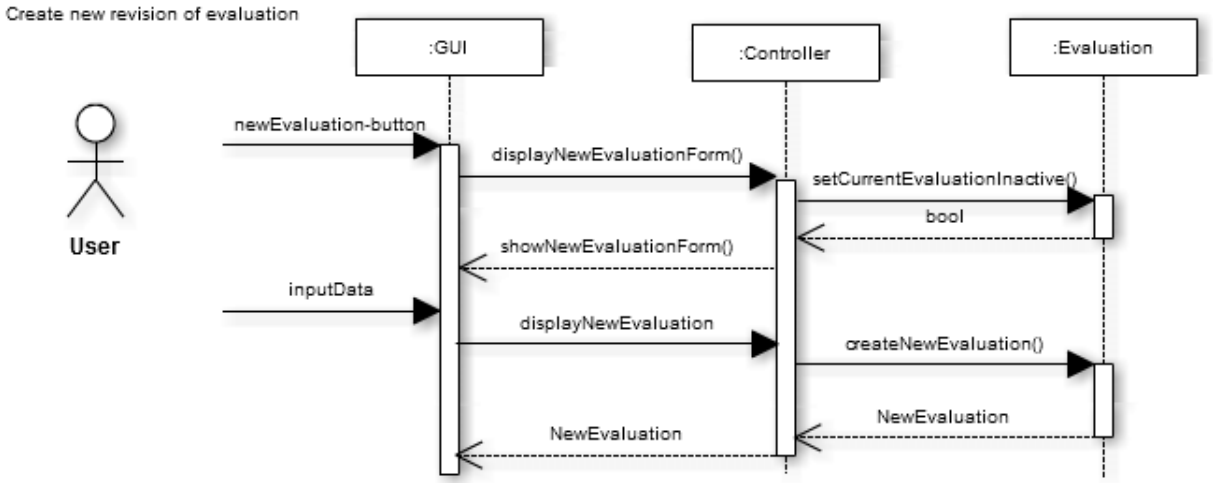


Figure 3.15: Sequence diagram, Create new revision of evaluation

3.6 Prototype

These images are the teams interpretation of the customers requirements, and the expectations to the final look of the system.The team realizes that there most likely will be changes to this prototype, as more knowledge is gain about the assignment. These images will however be a good guideline for future work.

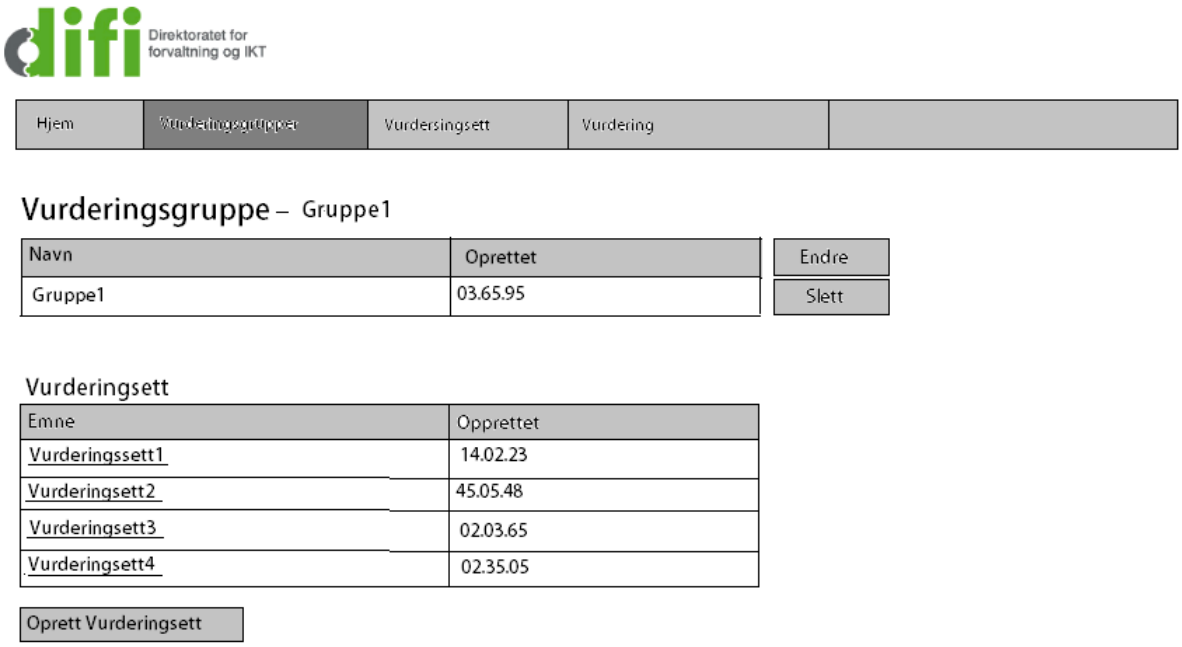


Figure 3.16: Evaluation group

This screen shows a selected evaluation group and all its containing evaluation sets. There is also a button for adding new sets and for editing and deleting the selected group.

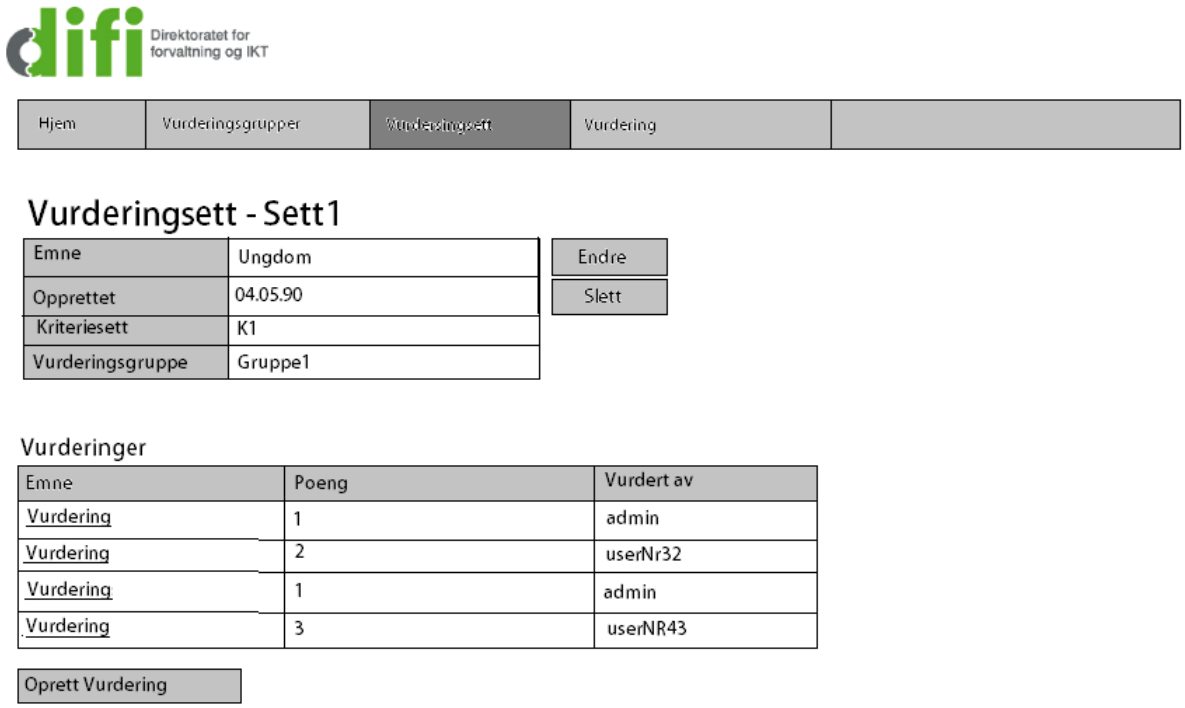


Figure 3.17: Evaluation set

This screen shows a selected evaluation set and all its containing evaluations. There is also a button for adding new evaluations and for editing and deleting the selected set.

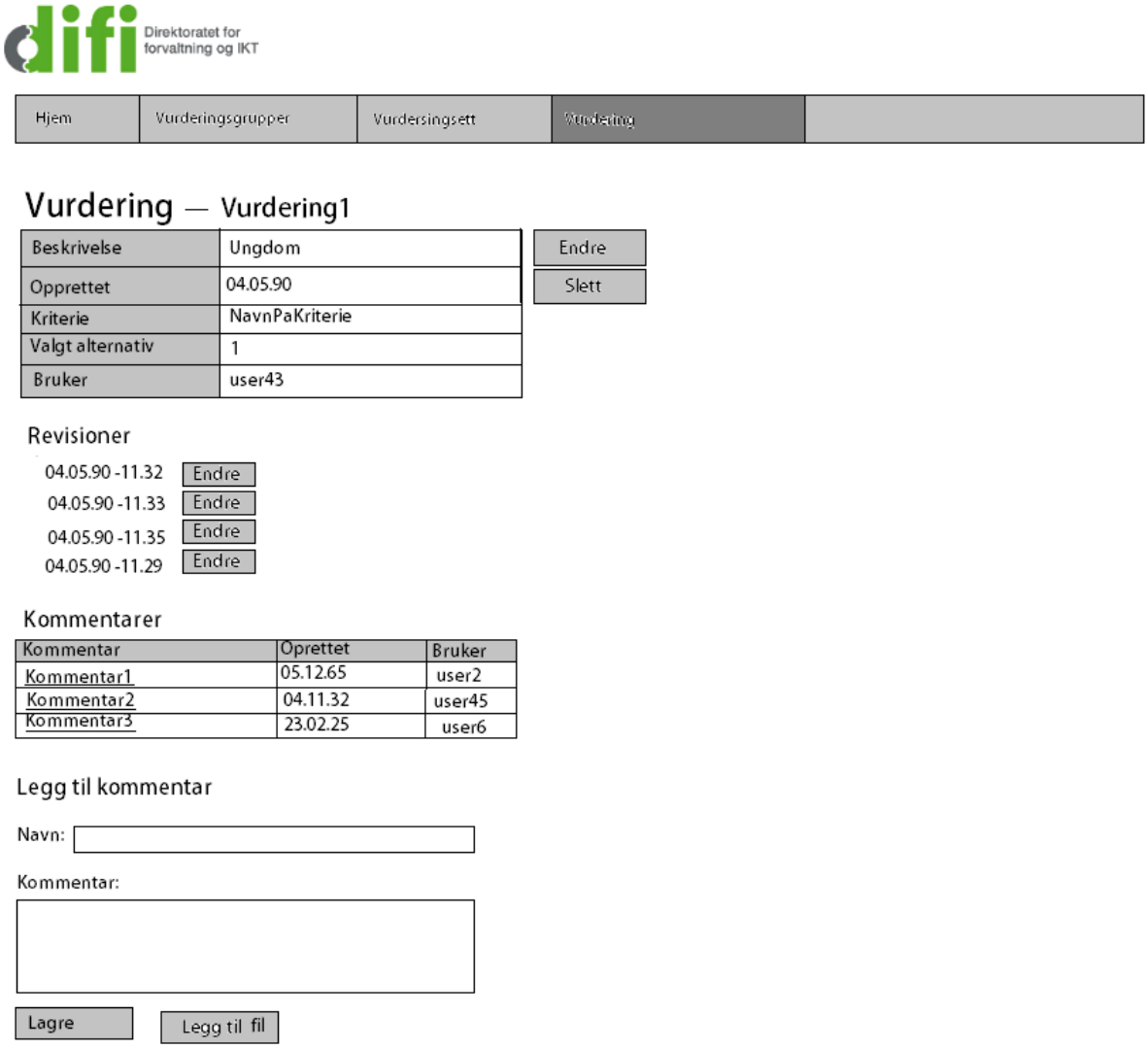


Figure 3.18: Evaluation

This screen shows a selected evaluation and all its containing comments. There is also a button for editing and deleting the selected evaluation, and a comment form that can be used to add new comments to this evaluation. All revisions of this evaluation is shown in a list.

Figure 3.19: Comment,

This screen shows a selected comment, with edit and delete buttons, and a list of all files on this comment.

Chapter 4

Requirements

This chapter will discuss the functional and non-functional requirements from the customer. This includes the prioritization of the requirements and textual use cases that reflect the requirements.

The goal for the project is to replace the current management of evaluations in KVRS and add selected functionality for a broader utilization in the long run. The functional requirements are from the customer.

4.1 Functional sub-goals

- **FR-1** - Establish a REST service for evaluation set.
- **FR-2** - Create the user interface in Drupal towards the REST service, which will replace functionality in KVRS.

4.2 Functional requirements

- **FR-1 - REST service**

  FR-1.1 - List all active/inactive evaluation groups

  The user needs to have the possibility to list all active and inactive evaluation groups. The list will display the attributes 'active', 'created', 'id' and 'name' for every evaluation group. The user can filter on active and inactive groups. Every group can contain zero or several evaluation set.

  FR-1.2 – Create evaluation group (logical unit)

The user can create a new, empty evaluation group. The attribute 'name' must be specified when created. 'Creation date', 'id' and 'active' are set automatically by the system.

FR-1.3 – Update evaluation group

The user can edit an existing evaluation group. 'Name' is the only attribute available for change. The values of 'creation date', 'id' and 'active' are unaltered.

FR-1.4 – Set evaluation group inactive

The user can mark a group as inactive. The 'active' attribute will change from 1 to 0 and the group will not be shown under FR-1.1. The user cannot undo this operation.

FR-1.5 – List all existing evaluation set

The user has the possibility to list all evaluation set. The list will display the attributes 'active', 'creation date', 'id', 'subject' and 'criteriaSet' for every evaluation set. The user can filter on active and inactive. Every set can contain zero or several evaluations.

FR-1.6 – Create evaluation set (logical unit)

The user can create a new, empty evaluation set. Every set has the attributes 'active', 'created', 'criteriaSet', 'id', 'subject' and a foreign key to the evaluation group it belongs to. The user needs to specify the values of 'criteriaSet', and subject. The rest of the attributes will be set by the system. Every evaluation set can contain zero or several evaluations.

FR-1.7 – Update evaluation set

The user can edit an existing evaluation set. 'Subject' is the only attribute available for change. The values of 'creation date', 'id', 'active', 'criteriaSet', and foreign key to evaluation set are unaltered.

FR-1.8 – Set evaluation set inactive

The user can mark a set as inactive. The 'active' attribute will change from 1 to 0 and the set will not be shown under FR-1.5. The user cannot undo this operation.

FR-1.9 – List all active/inactive evaluations in an evaluation set

The user has the possibility to list all active and inactive evaluations in an evaluation set. The list will display all the attributes attached to the set and the user can filter on active and inactive evaluations.

FR-1.10 – List all revisions of evaluation

The user has the possibility to list all revisions of an evaluation. The list will display all the attributes attached to an evaluation, a foreign key to the evaluation it is a revision of and a foreign key to the evaluation set it belongs to. Every evaluation can contain zero or several revisions.

FR-1.11 – Get revision of evaluation

The user can select a revision of an evaluation and get all the associated values of the revision, including foreign keys to evaluation set and evaluation. It does not matter if the revision is active or inactive.

FR-1.12 – Create evaluation in evaluation set

The user can create a new, empty evaluation. Every evaluation has the attributes 'active', 'created', 'criteria', 'id', 'text', 'revisionOf', 'userId', 'selectedOptionId' and a foreign key to the evaluation set it belongs to. The user needs to specify the values of 'criteria', 'selectedOptionId' and text. The rest of the attributes will be set by the system. Every evaluation can contain zero or several comments or files attached to it.

FR-1.13 – Create a new revision of evaluation

The user can edit an existing evaluation. The edited evaluation should then be newest revision of the evaluation, and therefore the only one possible for editing.

FR-1.14 – Mark evaluation deleted

The user needs to have the possibility to delete an evaluation. The 'active' attribute will change from '1' to '0'. The operation is not possible to undo. Only the newest revision of an evaluation is possible to mark as deleted, and if there are several revisions of an evaluation then all the revisions of the evaluation should be marked as deleted. The evaluation will not be listed in FR-1.10.

FR-1.15 – Get point distribution for a given criteria and evaluation set

This requirement has been canceled, since it requires data from the other Difi group, group 6 (A).

FR-1.16 – List all comments on evaluation

The user has the possibility to list all active and inactive comments of an evaluation. The list will display all the attributes attached to a comment, and foreign key to the evaluation it belongs to. Every evaluation can contain zero or several comments.

FR-1.17 – List all comments on revision of evaluation

The user has the possibility to list all comments of a revision of an evaluation. The list will display all the attributes attached to the revision, and foreign key to the revision it belongs to. Every revision can contain zero or several comments.

FR-1.18 – Add comment on revision of evaluation

The user can comment on a revision of an evaluation. The user needs to specify the attribute 'description', the attributes 'created', 'active', 'id', 'userId' and a foreign key to the evaluation is set automatically by the system.

FR-1.19 – Mark comment deleted

The user needs to have the possibility to delete a comment. The attribute 'active' will then change from '1' to '0'. The operation is not possible to undo. The comment will not be listed in FR-1.16.

FR-1.20 – Add file(s) on comment

The user has the possibility to upload a file on a comment.

FR-1.21 – Add file(s) on revision of evaluation

The user has the possibility to upload a file on a revision of an evaluation.

FR-1.22 – List all files on comment

The user has the possibility to list all files on a comment of an evaluation. The list will display all the attributes attached to the file, the file itself and a foreign key to the comment it belongs to. Every comment has zero or several files attached to it.

FR-1.23 – List all files on evaluation

The user has the possibility to list all files on an evaluation. The list will display all the attributes attached to the file, the file itself and a foreign key to the evaluation it belongs to. Every evaluation has zero or several files attached to it.

FR-1.24 – List all files on revision of evaluation

The user has the possibility to list all files on a comment of an evaluation. The list will display all the attributes attached to the file, the file itself and a foreign key to the revision it belongs to. Every revision has zero or several files attached to it.

- **FR-2 - Drupal integration;** below is the initial Drupal requirement from the customer. The plan is to make one commentator- and one admin-user. The customer later suggested that the team should concentrate on making the admin-user's functionality first, and if there were time at the end of the project, add the other user as well.

  FR-2.1 – Create a privilege and functionality that makes it possible to do all actions against the REST service.

  The system needs a user that has the possibilities to manage everything. No restrictions at all, available to control every part of the system. This is a root/admin user.

4.3 Non-functional requirements

- **NFR-1.1 – Usability:** Although the customer asked the team not to prioritize usability, it is still important that the system is usable by non-technical users. The team has chosen not to do advanced user testing but still considered whether the product will be understandable to the average user.
- **NFR-1.2 – Compatibility:** The system must also be usable on different platforms and browsers. Since the system is written in Java, it is portable to several platforms. Drupal also helps to make sure that the system will work in different browsers.
- **NFR-1.3 – Implementation:** This system must be installable on a UNIX-server.

- **NFR-1.4 – Security:** The data that is processed by the system is not too sensitive, so this is not something that needs high priority. However there can never be too much security.

4.4 Prioritization of the functional- and non-functional requirements

In relation with the product backlog, a prioritization of the requirement is necessary. These tables are developed in collaboration with the customer, and give an overview of the dependencies and the prioritization of the different requirements.

Table 4.1 gives an overview of the prioritization the functional requirements, sorted after the id of the requirement.

Requirements	**Priority / Importance**	**Dependencies**
FR-1.1	High	None
FR-1.2	High	None
FR-1.3	High	FR-1.2
FR-1.4	Medium	FR-1.2
FR-1.5	High	None
FR-1.6	High	None
FR-1.7	High	FR-1.6
FR-1.8	Medium	FR-1.6
FR-1.9	High	None
FR-1.10	Medium	None
FR-1.11	Medium	FR-1.12
FR-1.12	High	FR-1.6
FR-1.13	Medium	None
FR-1.14	Medium	FR-1.12
FR-1.15	Low	FR-1.6
FR-1.16	Low	None
FR-1.17	Low	None
FR-1.18	Medium	FR-1.12
FR-1.19	Low	FR-1.18
FR-1.20	Low	FR-1.18
FR-1.21	Medium	FR-1.13
FR-1.22	Low	FR-1.20
FR-1.23	Low	FR-1.21
FR-1.24	Low	FR-1.21

Table 4.1: Prioritization of functional requirements

Table 4.2 gives an overview of the prioritization the Drupal integration requirements, and the third table 4.3 gives an overview of the prioritization the non-functional requirements. The tables are sorted after the id of the requirement.

Requirements	**Priority / Importance**	**Dependencies**
FR-2.1	High	None

Table 4.2: Prioritization of drupal integration

Requirements	**Priority / Importance**	**Dependencies**
NFR-1.1	Low	None
NFR-1.2	Low	None
NFR-1.3	Low	None
NFR-1.4	Low	None

Table 4.3: Prioritization of non-functional requirements

4.5 Use cases

This section contains the textual use cases of the system. The reason why only textual use cases are included is because they are more detailed, and more relevant for the report than use case diagrams. The use case diagrams only include one user, which has all the rights, and will be connected to all the requirements. The textual use cases will look like table 4.4:

Use case ID	UseCaseID
Use case name	The name of the use case
Scope	The scope of the use case
Description	The goal of the use case, and sources of requirement
Preconditions	The preconditions that must be met before the use case can begin
Assumptions	The conditions that must be met for the use case to terminate successfully
Steps	The steps between the actor and the system that are necessary to achieve the goal
Variations	The different variations of the flow of the use case

Table 4.4: Use case, example

The use cases are designed in connection to how the team envisioned to solve the requirements. They are described through the front-end design (Drupal). The reason why only use cases from the front-end are described is because if the front-end works, it ensures that the REST server is functioning.

The use cases below are connected to the functional requirements through their numbers. For example use case with ID UC-1.1 (Appendix E) correspond to FR-1.1. In some cases a letter will follow the usual naming convention of the ID. For example UC-1.21A 4.5 and UC-1.21B 4.6 . This means that there are different ways of doing the corresponding requirement. Below are two examples of the textual use cases. The rest of the textual use cases is located in Appendix E, Extended textual use cases.

The actor in all the use cases is the admin, which has no limitations.

Use case ID	**UC-1.21A**
Scope	Evaluation and file
Description	Add file(s) on revision of evaluation
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select a revision of an evaluation 2. Click edit 3. Select add file 4. Select a file from your device (A new link add file will occur below the previous one) 5. Jump to step 2 as many times as wanted 6. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table 4.5: Use case, UC-1.21A

Use case ID	**UC-1.21B**
Scope	Evaluation and file
Description	Add file(s) on revision of evaluation
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Click on new evaluation 2. Fill in the required fields 3. Click add file 4. Select a file from your device (A new link add file will occur below the previous one) 5. Jump to step 2 as many times as wanted 6. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table 4.6: Use case, UC-1.21B

Chapter 5

Testing

This chapter will discuss the testing of the product. Several flaws and bugs can be discovered through thorough testing. Different modules and classes will be continuously tested when they are integrated in the system. The components will be tested through unit-, integration-, system and usability testing.

5.1 Test plan

Below is a description of the different types of tests the team has planned to use in the project. The team has decided to handle the tests as Scrum tickets. Before a development-ticket is set to “completed” in JIRA, it is moved from ”in progress” to ”ready for testing”. That way, it is guaranteed that everything that needs to be tested, will be tested. It is easy for other team members to see what parts of the system that are ready for testing.

Test cases for unit testing were created for every functionality the team members implemented. Since each unit test focuses on testing a small and individual part of the system, it has resulted in a lot of tests. The team has chosen not to include all the unit tests in the report, this because many of the tests are very similar and unnecessary to include.

5.1.1 Unit testing

Unit testing will be used to test the smallest components of the system. This will be a form of white box testing, because it requires some knowledge of the actual code. These tests will be performed as the different

Test ID	TestID
Test name	The name of the test
Purpose	The purpose of the test, and the aspects of the system being tested
Requirements	The different requirements that must be fulfilled for the test to pass
Test description	This is the description of what the input and output should be for the test to pass. 1. The different points may be linked to requirements from the customer.
Test result	This is the status of the test - if the test passed of failed.

Table 5.1: General test case

parts are finished, by someone with knowledge in the area being tested. Important and complex methods will be tested thoroughly with JUnit 4.0 framework. Drupal offers unit testing by their integrated module "SimpleTest". See section 5.2.1 for the results of the unit testing.

5.1.2 Integration testing

As the unit testing completes and becomes approved for the minor components, there is also the need to test whether they can be merged and run smoothly together. Since different team members are developing the components in parallel, merging of code and integration tests should be performed in groups consisting of the creators. All tests are performed outside the source environment to test the integration between different modules. cURL will be used to test the pairing of Drupal and the REST server. See figure 5.1 for these results. Other test results from the integration testing will be discussed in the iteration they were performed.

5.1.3 System testing

System testing is used to test the overall functionality and response time of the system. To be sure that all the functional and non-functional requirements given to us by the customer are covered, each of the use cases/criterias will be put into a test. This means to reformulate them into steps, equivalent to the action that needs to be taken by the user to perform the different tasks. System testing is a form of black box

testing and should not require any knowledge of the logic or design. The team as a group will preform the system testing. See section 6.2.2 for results of the system testing.

5.1.4 Usability testing

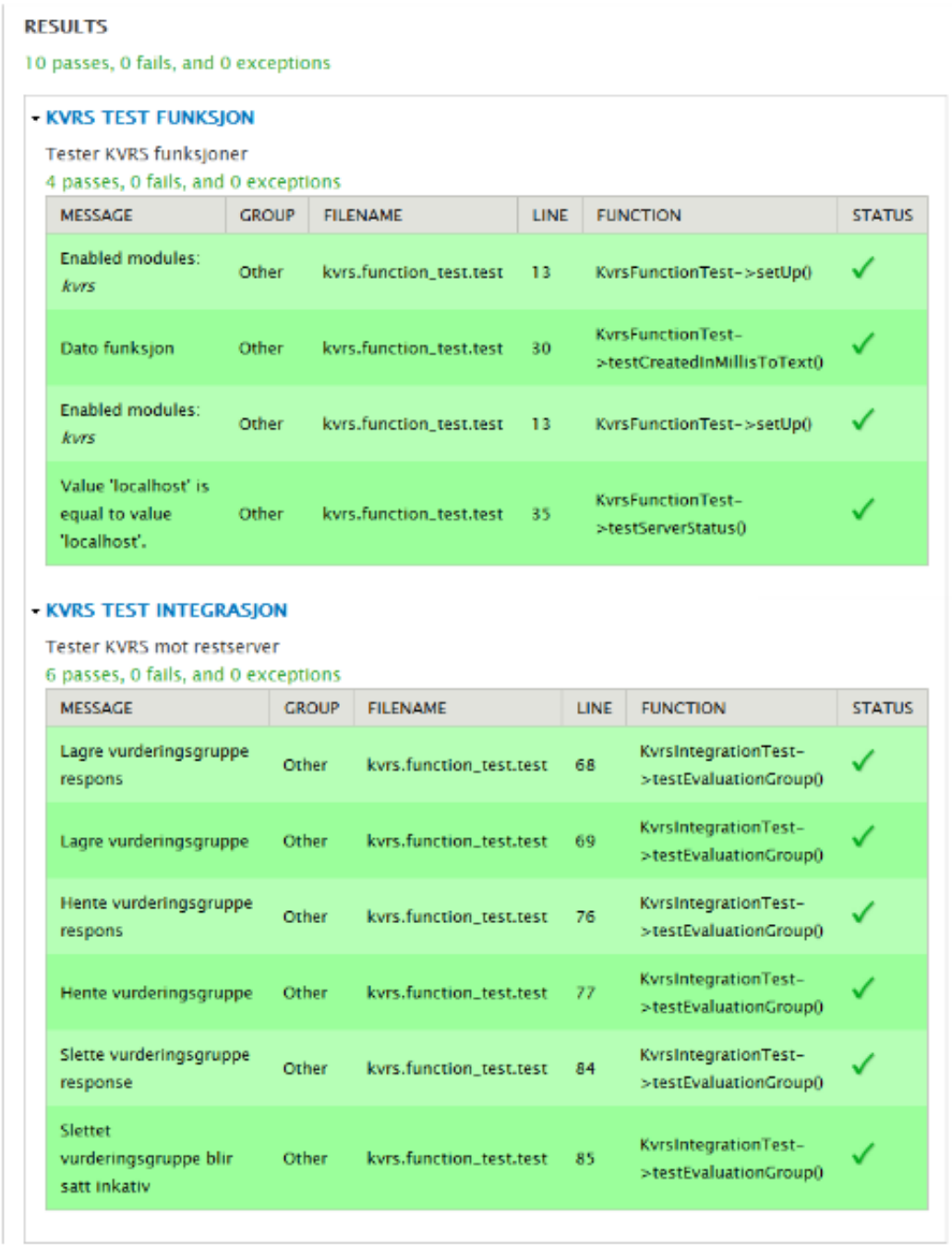
This is a system that will be used by real people in their work, and usability should be an important part of the development. However the customer has specified that the focus should be on the functionality of the system, and not the looks. The actual appearance of the program will be designed and made by the customer's employees when they receive the final product. The team could however be creative and make something that was possible within the time limit. Because of this usability is not the main focus, but it is still an aspect of the project that the team should keep in mind. The results from the usability testing will be discussed in the iteration they were performed.

5.2 Test results

This section displays the results from the different tests.

5.2.1 Unit testing results

The Drupal unit testing was executed through the integrated Drupal module "SimpleTest". The result is shown in figure 5.1. The 'kvrs test function' tests some internal assist functions, while the 'kvrs test integration' tests the actual methods for inserting, getting and delete. The team chose to write tests for evaluation group. The tests for evaluation set, evaluation, comment and file would have a similar structure.



RESULTS

10 passes, 0 fails, and 0 exceptions

KVRS TEST FUNKSJON

Tester KVRS funksjoner

4 passes, 0 fails, and 0 exceptions

MESSAGE	GROUP	FILENAME	LINE	FUNCTION	STATUS
Enabled modules: *kvrs*	Other	kvrs.function_test.test	13	KvrsFunctionTest->setUp()	✓
Dato funksjon	Other	kvrs.function_test.test	30	KvrsFunctionTest->testCreatedInMillisToText()	✓
Enabled modules: *kvrs*	Other	kvrs.function_test.test	13	KvrsFunctionTest->setUp()	✓
Value 'localhost' is equal to value 'localhost'.	Other	kvrs.function_test.test	35	KvrsFunctionTest->testServerStatus()	✓

KVRS TEST INTEGRASJON

Tester KVRS mot restserver

6 passes, 0 fails, and 0 exceptions

MESSAGE	GROUP	FILENAME	LINE	FUNCTION	STATUS
Lagre vurderingsgruppe respons	Other	kvrs.function_test.test	68	KvrsIntegrationTest->testEvaluationGroup()	✓
Lagre vurderingsgruppe	Other	kvrs.function_test.test	69	KvrsIntegrationTest->testEvaluationGroup()	✓
Hente vurderingsgruppe respons	Other	kvrs.function_test.test	76	KvrsIntegrationTest->testEvaluationGroup()	✓
Hente vurderingsgruppe	Other	kvrs.function_test.test	77	KvrsIntegrationTest->testEvaluationGroup()	✓
Slette vurderingsgruppe response	Other	kvrs.function_test.test	84	KvrsIntegrationTest->testEvaluationGroup()	✓
Slettet vurderingsgruppe blir satt inkativ	Other	kvrs.function_test.test	85	KvrsIntegrationTest->testEvaluationGroup()	✓

Figure 5.1: An image of testing in Drupal via SimpleTest

5.2.2 System testing results

The results from the system testing is displayed below through test cases.

Test 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7 show the tests done on the evaluations in an evaluation set. The rest of the test cases can be found in Appendix F.

Test ID	ST-08
Test name	Evaluation 1
Purpose	To test if the requirement FR-1.9, List all active/inactive evaluations in evaluation set, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set A list of the active evaluations appear 4. Click on "Show deleted" A list of the deleted evaluations appear
Test result	PASSED

Table 5.2: Evaluation test, list all active/inactive evaluations in evaluation set

Test ID	ST-9
Test name	Evaluation 2
Purpose	To test if the requirement FR-1.10, List all revisions of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation A list of the different revisions of the evaluation appear
Test result	PASSED

Table 5.3: Evaluation test, list all revisions of evaluation

Test ID	ST-10
Test name	Evaluation 3
Purpose	To test if the requirement FR-1.11, Get revision of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation A list of the different revisions of the evaluation appear 5. Click on the desired revision The information about the revision of the evaluation appear
Test result	PASSED

Table 5.4: Evaluation test, get revisions of evaluation

Test ID	ST-11
Test name	Evaluation 4
Purpose	To test if the requirement FR-1.12, Create evaluation in evaluation set, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on "Add new evaluation" 5. Choose criteria and option 6. Enter an explanation of the evaluation A note confirms that the evaluation is added
Test result	PASSED

Table 5.5: Evaluation test, create evaluation in evaluation set

Test ID	ST-12
Test name	Evaluation 5
Purpose	To test if the requirement FR-1.13, Create new revision of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on "Edit" 6. Choose a different option, and write an explanation 7. Click on "Save" A note confirms that the evaluation is updated appear
Test result	PASSED

Table 5.6: Evaluation test, create a new revision of evaluation

Test ID	ST-13
Test name	Evaluation 6
Purpose	To test if the requirement FR-1.14, Mark evaluation deleted, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on "Delete" A note confirms that the evaluation is deleted
Test result	PASSED

Table 5.7: Evaluation test, mark evaluation deleted

5.3 Test evaluation

Overall the testing has been very rewarding for the project. It has given the team valuable input, not only as to finding errors, but also for improving the functionality of the system.

During the development of the Java part of the project the team made a lot of JUnit tests, but because of a complicated error these tests refused to run properly. After days with no progress on this error and even after consulting the customer, the team decided to set the task on hold. Finding the solution could take any amount of time, and there was no guarantee that it would work in the end. Instead these methods where tested manually, adding, removing and updating data through the different methods and checking the database to see if the changes happened correctly.

At the end of the project the team also had some simple usability testing where the team members where given different tasks to complete, like adding, editing and deleting an evaluation group. This made it possible to do some last minute improvements on buttons and the flow between the screens.

Chapter 6

Iterations

This chapter is a summary of the different iterations that have been completed during the project. This includes the backlog of each sprint, project management, the result of the sprint, and the product backlog at the end of each sprint.

The solution can roughly be divided into 3 parts; Drupal, REST and JPA. Since the team consists of six people a decision was made to put two team members on each part. Typically after each sprint, the team members switched parts so everyone got to know all parts of the system. To preserve the progress it was always made sure that one of the original members stayed on the task, to teach and help the new person, by for example pair programming. Since all the team members have different experience and knowledge, it was discussed how this could be beneficial. It resulted in a Git course and internal teaching. Skype (Appendix A) was used to communicate within the team, and pair programming for knowledge transfer. This way, not only the competence was transferred between several members, but it was also a good introduction to a new theme or code bit.

Throughout the whole semester, except the holidays, the team has met three times a week, on Mondays, Wednesdays and Fridays. Every meeting has started with a stand-up where everyone explains what they have done since the last meeting, if they had any problems and what they are planning to work on until next meeting.

At the beginning of a new sprint the team always meets for a scrum meeting. This implies an evaluation of the last sprint, overall status, and estimating of new tasks from the backlog. If anyone had any preferences about what kind of task they wanted, this was taken into account.

The sprints are divided into separate numbers mainly to represent different deliveries. Sprint 1.1 aims towards first milestone which is the delivery of the preliminary report. Sprint 2.1 aims towards the customer demonstration and the sprint 2.2 towards the midterm report. Sprint 3.1 and sprint 3.2 aim to the delivery of the last report with feedback. Sprint 4.1 focuses on the second customer demonstration and end of sprint 4.2 is the final delivery. The team decided early that these milestones would be a good basis for the sprint length, and because of this there has been a tight connection between the sprints and milestones throughout the project.

6.1 Sprint 1.1

Sprint 1.1	**Dates**
Sprint planning	Monday, January 23rd
Timeline	January 23rd - February 3rd

Table 6.1: Dates, Sprint 1.1

Monday, January 23rd the team had their first meeting. Everyone told about themselves; what kind of experience they had, working patterns and their ambitions for this course. The task description was read and the team tried to understand main aspects. The entity relationship diagram and class diagram was roughly sketched. It was obvious that the team needed to contact the customer for a more detailed description. The team then decided to meet three times every week, at 10.15 am.

At Wednesday January 25rd, the team had their first Skype meeting with the customer contact. He explained the task very thorough and answered our other questions about the requirements, programs and technologies. The team now had a basic understanding of the structure of the project, and then started on modeling diagrams and the preliminary version of the report.

6.1.1 Sprint backlog

The sprint goal were to get an overview over the tasks and the how the complete application finally would look like. The team needed to get this right in order to start on the first version of the report. Due to new reading, setting up environments, and establishing contact with the customer and understanding the given project, the estimate is set to 81 hours of efficient work that can point to result.

The ticket names are generated from JIRA, the tool for tracking issues. The reason that tickets start at

DIFI-9 is due to earlier testing in JIRA.

Ticket:	DIFI-9	
Task:	Report, delivery 1	
Description:	Create diagrams and write the necessary content in the report, as defined in the subtasks.	
Subtasks:		**Estimate:**
	DIFI-10: Create risk analysis	8h
	DIFI-11: Create role descriptions	5h
	DIFI-12: Write about programs used in project	6h
	DIFI-13: Create high level class diagram	13h
	DIFI-14: Create use case diagrams	9h
	DIFI-15: Create sequence diagram	7h
	DIFI-16: Create flow diagram	5h
	DIFI-17: Setting up workbench (Git, Eclipse, Glassfish)	17h
	DIFI-19: Create ER-diagram	11h
Total estimate:		**81h**

Table 6.2: Tasks, sprint 1.1

6.1.2 Project management

With a basic understanding of the planning process and the sketching of different diagrams started. The team realized that the understanding of the tasks were incomplete. Therefore the team had a Skype meeting with the customer, read books and browsed the Web for answers. Soon the team was back on track and able to finish early drafts of diagrams. The different tickets for the next sprint were made. To be able to estimate the team played "planning poker". This was an interesting exercise the team was excited to see how well planning poker structured the next two weeks.

Figure 6.1: Planning poker, estimating tickets while planning the sprint

Planning poker is used to estimate the time to implement, test and document a ticket. The person with the most knowledge presents the task and what it implies. Individually each of the team member estimate, followed by an open board were all members get to see what the rest of the team has estimated as well. The point of Planning Poker to ensure that everyone's opinion is heard. After the estimates are revealed everyone gives proof of their estimation in order to discuss and set the most appropriate estimate or, if there are huge variations, a re-run.

The customer contact is an accomplished programmer. This resulted in requirements as to what the system should do and what it should look like and requirements and recommendations as to what technologies to use. This left the team with less options as to how the system should be developed. Time that otherwise would be needed to research different developing tools and software could then be used to get started with the technical work. The customer also provided the team with relevant reading material that helped the development get started.

6.1.3 Result

The sprint is over and the preliminary report is delivered. The first drafts of the diagrams were created, and the understanding of the tasks increased. Still there are some problems understanding the connection between an "evaluation" and an "option". The team plans to solve this at the next customer meeting. Besides the problem with the architecture, the team had some struggle setting up the environment regarding Eclipse with Glassfish, selecting the correct development tool versions, configure Git and other administrative problems.

Throughout this first sprint the main challenge was to understand the assignment given from the customer. The good communication with the customer contact was important to get more understanding about the problem. This meant first of all to understand the concept of the application, and secondly how to approach the tasks.

6.1.4 Product backlog

After the sprint is over, the remaining tasks in the product backlog are:

Task ID	**Task**	**Priority 1-5**
DIFI-21	JPA: Create JPQL queries	3
DIFI-22	JPA: Database connection	4
DIFI-23	JPA: Create database	4
DIFI-24	JPA: Create the JPA classes	5
DIFI-27	REST: Create the OptionResource class	3
DIFI-29	REST: Create the EvaluationGroupResource class	3
DIFI-31	REST: Create the EvaluationSetResource class	3
DIFI-33	REST: Create the FileResource class	1
DIFI-35	REST: Create the CommentResource class	3
DIFI-37	REST: Create the EvaluationResource class	3
DIFI-40	JPA: Create connection between Drupal and REST	2
DIFI-67	REST: JSON communication with objects	3
DIFI-68	REST: Connect beans and Glassfish	3
DIFI-69	Create prototype to GUI	3

Table 6.3: Product backlog

6.2 Sprint 2.1

Sprint 2.1	**Dates**
Sprint planning	Wednesday, February 1st
Timeline	February 6th - February 17th

Table 6.4: Dates, Sprint 2.1

6.2.1 Sprint backlog

The goal this sprint was to start the implementation. This included a lot of self study.

Ticket:	DIFI-20	
Task:	JPA	
Description:	The JPA classes are the classes that are equivalent with the tables in the database. Create the database and JPA classes, and set up a persistent connection between the database and the JPA classes. This will create the database tables on the basis of the JPA classes. The creation of the JPA classes will also involve creating JPQL-queries so that is possible to retrieve information from the database.	
Subtasks:		**Estimate:**
	DIFI-21: Create JPQL-queries	8
	DIFI-22: Establish the database connection	8
	DIFI-23: Create a database	8
	DIFI-24: Create JPA classes for creating attributes and tables	8
	Sum, DIFI-20	**32h**

Ticket:	DIFI-64	
Task:	REST: Create resource classes	
Description:	The resource classes are the classes that is is going to be the REST implementation. Create methods for dealing with http request (PUT, GET and DELETE). These classes are going to use the Enterprise Java Beans to submit/retrieve data from the JPA entities.	
Subtasks:		**Estimate:**
	DIFI-27: Create OptionResource	5
	DIFI-29: Create EvaluationGroupResource	10
	DIFI-31: Create EvaluationSetResource	10
	DIFI-35: Create CommentResource	10
	DIFI-37: Create EvaluationResource	15
	Sum, DIFI-64	**50h**

Ticket:	DIFI-67	
Task:	REST: JSON communication with objects	
Description:	Create working code that handles JSON in the rest architecture. Figuring out how JSON communicate, both sending and receiving JSON as strings.	
	Sum, DIFI-67	**16h**

Ticket:	DIFI-68	
Task:	JPA - Create and connect EJBs to our MySQL database with Glassfish	
Description:	Enterprise Java Beans (EJBs) was selected to manage the database, and this EJB is connected through an entity in our MySQL database. One EJB is necessary for each object, which gives a total of 6 EJBs.	
	Sum, DIFI-68	**25h**
Total estimate:		**123h**

Table 6.5: Tasks, sprint 2.1

6.2.2 Testing

Visual testing was used to test if the database and the EJBs worked as intended. This was done thoroughly by testing code written by other team member's.

6.2.3 Design and implementation

During the sprint the MySQL database was created. The database was easy to create at one of NTNU's MySQL servers.

The team read about Java Persistence API (JPA) and used the new knowledge to set up the entities using JPA. The Enterprise Java Beans (EJB) was created, and used to manage the connection and synchronization with the database. The team also created the parts of the REST service, and configured this to fetch and insert data through the EJB's via HTTP requests. JPA is a very easy and smart way to handle database mappings. It is very commonly used when developing a Java Enterprise application and is often combined with Enterprise Java Beans (EJB). Creating the persistence.xml file and setting up the Glassfish server correctly were the most challenging tasks. The persistence file contains the necessary information to set up the connection with the database and which classes that should be persisted.

JPA objects are mostly simple objects with private fields used by get and set methods. What makes the objects special are the "@Entity" annotation that specifies that the object is a database entity. They are mostly very easy to set up like a simple java object, just with a few extra annotations.

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;

@Column(name = "Text", length = 2000)
private String text;
```

Figure 6.2: Example of JPA code

In the example in figure 6.2, the first annotation is the "@ID" tag. This annotation specifies that the attribute "id" is going to be the primary key in the database table. The annotations defines that the id is automatic incremental, which means that the id number increases by one with every new row that is added to the database. The second one is a normal text column, and the annotation defines that column can have a maximum length of 2000 characters.

The objects can also contain queries to get specific information from the database. These are not ordinary

SQL queries but Java Persistence Query Language (JPQL) queries. The main difference between normal SQL and JQPL is that normal SQL uses the tables and columns in the database in the query, but JQPL uses the entity name and fields from JPA object. Figure 6.3 shows an example of a JPQL query.

```
@NamedQueries ({
        @NamedQuery(name = Evaluation.BY_ID, query = "SELECT e FROM Evaluation e WHERE e.id = :id")
})
```

Figure 6.3: Example of JPQL query

Enterprise JavaBeans (EJB) contains the logic of a Java Enterprise application. In this project the EJBs are used to perform queries, save objects to database and delete them. There were a lot of changes in the Enterprise JavaBeans the last years, and therefore it was not that easy to find good examples of how to use the newest version of EJB, version 3.1. EJB 3.0 used some extra interfaces to do the work, but in version 3.1 these interfaces were not longer necessary. Even though it was not easy to find a good example, the JavaBeans were fairly simple to set up and use. The stateless annotation let us use the EJB class from other classes without the need to instantiate the EJB in each class it would be used. This is done by using the "@EJB" annotation in another class.

```
@EJB EvaluationEJB evaluationEJB;
```

Figure 6.4: Example of declaration of EJB

By calling the code in figure 6.4 from another class it let us use all of the EJB's methods.

To be able to persist the objects to the database there is a class that all the EJBs extends. This class contains the EntityManager which is defined with the annotation showed in figure 6.5.

```
@PersistenceContext(unitName = "kvrs-vurdering")
protected EntityManager em;
```

Figure 6.5: Example of declaration of the EntityManager

In the beginning of the project the team had a superclass with the most common methods like delete and save, that all the EJBs used. Later the team realized that the EJBs had to get their own methods, because there were different requirements in terms of validation of which fields in an object that needed data.

The team's impression of Enterprise JavaBeans is that they are easy to use as long as you know some small lines of code that defines the EJB.

Java API for RESTful Web Services (JAX-RS) is an API that provides support for creating web services according to the Representational State Transfer architectural style. In the beginning of the project the team looked at this part as the most demanding and time-consuming part of the REST-service implementation. The team was prepared to create all the necessary methods to get the REST service up and running, but soon realized that most of the functionality needed was already given to us by JAX-RS. When the team understood how the REST service worked, the classes were fairly easy to set up. The REST classes contains methods for using the GET, PUT and DELETE HTTP methods. This is set up by using the appropriate annotations.

```
@GET
@Path("{id}")
@Produces(MediaType.APPLICATION_JSON)
public Evaluation get(@PathParam("id") int id) {
    Evaluation evaluation;
    try {
        evaluation = (Evaluation)evaluationEJB.getByID(id);
    } catch (Exception e) {
        throw new WebApplicationException(500);
    }
    if (evaluation == null) {
        throw new WebApplicationException(404);
    } else return evaluation;
}
```

Figure 6.6: Example of a method in the RESTful API

The example in figure 6.6 shows a GET method for getting a specific object defined by an id. The "@GET" annotations expresses that this is a method that would get called if the HTTP method GET with the path leading to this class. The path is defined by the "@Path" annotation. The "@Produces" annotation specifies what the method returns. In this project JavaScript Object Notation (JSON) was used to represent the data when transferring through the HTTP protocol. Therefore the "@Produces(MediaType.*APPLICATION_JSON*)" was used in all the methods that was suppose to return something.

The customer required the user interface to be in Norwegian. Since the paths used to query the REST-service are the user interface of the service, we chose to implement the paths in Norwegian instead of English.

The team's impression is that the RESTful web service is elegant and simple when passing information over the Internet. The team had no problems developing the web service, but at first made it more complicated than necessary. In the beginning the conversion from java objects to JSON strings were manually done by using an external library, this was before the team discovered the "@Produces" annotation that does the job automatic. JAX-RS caused a minor problem; the format of the JSON string is dependent of the number

of elements being sent. If the team is supposed to list all the evaluations and there is only one, the JSON string does not contain any list, just the one element. If there are several elements, the JSON string contains a list with the elements. Ideally the format of the JSON string would be the same despite on one or several elements, but this problem was solved by checking if the JSON string was a list or not in the front-end client.

Prototypes of the main components in the design were drawn, and the team planned how the front-end layers should be structured.

6.2.4 Project management

The team underestimated the tasks this sprint (more discussed in the sprint result section), and wondered if it was their own work and effort that lacked. Had the team actually worked as much as promised? The team discussed it, and it turned out that almost everyone had met some problems during this sprint. The problems had not been easy to solve, which again decreased the motivation. The team discussed how to solve these kind of problems and agreed that at some point, it is a lot easier to team up with someone else and solve the problem together, by for example pair programming. To keep the motivation up, the team scheduled a team building.

During the first sprint, and the beginning of the second, the team had some problems with members that were late to meetings. The rest of the team had to wait for the other members since they had not heard from them. As a result, it were decided to make a collaboration contract. The highlights of the agreement is that all team members must be ready for the stand-up 10.30, tell in advance about absence, and if some are not present at a meeting they have to make sure they are updated on the work. If the points in the agreement are broken, the team will decide if the member should be punished. The punishment is either one bottle of beer (0.33 l) for each team member, or to buy something the team can enjoy at a meeting (equivalent the cost of beer). A copy of the contract is attached, Appendix B.

6.2.5 Result

The team had estimated 20 hours per person, and the sprint lasted for two weeks. With a focus factor on 0,65 there was a total of 156 hours to allocate the tickets. At the end of the sprint, the team had only finished 65% of the tasks planned. It was obvious that the tickets were underestimated, and the remaining tasks had to be moved over to sprint 2.2. The wrong time estimate was a good experience for the team to bring on further in the project. The team needs to calculate more time for reading books, and handle problems that occur more often than hoped. It was a big mistake to only estimate the main tasks, and not

the subtasks. For example was the tickets DIFI-20 and DIFI-64 divided into several sub tasks, but only the main tasks were estimated. The team could have chosen whether or not to include the different tasks if the estimates on the workload had been more accurate. Task DIFI-20 will be moved to sprint 2.2 because of the underestimation. This causes changes to the original plan.

6.2.6 Product backlog

After this sprint, the remaining tasks in the product backlog are:

Task ID	**Task**	**Priority 1-5**
DIFI-33	Create the FileResource class	1
DIFI-40	Create connection between Drupal and REST	4
DIFI-41	Implement a way to transfer data between REST and JPA	4
DIFI-64	REST: Complete the rest of resource classes	4
DIFI-66	HTTP methods between machines	4
DIFI-69	Create prototype to GUI	5
DIFI-73	Create schema to create EvaluationSet	3
DIFI-74	Show all data on Evaluation	3
DIFI-76	Show all data on EvaluationSet	3
DIFI-78	Drupal: Create comment module	3
DIFI-79	Show all comments	2
DIFI-80	Create scheme to create EvaluationGroup	3
DIFI-81	Create scheme to create Evaluation	2
DIFI-83	Get the collaboration agreement signed and scanned	1
DIFI-86	Get an overview and create a plan to fill the report	3
DIFI-87	Solve the id crash between REST and Drupal	2
DIFI-88	Write the needed information in the report according to the plan	2
DIFI-91	Change PUT methods to use JAX-RS	3
DIFI-92	Change variables and method names to be more consistent	1

Table 6.6: Product backlog

6.3 Sprint 2.2

Sprint 2.2	**Dates**
Sprint planning	Monday, February 20th
Timeline	February 20th - March 9th

Table 6.7: Dates, Sprint 2.2

6.3.1 Sprint backlog

During this sprint is was two main goals; to create the first Drupal module and to have a running application to show the customer contact at Friday March 2nd.

Ticket:	DIFI-40	
Task:	REST/Drupal - Implement a way to transfer data between REST and Drupal	
Description:	Java objects have to be parsed into strings so they are easier to send over the Internet. A method that parses objects to string, and that parse strings back to the object it belonged, must be implemented in the resource classes. In Drupal the team need to figure out how to make a JSON string from the data in the PHP fields, and how to decode JSON strings that comes from the resource classes.	
	Sum, DIFI-40	**14h**

Ticket:	DIFI-41	
Task:	REST/JPA - Implement a way to transfer data between REST and JPA	
Description:	EJBs need to have some kind of receive method and a persist method to store the JSON strings in the database.	
	Sum, DIFI-41	**14h**

Ticket:	DIFI-64	
Task:	REST - Fix the resource classes we did not finish in sprint 2.1	
Description:	Implement the methods put, get, getAll and delete in the 6 resource classes. All these methods need to use JSON as interchange format. Only put and get in the resource classes evaluationGroup, evaluationSet and comment was implemented in sprint 2.1, so the rest of the logic must be implemented in this sprint.	
	Sum, DIFI-64	**19h**

Ticket:	DIFI-66	
Task:	HTTP methods between machines.	
Description:	Use HTTP methods (GET, PUT, DELETE) to get, set and change data on other machines.	
	Sum, DIFI-66	**6h**

Ticket:	DIFI-69	
Task:	Create prototype for the Drupal interface	
Description:	Create a prototype to help develop the graphical user interface in Drupal.	
	Sum, DIFI-69	**9h**

Ticket:	DIFI-80	
Task:	Drupal - Create module for evaluation group	
Description:	A logged in user can create a new evaluation group. Name is the only required field, created and id is generated by the system, active is default "true". After an evaluation group is created, the user will go to the evaluation group view and see the new evaluation group. A logged in user can also list all evaluation groups, expand one group and see all the evaluation sets inside a group, and sort this list by created date, active and name.	
	Sum, DIFI-80	**21h**
Total estimate:		**83h**

Table 6.8: Tasks, sprint 2.2

6.3.2 Testing

As an early test stage the team used the web-browser to see if the REST server returned correct responses.

6.3.3 Design

After consulting the customer contact the team had an architectural choice to make. Either the data could be viewed as pure information in Drupal or the team could create the structure in Drupal as well (by creating content pages). This was the team and group 6's (Appendix A) decision to make. A requirement was to solve this in the same way as group 6 which worked on another part of the same system. To use Drupal as a pure view is less scalable than if one wanted to use other modules combined with the data. On the other hand this solution will most probably be the easiest one (regarding code). After a discussion the team decided to try to build up the content pages from scratch - group 6 agreed to this.

6.3.4 Project management

The customer contact visited Trondheim in conjunction with a Drupal course he was holding. The team took this opportunity to give a short presentation of the project in addition to taking the course. The customer was pleased with the result of the application so far and he was also impressed by the progress made in the amount of time.

The supervisor asked for weekly status reports. The reports should contain an overview over which tasks that was planned and which tasks that were finished. In addition, progress according to the plan, how the cohesion in the team had been and how the weekly meeting with the customer contact went should be included.

6.3.5 Result

The team managed to meet the two main goals this sprint. The team had a running application, as well as a plan to solve the remaining tasks. The customer contact was comfortable to know that the architecture and the planning were up to speed. Further the team will focus on expanding previous work to meet the requirements. The team had some problems with the implementation of Drupal, there was a lot to read up on and implementation was not as easy as expected.

6.3.6 Product backlog

After this sprint, the remaining tasks in the product backlog are:

Task ID	**Task**	**Priority 1-5**
DIFI-33	Create the FileResource class	2
DIFI-73	Create schema to create EvaluationSet	3
DIFI-74	Show all data on Evaluation	3
DIFI-76	Show all data on EvaluationSet	3
DIFI-78	Drupal: Create comment module	3
DIFI-79	Show all comments	2
DIFI-81	Create scheme to create Evaluation	2
DIFI-83	Get the collaboration agreement signed and scanned	1
DIFI-85	Drupal - Manage date field in all modules	3
DIFI-86	Get an overview and create a plan to fill the report	3
DIFI-87	Solve the id crash between REST and Drupal	2
DIFI-88	Write the needed information in the report according to the plan	2
DIFI-91	Change PUT methods to use JAX-RS	3
DIFI-92	Change variables and method names to be more consistent	1
DIFI-93	Get the project running with maven	4
DIFI-94	Create GANT diagram	3
DIFI-95	Fix “save” bug in JPA	5
DIFI-96	Drupal: Change the edit field to retrieve data from the REST server	3
DIFI-97	Drupal: Hide the fields ID and created in form	1
DIFI-98	REST: Add the possibility to change an object without the need of all the relations	3
DIFI-99	Drupal: Get a object in return when a new object is stored in the database	3
DIFI-100	Drupal: Get all nodes from RESTserver and not from Drupal’s database	4

Table 6.9: Product backlog

6.4 Sprint 3.1

Sprint 3.1	**Dates**
Sprint planning	Monday, March 12th
Timeline	March 12th - March 23rd

Table 6.10: Dates, Sprint 3.1

6.4.1 Sprint backlog

Ticket:	DIFI-76	
Task:	Drupal - Show all data on EvalutionSet.	
Description:	Make methods in Drupal to show all data in an EvaluainoSet.	
	Sum, DIFI-76	**5h**

Ticket:	DIFI-85	
Task:	Drupal - Manage field “date” in all the modules.	
Description:	Make the field “date” invisible in the add form, and visible in views.	
	Sum, DIFI-85	**8h**

Ticket:	DIFI-86	
Task:	Make an overview of what the content in the final report should be.	
Description:	Make an overview of what the content in the final report should be, and make a plan on how the work in the next weeks must be to meet this goal.	
	Sum, DIFI-86	**10h**

Ticket:	DIFI-91	
Task:	REST: Change the put method in the resource classes.	
Description:	The REST put methods are now using an external jar file to receive a JSON-string as input. The methods to utilize JAX-RS built-in conversion from JSON string to Java objects need to be changed. The task includes making an adapter for handling relations between entities during marshalling/unmarshalling.	
	Sum, DIFI-91	**17h**

Ticket:	DIFI-92	
Task:	Change method- and variable names to be more consistent.	
Description:	Makes cleaner code, and the code easier to read later.	
	Sum, DIFI-92	**2h**

Ticket:	DIFI-93	
Task:	JPA - Maven implementation.	
Description:	Up to this point the team have used a normal dynamic web project in eclipse. The customer required Maven for easy packaging. The team needs to find out how dependencies in maven work, how to make a eclipse project in maven, how to redeploy the .war file on the Glassfish server after compiling.	
	Sum, DIFI-93	**5h**

Ticket:	DIFI-94	
Task:	Gantt chart.	
Description:	Make a gantt chart to illustrate start and finish dates of the most important parts of the system.	
	Sum, DIFI-94	**4h**

Ticket:	DIFI-95	
Task:	REST: Fix the “save”-bug in JPA.	
Description:	When the system saves a new entity to REST the many-to-one relations do not update. Example: If the system saves a new instance of EvaluationSet, the parent EvaluationGroup does not update. The evaluation group does not get the new evaluation set in its list of evaluation set before the server has been restarted.	
	Sum, DIFI-95	**6h**

Ticket:	DIFI-96	
Task:	Drupal: Change where to retrieve data from.	
Description:	Change the fields in edit to get data from REST and not from Drupal’s own database.	
	Sum, DIFI-96	**8h**

Ticket:	DIFI-97	
Task:	Drupal: Hide fields in form.	
Description:	Make the fields ID and created hidden in form, but still visible in view.	
	Sum, DIFI-97	**3h**

Ticket:	DIFI-98	
Task:	REST: Change an entity without sending all information.	
Description:	Add the ability to change an entity without the need to send all relational data.	
	Sum, DIFI-98	**2h**

Ticket:	DIFI-99	
Task:	REST: Return the inserted object when PUT is used	
Description:	When the system creates a new object or edited an old object, the group want to return the object from the REST-server.	
	Sum, DIFI-99	**4h**

Ticket:	DIFI-100	
Task:	Drupal: Show nodes from REST-server instead of Drupal	
Description:	When creating an object via Drupal and displaying it to the user is the data from Drupal's own database used. This should be the data from the REST-server, but Drupal is closely integrated with it's own database.	
	Sum, DIFI-100	**15h**
Total estimate:		**84h**

Table 6.11: Tasks, sprint 3.1

6.4.2 Testing

The team tried to make some unit testing for the EJB's. Because of problems connecting the test classes to another test database the team could not proceed to execute any automatic tests.

6.4.3 Design

The customer required the team to use JAX-RS API to convert Java objects to and from JSON. This would make the logical code easier. This was the original solution, but because of problems with relations regarding Java objects, the team had chosen to manually try to do the conversion. In Drupal the focus was still on expanding the first module.

The team struggled with fixing bugs related to relations between Java objects, especially during cascading updates. Up until now Java objects had lists containing all related objects, e.g an evaluation group would contain a list of all it's evaluation sets, as well as evaluation sets having the related evaluation group as an attribute. During this sprint the team figured out that this was not necessary, since these lists were never in use. We solved the bugs by removing the lists and implementing simple integer attributes containing the primary key (id) of the related object, e.g all evaluation sets will contain the primary key of the evaluation group it belongs to. If we ever need to know all the evaluation sets in an evaluation group we just query the database instead of using the lists.

There were problems when starting to use maven, first the team had to restructure all the folders holding the classes to be more maven friendly, and second the pom.xml file needed to be configured. This was very time-consuming to get right.

6.4.4 Project management

Because of a lot of dependencies within the project, problems with Glassfish installations, and the customer's wish, the team started to use Maven to build and manage the project.

Since the customer is established in Sogndal the communication has mainly been over Skype, email and phone. This could have caused difficulties regarding progress during the develop process, but granting the customer access to the Git repository solved this. The Skype meetings were also vital since this was the team's main communication with the customer.

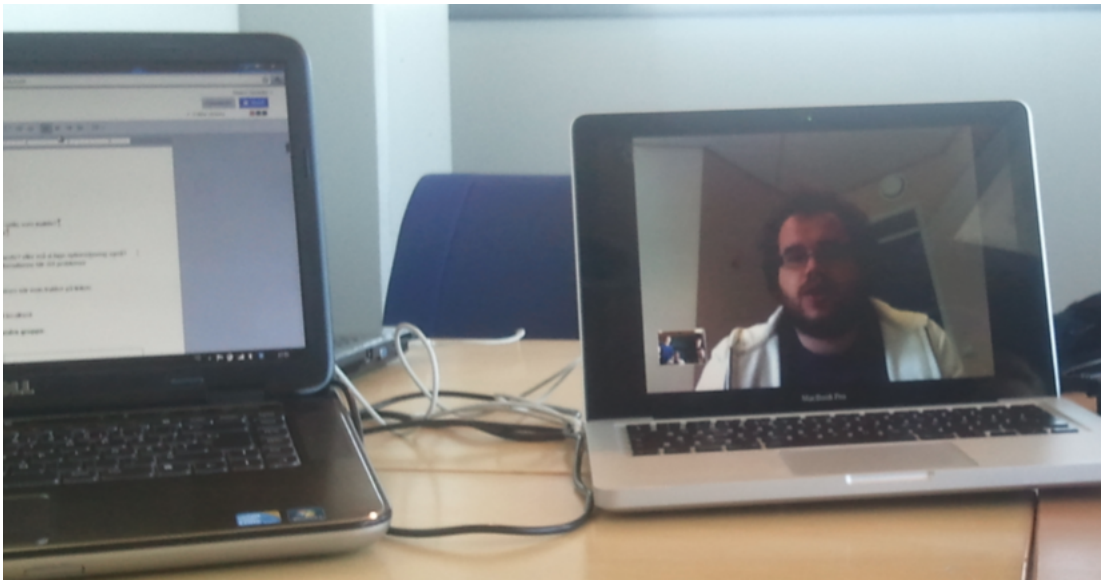


Figure 6.7: Skype meeting with the customer contact

6.4.5 Result

The plan the team made for the rest of the report became redundant because of the feedback from the supervisor. Since the team got some constructive criticism on the mid-term delivery the team chose to focus more on writing the report. To set up Maven and use it together with Eclipse turned out to be some struggle at first but was solved by an eclipse plugin.

The team did not manage to finish DIFI-100 during sprint 3.1. The task is therefore transferred back to the product backlog.

6.4.6 Product backlog

After this sprint, the remaining tasks in the product backlog are:

Task ID	**Task**	**Priority 1-5**
DIFI-33	Create the FileResource class	2
DIFI-73	Create schema to create EvaluationSet	3
DIFI-74	Show all data on Evaluation	3
DIFI-78	Drupal: Create comment module	3
DIFI-79	Show all comments	3
DIFI-81	Create scheme to create Evaluation	4
DIFI-83	Get the collaboration agreement signed and scanned	1
DIFI-87	Solve the id crash between REST and Drupal	2
DIFI-88	Write the needed information in the report according to the plan	2
DIFI-100	Drupal: Get all nodes from RESTserver and not from Drupal's database	4
DIFI-102	Report: Product	4
DIFI-103	Report: Documentation	4
DIFI-104	Report: Customer	4
DIFI-105	Report: Group	4
DIFI-106	Report: Process	4
DIFI-107	File handler	3
DIFI-129	Write tests to EJB	2

Table 6.12: Product backlog

6.5 Sprint 3.2

Sprint 3.2	**Dates**
Sprint planning	Monday, March 26th
Timeline	March 26th - April 13th

Table 6.13: Dates, Sprint 3.2

6.5.1 Sprint backlog

Ticket:	DIFI-73	
Task:	Drupal - Create module for evaluation set	
Description:	A logged in user can create a new evaluation set. Title is the only required field, created and id is generated by the system, subject is optional and active is default set 'true'. After an evaluation set is created, the user will go to the evaluation set view and see the new evaluation set. A logged in user can also list all evaluation sets in an evaluation group, and sort this list by created date and title.	
	Sum, DIFI-73	**18h**

Ticket:	DIFI-74	
Task:	Drupal - Show all data on Evaluation	
Description:	A logged in user needs to have the possibility to list all evaluations and the data associated to it. The attributes wanted are created, name, criteria, selectedOptionID, and revisionOf.	
	Sum, DIFI-74	**5h**

Ticket:	DIFI-78	
Task:	Drupal - Create module for comment.	
Description:	A logged in user can create a new comment on an evaluation. Name is the only required field, the attribute created, id, and userID is generated by the system, and deleted is default 'false. After a comment is created, the user will go to the comment view and see the evaluation with the new comment bellow. A logged in user will also be able to see all the comments inside an evaluation.	
	Sum, DIFI-78	**14h**

Ticket:	DIFI-79	
Task:	Drupal - Show all comments	
Description:	Each comment has a relation to an evaluation. According to the FR-1.16, needs the user to have the possibility to show all comments based on an evaluation.	
	Sum, DIFI-79	**5h**

Ticket:	DIFI-81	
Task:	Drupal - Create module for evaluation	
Description:	A logged in user can create, edit, delete a new evaluation inside an evaluation set. Name and selectedOption are required fields, created, id, edited, deleted, userID, revisionOf is generated by the system, active is default 'true' and text is optional. After an evaluation is created, the user will go to the evaluation view and see the new evaluation. A logged in user can also list all evaluations inside an evaluation set, and sort this list by created date, active and title.	
	Sum, DIFI-81	**18h**

Ticket:	DIFI-83	
Task:	Translate, sign and scan the collaboration agreement	
Description:	The team has created a collaboration agreement between the group members and this needs to be translated from norwegian to english, signed by the group members and scanned.	
	Sum, DIFI-83	**2h**

Ticket:	DIFI-100	
Task:	Drupal: Show nodes from REST-server instead of Drupal	
Description:	When creating an object via Drupal and displaying it to the user is the data from Drupal's own database used. This should be the data from the REST-server, but Drupal is closely integrated with it's own database.	
	Sum, DIFI-100	**15h**

Ticket:	DIFI-102	
Task:	Report: Product	
Description:	After feedback from the customer the team needed to provide a more detailed requirement description. Be more consequent on the ID and prioritization between tasks. Create an overall architecture diagram and provide more detailed information about each task in the iterations	
	Sum, DIFI-102	**10h**

Ticket:	DIFI-103	
Task:	Report: Documentation	
Description:	After feedback from the customer we needed to implement some new features in the documentation phase. The comments were introduction, pre-study, project management, requirements, appendixes, and other minor improvements.	
	Sum, DIFI-103	**4h**

Ticket:	DIFI-104	
Task:	Report: Customer	
Description:	After feedback from the customer the team needed to improve the information about the customer management and supervisor interaction.	
	Sum, DIFI-104	**4h**

Ticket:	DIFI-105	
Task:	Report: Group	
Description:	After feedback from the customer the team needed to improve the information about each member and the collaboration agreement be translated form norwegian til english.	
	Sum, DIFI-105	**6h**

Ticket:	DIFI-106	
Task:	Report: Process	
Description:	After feedback from the customer the team needed to improve their documentation about the process. The tickets were integration testing in the teat plan, introducing other development methods possible, GANTT diagram, activity plan and a updated risk analysis.	
	Sum, DIFI-106	**12h**
Total estimate:		**79h**

Table 6.14: Tasks, sprint 3.2

6.5.2 Testing

Visual testing was used when creating Drupal modules and group members testing others group members functions when issues were finished. Especially cURL was tested thoroughly.

6.5.3 Design

The work with Drupal continued. As everything else was nearing completion, Drupal had main focus together with the report. Not much progress was made, which in turn caused some frustration. In between the sprint 3.2 and 4.1 new tasks were added and old tasks removed. One of the tasks removed was DIFI-100 because of a design change in Drupal. Explanation about this and the new tasks in the product backlog is located in section 4.1.

6.5.4 Project management

The team was informed that Difi had, and would be doing, some internal changes. This had some effect on the progress. For 3-4 weeks during the development the team had almost no contact with the customer. This could have resulted in a very negative impact on the development, but since this happened towards the end of the project, combined with the fact that it was somewhat expected, the team managed to keep up the workflow. This was solved by working on tasks that the team knew how to finish, and postponed things that needed guidance.

It was clear from the start that although the team was doing well with the practical work, the report still needed a lot of work. Through out the first meetings this was the main feedback, but the team also got the impression that the supervisor was satisfied with the work done and the teamwork. The feedback has gradually gotten more specific as to what parts of the report that need improvement. After the third delivery we got an extensive list of good and bad things with the report, and this feedback was essential for how the work with the report continued.

6.5.5 Product backlog

After this sprint, the remaining tasks in the product backlog are:

Task ID	Task	Priority 1-5
DIFI-33	Create the FileResource class	3
DIFI-88	Write the needed information in the report according to the plan	2
DIFI-107	File handler	4
DIFI-108	Drupal: Evaluation	4
DIFI-109	Drupal: Evaluation set	4
DIFI-110	Drupal: Evaluation group	5
DIFI-111	Drupal: Comment	3
DIFI-112	Drupal: Data flow	3
DIFI-125	Fix error handling	2
DIFI-126	Implement revision functionality	3
DIFI-127	Fix errors when running tests on server	2
DIFI-129	Write tests to EJB	3
DIFI-130	Transfer data from REST-server to Drupal	5
DIFI-131	Remove all active an id fields	2
DIFI-136	Show inactive evaluation groups	3

Table 6.15: Product backlog

6.6 Sprint 4.1

Sprint 4.1	Dates
Sprint planning	Monday, April 16th
Timeline	April 16th - April 27th

Table 6.16: Dates, Sprint 4.1

6.6.1 Sprint backlog

Ticket:	DIFI-108	
Task:	Drupal: Evaluation.	
Description:	Create the evaluation.inc file in Drupal by using php. This file includes forms, tables and all the functionality needed in the Evaluation section.	
	Sum, DIFI-108	**16h**

Ticket:	DIFI-109	
Task:	Drupal: EvaluationSet.	
Description:	Create the evaluationset.inc file in Drupal by using php. This file includes forms, tables and all the functionality needed in the EvaluationSet section.	
	Sum, DIFI-109	**11h**

Ticket:	DIFI-110	
Task:	Drupal: EvaluationGroup.	
Description:	Create the evaluationgroup.inc file in Drupal by using php. This file includes forms, tables and all the functionality needed in the EvaluationGroup section.	
	Sum, DIFI-110	**11h**

Ticket:	DIFI-111	
Task:	Drupal: Comment.	
Description:	Create the comment.inc file in Drupal by using php. This file includes forms, tables and all the functionality needed in the comment section.	
	Sum, DIFI-111	**13h**

Ticket:	DIFI-112	
Task:	Drupal: Data flow.	
Description:	Implement all needed buttons to navigate between the pages. This also includes passing the needed data to the next page.	
	Sum, DIFI-112	**4h**

Ticket:	DIFI-125	
Task:	Fix error handling.	
Description:	Handle all possible errors in an appropriate way.	
	Sum, DIFI-125	**2h**

Ticket:	DIFI-126	
Task:	Drupal - Implement revision functionality.	
Description:	Add the possibility to show all revisions of an Evaluation. This also includes the functionality to add new revisions.	
	Sum, DIFI-126	**20h**

Ticket:	DIFI-127	
Task:	Fix errors when running tests on the server.	
Description:	Find a way to solve the errors.	
	Sum, DIFI-127	**15h**

Ticket:	DIFI-129	
Task:	Write tests to EJB.	
Description:	Write all remaining tests to check if the EJB classes are working.	
	Sum, DIFI-129	**20h**

Ticket:	DIFI-130	
Task:	Transfer data from REST-server to Drupal	
Description:	Find a fast en stable way to transfer JSON strings from REST to Drupal	
	Sum, DIFI-130	**8h**

Ticket:	DIFI-131	
Task:	Remove all visible id and active fields.	
Description:	Remove all the fields showing id and active in all pages in Drupal.	
	Sum, DIFI-131	**2h**

Ticket:	DIFI-136	
Task:	Show inactive Evaluation Groups.	
Description:	Add a button to show all inactive Evaluation Groups.	
	Sum, DIFI-136	**3h**
Total estimate:		**123h**

Table 6.17: Tasks, sprint 4.1

6.6.2 Testing

Testing in this sprint mostly consisted of testing the functionality through the newly created interface in Drupal by clicking around in the web browser.

6.6.3 Design and implementation

To communicate between Drupal and the REST-server the team chose to use cURL. Using URL syntax, cURL makes it possible to get and send information over HTTP. With cURL sending/receiving the JSON string to/from the rest server will ensure that the system gets a fast, stable and easy way of transferring data. Figure 6.8 shows an example of a cURL request where you try to get an evaluation group by an id, extracts the http code and the data and decode the JSON string to an array, called evaluationgroup. All the associated values to an evaluation group is now inside the array and are available for use.

As you can see in figure 6.9, the team is using a method called kvrs_curl in the file kvrs.curl. Below is kvrs_curl, which is the actual cURL request, with the variable 'KVRS_BASE' being the IP address of the rest server and port number.

In this sprint some major changes in the Drupal code were necessary, as described below. This meant that the team had to throw away nearly all of the current Drupal code and start over.

```php
function kvrs_curl($url, $method, $data='') {

    $curl = curl_init('http://' . variable_get('KVRS_BASE') . $url);
    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $method);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);

    if ($method == 'PUT') {
        $json = json_encode($data);
        curl_setopt($curl, CURLOPT_POSTFIELDS, $json);
        curl_setopt($curl, CURLOPT_HTTPHEADER, array(
            'Content-Type: application/json',
            'Content-Length: ' . strlen($json))
        );
    }
    $response = curl_exec($curl);
    $http_code = curl_getinfo($curl, CURLINFO_HTTP_CODE);
    curl_close($curl);
    return array($http_code, $response);
}
```

Figure 6.8: The teams implementation of cURL

```php
$data = kvrs_curl('vurderingsgruppe/' . $id, 'GET');
$http_code = $data[0];
$json = $data[1];
$evaluationgroup = json_decode($json, true);
```

Figure 6.9: Example of how to use curl in php

6.6.4 Project management

The team had some problems getting in touch with the customer and needed to get a confirmation on the Drupal progress so far. After a few weeks the team managed to get contact with the customer and he explained that it did not matter which of the implementation ways used, as long as the team collaborated with the other Difi group, group6, and implemented Drupal with the same structure and approach.

The team called a meeting with group 6 for a discussion and clarification, and the team members must admit the high frustration level. Still the team kept an open mind before the meeting. Even though group 6 made a huge mistake (from our team's point of view) not to discuss their decision or at least inform the team, the result of the meeting was to change the Drupal code structure (see section 5.2.2). The customer agreed upon this decision. The team and group 6 decided to create a convention agreement to simplify the work for the people working at Difi when the two systems are merged together.

Since the team now had gotten information about the structure, from the convention agreement, the work on the evaluation group started again. Things went good and the first of five entities in our module were finished quite quickly. Developing the rest of the entities and connecting them together was now the teams new goal. This worked out without any huge problems and things started to actually look and act like a real system. Next focus is making the site more user friendly, like inserting buttons, redirecting to the correct page after an action, creating navigation bars, sorting tables and all the rest of the little issues that takes a lot more time than estimated. Concurrent, everyone reported back if they experienced some errors, bugs, or some functionality that should be different.

The major changes in Drupal resulted in several of the Scrum tickets expire (both in the product backlog and the completed tickets in earlier sprints). New ones had to be added. The teams reconciled and the team invited group 6 to the next team building.

6.6.5 Result

The team managed to create all the planned pages in Drupal. Although not all the functionalities were working as well as they should, the pages were visually looking very good. It turned out that it was a lot easier to work with PHP in Drupal, rather than using nodes and hooks.

6.6.6 Product backlog

After this sprint, the remaining tasks in the product backlog are:

Task ID	**Task**	**Priority 1-5**
DIFI-33	Create the FileResource class	5
DIFI-88	Write the needed information in the report according to the plan	2
DIFI-107	File handler	5
DIFI-128	Fix methods in FileResource, FileEJB and File to get the fileuploader working	5
DIFI-133	The code fails when an element no longer is returned as a list	3
DIFI-138	Multiupload of files to comment and evaluation	2
DIFI-148	Links to show where you are and the possibility to go back (breadcrumbs)	3
DIFI-149	Drupal: Bugfixing	3
DIFI-150	Drupal: Final testing	4

Table 6.18: Product backlog

6.7 Sprint 4.2

Sprint 4.2	**Dates**
Sprint planning	Monday, April 30
Timeline	April 30 - May 25

Table 6.19: Dates, Sprint 4.2

6.7.1 Sprint backlog

Ticket:	DIFI-33	
Task:	REST: FileResource.	
Description:	Create FileResource class in REST server. This class includes the functionality to use HTTP methods to GET, PUT, and DELETE files between the REST server and database.	
	Sum, DIFI-33	**12h**

Ticket:	DIFI-88	
Task:	Improve the report.	
Description:	Read through the report and find out what needs improvement, if there should be any changes, and where the report need more information.	
	Sum, DIFI-88	**40h**

Ticket:	DIFI-107	
Task:	Drupal: File manager.	
Description:	Create the file manager functionality in Drupal to handle downloading and uploading files from and to the REST server.	
	Sum, DIFI-107	**10h**

Ticket:	DIFI-128	
Task:	Fix FileResource, FileEJB and File to get the FileUploader working.	
Description:	Fix all the respective Java classes to to make the fileUploader functionality work.	
	Sum, DIFI-128	**10h**

Ticket:	DIFI-133	
Task:	Drupal: Code fails when an element no longer is returned as a list.	
Description:	Find a way to handle the inconsistent way of transferring JSON objects from JAX-RS when the sending list only contains one element.	
	Sum, DIFI-133	**4h**

Ticket:	DIFI-138	
Task:	Drupal: MultiUpload of files.	
Description:	Handle multiupload of files in Drupal by extending the existing form.	
	Sum, DIFI-138	**20h**

Ticket:	DIFI-148	
Task:	Drupal: Navigation bar	
Description:	Create a navigation tree that gives the user an overview over the structure in the system, for easy navigation and interaction.	
	Sum, DIFI-148	**5h**

Ticket:	DIFI-149	
Task:	Drupal: Bug fixing	
Description:	Fix all known bugs reviled by the Final testing	
	Sum, DIFI-149	**36h**

Ticket:	DIFI-150	
Task:	Drupal: Final testing	
Description:	Test all features in the system by integration, system and user testing to make sure the system is properly quality assured	
	Sum, DIFI-150	**26h**
Total estimate:		**163h**

6.7.2 Testing

The team has conducted unit-, integration-, system-, and usability testing this sprint. Because new components and functionality was created, unit testing was necessary. The new components needed to be integrated with the system, and this was tested through integration testing. The usability was tested at the same time as the system testing – one team member went through all the requirements and tested them in Drupal. Parts of the design on the Drupal site was changed because the members thought it would be more user-friendly to do it in another way. Some errors were discovered through the system testing, but they were quickly corrected.

6.7.3 Project management

Sunday 6th of May the team had an extraordinarily meeting where the customer demonstration was thoroughly planned.

The 7th of May the demonstration of the completed application was conducted through Skype. Four people represented the customer at the meeting: the customer contact, and three employees who will use the application in their work. The presentation was done together with group 6 (the other team working for the customer), and consisted of giving the customer a "tour" through the application. Starting with data flow diagrams, followed by sending HTTP request to the REST server, and analyzing the output. Further was the graphical user interface in Drupal demonstrated, this showed that all the requirements was met. The customer contact and the customer representatives were very satisfied with the demonstration and the application. They requested that the final product should be delivered to them as soon as possible.

6.7.4 Result

During the sprint 4.2, the team successfully completed the fileuploader and the tasks related to the functional requirements FR-20 - FR-24. In addition, comprehensive testing and solid bug prevention was completed. Improving the report and giving the code a final touch has required a lot of effort but the goals for this sprint was reached with satisfying results.

Working with Drupal for the last 3 months, the team has experienced Drupal's advantages and disadvantages. The impression is that as long as you use Drupal's predefined GUI and the click and drag features everything works as a charm. When you try to implement the functions on your own, it gets trickier. Drupal does not support manipulating functions programmatically very well, and the documentation is incomplete. Drupal

has been a lot more work than the team first estimated, and has caused a lot of frustration in times, but as the time went by and the team got more and more knowledge about how Drupal actually work, things went of course better. We are very pleased with our final result and hoping that the customer is too.

6.7.5 Git

The team has worked on several branches, and the merging when the task is completed has been a fantastic feature. The team is pleased with Git and it's functionalities and many of the team members have got a new favorite version control system.

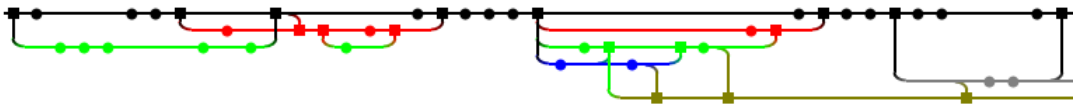


Figure 6.10: Graph of different branches evolving and merging back to master (Black)

6.7.6 Updated system architecture diagrams

An updated version of the class diagram in figure 6.11

An updated version of the entity relation diagram in figure 6.12

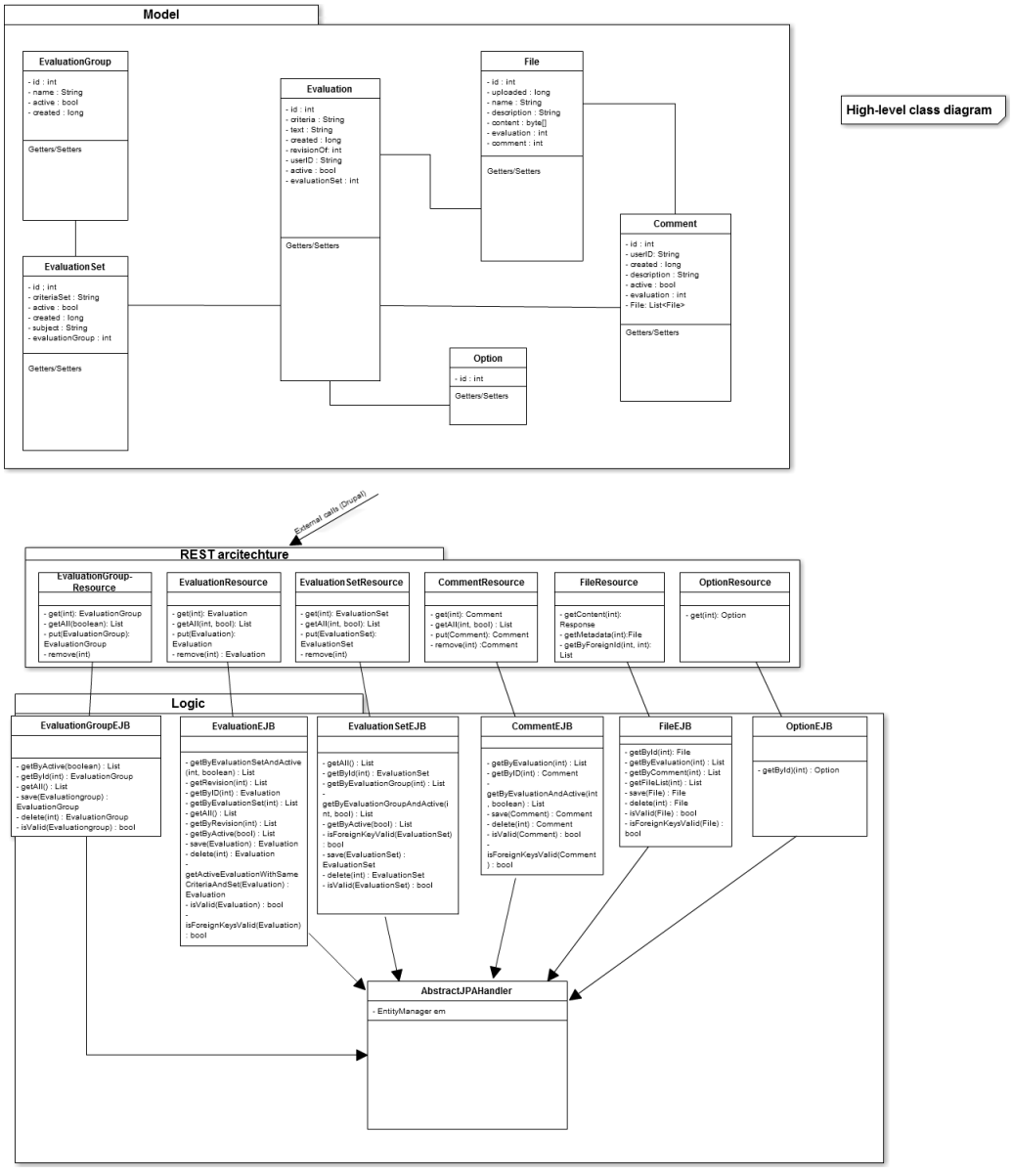


Figure 6.11: The final version of the class diagram

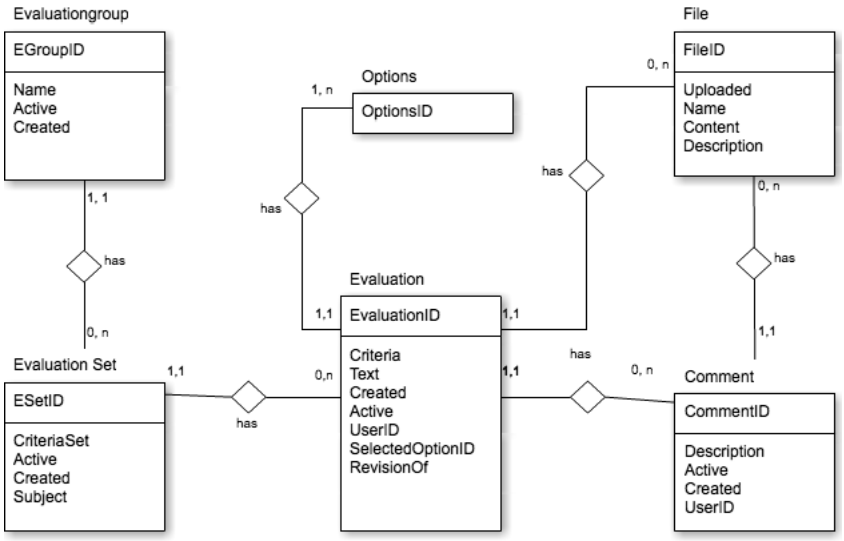


Figure 6.12: The final version of the Entity Relation diagram

Chapter 7

Closure

This chapter sums up the team's conclusions, lessons learnt and a discussion of the result.

7.1 Challenges summary

Drupal was the largest challenge during the project. The amount of work required to complete Drupal tasks was higher than expected. None of the team members had any experience with Drupal or with PHP, and the time needed to get the required knowledge was underrated by the team. This occasionally caused frustration. To fix this problem the entire team attended a course arranged by the student organization, Online. This course covered the basics of Drupal. The team was provided with a book about Drupal by the customer that was helpful [11]. Drupal remained a challenge, and as the development went on it was decided to delegate more people to work intensively with Drupal. At the end of the project, when the application was nearing completion, only the parts involving Drupal was lagging behind. In the last months of the project the team did some changes to the overall design of the Drupal part. Up until then the architectural plan had been complicated. After a meeting with the customer and the other Difi group, it was decided to go for an alternative solution.

7.2 Group structure

At the start of the project the team appointed different roles to different members. All of these roles has been executed as planned but some more than others. In meetings where everyone is present, the roles as

leader, Scrum master and secretary were working fine. However everyone cannot attend all meetings at all times, and in cases where different roles were missing other members would fill in. The same applies to the more programming relevant roles. This is a example of how agile development can be used to keep the project going, since all members share responsibility and knowledge of the progress at all time. This has at least given the different team members a taste of the different roles. In the end it was a good choice to appoint responsibility to different members. The impact has been positive since this helped to keep all aspects (customer, supervisor, testing, report and more) of the project up to date. The team made a good call as to who were suited for which roles, and no roles where switched between members during the development.

7.3 Risk management

One of the risks rated highest was "Wrong time estimate", and although the team worked hard to avoid this, it still occurred. Estimating is hard, and the only way to make it easier is with more experience and knowledge about the area that is to be estimated. Although this has been a small frustration for the team, it has not been too hard to get through. As planned in the risk analysis this was solved by working hard when a problem occurred and allocating tasks. The other high rated risk was the "Discovery of unknown tasks", and this has become more apparent later in the development. The consequences is that tasks that was estimated to require a given amount time suddenly is extended or postponed, which in turn means delays. "Loss of data", "Internal strife", "Miss the deadline" and "Prolonged sickness" are all risks that where completely avoided.

7.4 Customer relations

In the beginning of the project the communication with the customer happened regularly, mainly over email and Skype. This was vital in the starting phase since the team needed time to get to know the assignment. For the first three weeks during the planning phase the team required a continues communication with the customer to make sure the development was on the right track. During the project, the customer and the team had some communication problems. This was due to internal changes at the customer, which in turn meant that the customer contact had a lot of other important tasks to managed. Since the team and the customer contact had a common understanding, due to good communication the first weeks, the downtime in meetings did not effect the development as much as it would if it had occurred earlier in the development.

7.5 Supervisor interaction

The team has gotten important feedback and advises from the supervisor through regular meetings during the semester. Since the teamwork has gone smoothly it has mainly been the report the team needed and received help with. The supervisor assisted the team with the interaction and collaboration towards the other Difi group, and made sure the team followed the overall plan.

7.6 Development method

The team chose to use a customization of the Scrum method. The team had three stand up-meetings every week. Although this has gone well the team believe Scrum will work best if one has the opportunity to meet every day. The team has experienced that Scrum is well suited for smaller projects. Tracking progress and the possibility to monitor remaining work during the project were key factors of Scrum for the project. One of the challenges with Scrum was the estimation of the tickets, which was at times hard to do correctly.

7.7 Social relations

The team arranged meetings outside of the project, involving activities like Playstation tournaments and bowling. The decision to do this was made in order to become better acquainted and more confident at the workplace. This has certainly been successful, and although the team started out with a good work-relationship, this has evolved and become even better. These activities has helped to motivate and keep the moral high. The team had no real conflicts, only mild discussions on development issues. This is most likely due to the good starting point, the good working environment described above, and the effort made to have fun, as well as getting work done. For some meetings members would bring snacks like fruit and chocolate and the team arranged team-building exercises like pizza night, waffles and bowling with group 6. The teams opinion is that the collaboration agreement has worked as planned and made the group members more structured. Some of the group members has expressed their disapproval of the early meeting time, however they have fulfilled their obligations mentioned in the contract.

7.8 Further development

The first and necessary part is merging the project with the other Difi group, group 6. The customer will use their own CSS style schemas and fitting the criteria sets, criteria and options attributes from group 6. There is a bit of work, but the groups have collaborated to some extent to make the overall structure and conventions more similar. The customer will then possess a fully working system for evaluating public websites.

Further on, there are a lot of possibilities for expanding the application. With small modifications the system can be used to evaluate other self-imposed type of websites. There may of course be some modifications of the fields and changing of the criteria sets and criteria's. The file and comment functionality is separated from the rest of the functionality, to make it easy to implement comment and file functionality wherever it is wanted. An example is the possibility to implement the functionality of adding a comment on an evaluation set, adding a file to an evaluation group, and so on. With some larger changes the system may be used to evaluate other items, not necessary websites. Examples are programs, apps and documents.

7.9 Conclusion

The whole project has been a huge learning experience for all of the members in the team. Every member has worked hard and learned a lot of new technology, with specific knowledge in Java EE6, JPA 2.0, RESTful web services and Drupal. In addition to the new technical knowledge the project has been a real-life example of how to use Scrum as a development method and how to organize a project of this size.

In the beginning of the project, the team wanted to rotate the group members on the different technologies, but as the project evolved the team experienced that this was hard to accomplish. It was inefficient and since there always were two experts on each technology.

The team is very pleased with the result. All the requirements from the customer are ticked and done in a way that every member can be proud of what they have produced. It has been a lot of work, and of course there have been some complex tasks which have reduced the efficiency in periods, but the overall throughput has been good. The members have motivated each other and the social gatherings have exclusively been a positive effect on the cooperation between members.

7.9.1 Evaluation from the customer

After approval from the customer the team has translated the customer's evaluation of the project. The original can be found in Appendix H.

Difi, earlier Norge.no, has for years held "Kvalitetskonferansen" (Conference of Quality), a conference where the best websites of the public sector gets awarded. This is a visible tool that helps to improve websites and services of the public sector, and one sees clearly how the criteria sets developed by Difi has implications for development and acquisition of public sites and services.

Through all the years, Difi has used an internal task system called "KVRS", and even though the current tool still works okey, we see the advantage in a new tool making the job easier. We saw this as a great project for students taking this subject and divided the system into two separate tasks: One for criteria set and one for evaluation. This group has implemented the system for Evaluation, a system that is required to communicate both via REST-full web-services for future fully automatically testing on some criterias, but also be easy to use for the many consultants we hire to perform the evaluations. We also see a huge potential for the product in connection with establishing a competence center for universal design inside Difi.

Demonstration and code review show that we have got delivered a product that is almost ready for release. The product meets all our needs. It has sometimes been difficult to find time to follow up the project, but since requirements have been unchanged from day one, it does not appear to us that the product has been affected by the lack of time.

The time for activating the product has not been determined yet, but it expects to be in use well before the evaluations will be carried out in the spring. (The quality conference has got a new cycle his year, and next time will be spring 2013.) We are very satisfied with the product and the product owners are looking forward to starting using the program.

Bibliography

[1] Andrew Burgess, *Getting good with git*. Rockable press, 2010.

[2] Antonio Goncalves, *Beginning Java^{TM} EE 6 Platform with GlassFish^{TM} 3: From Novice to Professional*. Apress Inc, 2009.

[3] Barry W. Boehm, A spiral model of software development and enhancement, *http://dl.acm.org/citation.cfm?doid=12944.12948*. August 1986.

[4] Don Wells, Extreme Programming, A gentle introduction, *http://www.extremeprogramming.org/*. September 2009.

[5] Dr. Winston W. Royce, Managing the development of large software systems, *http://www.cs.umd.edu/class/ spring2003/cmsc838p/Process/waterfall.pdf*. 1970.

[6] Drupal, *www.drupal.org*. retrieved in February 2012.

[7] Extreme Programming: A gentle introduction *http://www.extremeprogramming.org/* retrieved in April 2012.

[8] Karl Scotland, Aspects of Kanban, *http://agilemanagement.net/index.php/Blog/ the_principles_of_the_kanban_method/*. Summer 2010.

[9] Ken Schwaber and Jeff Sutherland, The Scrum Guide, *http://www.scrum.org/storage/scrumguides/ Scrum_Guide.pdf*. October 2011.

[10] Java XML & JSON binding, *http://blog.bdoughan.com/*. retrieved in February 2012.

[11] Todd Tomlinson and John VanDyk , *Pro Drupal 7 Development*. 3rd Edition, Apress Inc, 2010.

[12] Waterfall Model, ADVANTAGES, EXAMPLES, PHASES AND MORE ABOUT SOFTWARE DEVELOPMENT *http://www.waterfallmodel.com/* retrieved in April 2012.

Appendix A

Glossary

Some words used in the report may need some more explanation.

- Difi - Agency of Public Management and eGovernment, has the task to develop and renew the public sector.
- Evaluation - An evaluation as what is produced when an evaluator answers a criteria. This consists of a chosen alternative and a explanation why this is alternative selected.
- Evaluation set - This is a collection of evaluations. This will often be for a given website. For example, a website will be considered at the three main topics "Availability", "Customization" and "Useful content", which again contains 10 to 12 evaluations.
- Evaluation group - This is used to group sites/evaluationsets. An example of a evaluation group may be "Best pages 2012".
- Group 6 - This is one of the other groups in the course IT2901. They chose to work with the other half of the system this project applies to.
- PHP - This is a general-purpose server-side scripting language.
- Revision - Another version of the same evaluation.
- Skype - This is a proprietary voice-over-Internet Protocol service and software application, used by the team for most distance communication.
- Ticket - A description of one work task, used in Scrum (further explanation at page 13).

Appendix B

Agreement

Cooperation agreement for group 8, Project II

Attitudes

- Take responsibility for your own actions.
- Constructive criticism should always be encouraged. Be open for constructive criticism.
- Give compliments; build up each other's confidence.
- Ask if you have any questions. There are no stupid questions.
- Everybody has responsibility for their own, and the groups' activity.
- Avoid alliances.

Meeting time, absence and duties

- Working hours are standardized from 10.15-11.00 Monday, Wednesday and Friday, and after that the members of the group can work wherever they want. All the group members must be ready for stand-up 10.30.
- In the beginning of each sprint, 4 hours are set of for sprint planning.
- Everybody should notify the group in advance about absence, and if they are late. A buffer on possibly 5 minuets can be accepted.

- In case of absence, the absent must make sure to catch up on work and decisions that are made. But at the same time, the rest of the group should also inform the absent about important information and events.
- If the agreement/meeting points are broken the guilty group member should be punished. The person may choose to buy a snack to bring to the meeting (equivalent to the cost of a 6-pack) – which must be brought within the current sprint, or a round of beer (6-pack) that the group member must bring to team building.

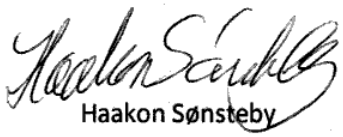
Work rules

- All group members shall be included in joint decisions. Those who are not present when a decision is made will be able to express their opinion during the next group meeting.
- Group log/minutes from meetings with the customer, supervisor and scrum meetings, should be prepared before the next meeting.
- The group must plan a team building/socialization night to strengthen the team spirit. Preferably once a month.
- The members of the group must present any problems regardless of the significance.
- The group members should not exclude any problems.
- Group relevant conflicts should solved within the group.

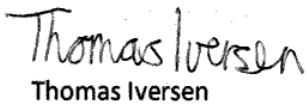
Revisjon 01

Dato: 27.02.12 Sted: Trondheim

Underskrift

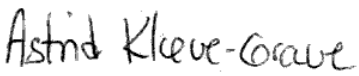

Haakon Sønsteby


Daniel Tandberg Abrahamsen


Thomas Iversen


Mathilde Ødegård Oftedal


Anders Palfi


Astrid Kløve-Graue

Appendix C

Minutes with customer

As an example of minutes with the customer, the team have selected two minutes; The first meeting, at 30.01.2012, and the demo when the customer was in Trondheim 02.03.2012.

C.1 Meeting with the customer, 02.03.12

- Agenda

  Issue 1. A status update wanted both by the customer and the group.

  Issue 2. Get clarification around how our group can work with the other group.

  Issue 3. The group got some technical questions concerning drupal.

- Summery

  Issue 1.

  - The customer gave us an update about their situation; Because of internal reorganization at Difi, the project has been given a low priority. The group got promised that the situation will be better soon.
  - The customer wants us to send him a reminder mail where we tell him our meeting times. Remember this at the meeting 5. March!
  - We have a working application with the database at the base level, and Drupal on the top level. We manage to retrieve, add, and delete from Drupal - and at the same time update the database.

- We are yet to complete the first module in Drupal. When this is done we can use it as a template for the remaining modules. There is a lot of work remaining in this area.
- We are missing all the the classes in REST, except EvaluationGroup. We have completed the EvaluationGroup class in our REST-service, but the other classes are not finished yet.
- We are yet to complete the connection between Drupal and REST so that the resource classes can be used by other methods as well.

Issue 2.

- Both the customer and the group want to work together with the other group (group 6) on some of the project. The customer is going to talk to the course manager about this. He also explains the connection between our groups. He draws both groups ER-diagrams, and explaining the context.

Issue 3.

- The customer gave us some tips on how we could use drupal:
  * Best practice is to create the code from scratch.
  * We have to decide, in cooperation with the other group, how we want to use drupal; A content page with all information, or basic pages where we have to handle the views etc. ourselves.

C.2 Skype meeting with the customer, 30.01.12

- Agenda

  Issue 1. The group got questions regarding the use cases.

  Issue 2. The group got questions regarding the domain model.

  Issue 3. The group got questions regarding the usage of Virtual Box.

  Issue 4. The group and the customer got to agree on what kind of documentation that should be delivered along with the product.

- Summery

  Issue 1.

  - Should a logged in user see every evaluation with content?
    - Everybody should see everything.
    - The group should not use a user table, but an identifier (String) in the database to keep track of users.
    - The REST-service should not not have any form of user control.
  - Criteriaset; Should this be generic for each page to be evaluated, or should the criteria set limit which fields are visible?
    - Criteriaset is just an identifier and should not limit the evaluation set.
    - There should be an evaluation set for every page evaluated.
  - Who have access to create and update evaluation groups? Only administrator, or also an evaluator?
    - The product should only have one user who is able to do everything, a role (administrator) that has every right.

  Issue 2.

  - Can evaluation set consist of several evaluations from different web pages?
    - The evaluation consists of the options that are selected. The evaluation must be saved. One evaluation set contains evaluations only for one web page.
  - The customer explains the domain model in more detail
    - Files can be linked to a comment or to an evaluation. It is desirable to see which revision the file was uploaded to.

    * If the group needs any examples of how evaluation groups, evaluation sets and evaluations should look like or how they are connected, there are plenty of examples at the website kvalitet.difi.no, their previous webpages for the project we are developing.

– What is the option entity for?

    * This not apart of this groups product, but an example is;

        Case: find contact information on the page:

        Opt1 – at the bottom of the page, 2 points.

        Opt2 – on a separate page, 1 point.

        Not possible to find, 0 point.

    * It is Difi's job to integrate, not part of the project.

Issue 3.

– Why should the group use Virtual Box? What are the pros and cons?

    * With Virtual Box you do not have to mess up the operating system on your computer.
    * The environment is the same across the computers.
    * Need only to test against Ubuntu, as this is the operative system Difi uses.
    * The operative environment is similar to Difis environment. This simplifies the workspace.

Issue 4

– Only minor documentation is required, but the customer would like to have;

    * How to install glassfish
    * How to install the .jar files

Appendix D

Minutes with supervisor

We have selected the two most important minutes from the meeting with the supervisor:

D.1 Meeting with the supervisor 26.03.12

- Agenda

  Issue 1. The supervisor is going to give feedback on the report delivered 9th of March.

  Issue 2. Talk about the possibility of cooperation with the other group that develop to the other part of the product.

  Issue 3. General status update.

- Summery

  Issue 1.

  - The report needs more structure
    - Minutes of meetings must be in the report.
      - The supervisor want to see how we cooperate with the customer.
    - The supervisor wants us to send a weekly status report to him every Monday. By doing this the supervisor will get a better insight of the group's progression.

  Issue 2.

- The superivsor said it is important that the group meet the requirements of the course. As long as the group do this it is no problem to work together with group 6 on some parts of the product. It is important that we distinguish between what have been done in cooperation with the other group.

  Issue 3.

- The supervisor asked some questions;
  - How are the progress?
    - Still a lot to do, but now the groups main fucos is on drupal.
  - Do the group have any internal deadlines?
    - The group have come to an agreement to get the product done in the beginning of May, aswell as some minor deadlines when it comes to Drupal.
  - How are the communication with the customer?
    - The customer is very busy at the moment, but the group wants to have weekly meetings with the customer.
- A new meeting with the supervisor is scheduled for 13th of April. The supervisor wants a new version of the report on the 11th of April.

D.2 Meeting with the supervisor 30.01.12

- Agenda

  Startup meeting with the supervisor.

- Summery
  - The groups supervisor is Anh Nguyen Duc.
  - The group and the supervisor want to meet every other week. Anh prefers Fridays at 11.00.
  - The group presents each member to the supervisor, and explains what the group think of the task.
  - The supervisor explains the group it is not just the project that counts on the grade it is the process as well!
  - The supervisor tells the group that the group need to make a plan. If the project is to large we have to notify the customer as early as possible. It is important that the group prioritize requirements.

Appendix E

Extended textual use cases

Use case ID	**UC-1.1**
Scope	Evaluation group
Description	List all active/inactive evaluation groups
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	None
Steps	1. Click on the evaluation group link 2. Click on the "Show all all evaluation groups" button 3. Click on the "Show deleted" button
Variations	None

Table E.1: Use case, UC-1.1

Use case ID	**UC-1.2**
Scope	Evaluation group
Description	Create evaluation group
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	None
Steps	1. Click on "Create new evaluation group" 2. Fill in the required fields 3. Click save
Variations	None

Table E.2: Use case, UC-1.2

Use case ID	**UC-1.3**
Scope	Evaluation group
Description	Update evaluation group
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exists an evaluation group
Steps	1. Click on the evaluation group to change 2. Click edit 3. Fill in the required fields 4. Click save
Variations	(a) If no evaluation group exists, create one and do step 1 again

Table E.3: Use case, UC-1.3

Use case ID	**UC-1.4**
Scope	Evaluation group
Description	Set an evaluation group as inactive
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group
Steps	1. Select the wanted evaluation group 2. Click edit 3. Uncheck the active checkbox 4. Click save
Variations	(a) If no evaluation group exists, create one and do step 1 again

Table E.4: Use case, UC-1.4

Use case ID	**UC-1.5**
Scope	Evaluation set
Description	List all existing evaluation set
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group
Steps	1. Click the wanted group
Variations	(a) If no evaluation group exists, create one and do step 1 again

Table E.5: Use case, UC-1.5

Use case ID	**UC-1.6**
Scope	Evaluation set
Description	Create evaluation set
Preconditions	1. A browser capable of opening Drupal sites installed. 2. Server running
Assumptions	1. There exist an evaluation group
Steps	1. Click on the wanted group 2. Click on new evaluation set 3. Fill in the required fields 4. Click save
Variations	(a) If no evaluation group exists, create one and do step 1 again

Table E.6: Use case, UC-1.6

Use case ID	**UC-1.7**
Scope	Evaluation set
Description	Update evaluation set
Preconditions	1. A browser capable of opening Drupal sites installed. 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set
Steps	1. Select the evaluation set to change 2. Click edit 3. Fill in the required fields 4. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set and do step 1 again (b) If no evaluation set exists, create one and do step 1 again

Table E.7: Use case, UC-1.7

Use case ID	**UC-1.8**
Scope	Evaluation set
Description	Set an evaluation set as inactive
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set
Steps	1. Select the evaluation set 2. Click edit 3. Uncheck the active checkbox 4. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set and do step 1 again (b) If no evaluation set exists, create one and do step 1 again

Table E.8: Use case, UC-1.8

Use case ID	**UC-1.9**
Scope	Evaluation set and evaluation
Description	List all active/inactive evaluations in evaluation set
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set
Steps	1. Click on the wanted evaluation group 2. Click on the wanted evaluation set. 3. Click on "Show deleted"
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set and do step 1 again (b) If no evaluation set exists, create one and do step 1 again

Table E.9: Use case, UC-1.9

Use case ID	**UC-1.10**
Scope	Evaluation
Description	List all revisions of evaluations
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Click on the evaluation link 2. Select an evaluation
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.10: Use case, UC-1.10

Use case ID	**UC-1.11**
Scope	Evaluation
Description	Get revision of evaluation
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Click on the evaluation link 2. Click on an evaluation 3. Click on the wanted evaluation from the list
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.11: Use case, UC-1.11

Use case ID	**UC-1.12**
Scope	Evaluation set and evaluation
Description	Create evaluation in evaluation set
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set
Steps	1. Select wanted set 2. Click on "New evaluation" 3. Fill in the required fields 4. Select the evaluation set from a list 5. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set and do step 1 again (b) If no evaluation set exists, create one and do step 1 again

Table E.12: Use case, UC-1.12

Use case ID	**UC-1.13**
Scope	Evaluation
Description	Create new revision of evaluation
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select the wanted evaluation 2. Click on "Edit" 3. Change the some data 4. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again. (b) If no evaluation set exists, create one with and evaluation. Then do step 1 agai (c) If no evaluation exists, create one and do step 1 again

Table E.13: Use case, UC-1.13

Use case ID	**UC-1.14**
Scope	Evaluation
Description	Mark evaluation deleted
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select the wanted evaluation 2. Click delete
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.14: Use case, UC-1.14

Use case ID	**UC-1.16**
Scope	Evaluation and comment
Description	List all comments on evaluation
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select an evaluation
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.15: Use case, UC-1.16

Use case ID	**UC-1.17**
Scope	Evaluation and comment
Description	List all comments on revision of evaluation
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select the wanted evaluation 2. Select the wanted revision
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.16: Use case, UC-1.17

Use case ID	**UC-1.18**
Scope	Evaluation and comment
Description	Add comments on revision of evaluation
Preconditions	1. A browser capable of opening Drupal sites installed 2. Server running
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select the wanted evaluation 2. Select the wanted revision 3. Fill in the required fields in the comment section 4. Click on add comment
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.17: Use case, UC-1.18

Use case ID	**UC-1.19**
Scope	Comment
Description **Preconditions**	Mark comment deleted 1. UC-1.11 2. An existing comment on the evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation 4. There exist a comment
Steps	1. Select the wanted comment 2. Click on "Delete comment"
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and then create a comment. Do step 1 again (b) If no evaluation set exists, create one with and evaluation and a comment. Then do step 1 again (c) If no evaluation exists, create one and a comment. Do step 1 again (d) If no comment exists, create one and do step 1 again

Table E.18: Use case, UC-1.19

Use case ID	**UC-1.20**
Scope	Comment and file
Description	Add file(s) on comment
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation 4. There exist a comment
Steps	1. Fill in the required fields in the comment section 2. Click add file in comment section 3. Select a file from your device 4. Jump to step 2 as many times as wanted 5. Click add comment
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and then create a comment. Do step 1 again (b) If no evaluation set exists, create one with and evaluation and a comment. Then do step 1 again (c) If no evaluation exists, create one and a comment. Do step 1 again (d) If no comment exists, create one and do step 1 again

Table E.19: Use case, UC-1.20

Use case ID	**UC-1.22**
Scope	Evaluation and file
Description	List all files on comment
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation 4. There exist a comment
Steps	1. Select the wanted evaluation
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and then create a comment. Do step 1 again (b) If no evaluation set exists, create one with and evaluation and a comment. Then do step 1 again (c) If no evaluation exists, create one and a comment. Do step 1 again (d) If no comment exists, create one and do step 1 again

Table E.20: Use case, UC-1.22

Use case ID	**UC-1.23**
Scope	Evaluation and file
Description	List all files on evaluation
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select the wanted evaluation
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.21: Use case, UC-1.23

Use case ID	**UC-1.24**
Scope	Evaluation and file
Description	List all files on revision of evaluation
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select the wanted evaluation 2. Select the wanted revision
Variations	(a) If no evaluation group exists, create one. Then create an evaluation set with and an evaluation, and do step 1 again (b) If no evaluation set exists, create one with and evaluation. Then do step 1 again (c) If no evaluation exists, create one and do step 1 again

Table E.22: Use case, UC-1.24

Appendix F

Extended test results

Test F.1, F.2, F.3 and F.4 show the tests done on the evaluation groups.

Test ID	ST-01
Test name	Evaluation group 1
Purpose	To test if the requirement FR-1.1, List all active/inactive evaluation groups, is met through the Drupal site
Requirements	Running system
Test description	1. Click on "Show all evaluation groups" All the active evaluation groups appear 2. Click on "Show deleted" All the deleted (inactive) evaluation groups appear
Test result	PASSED

Table F.1: Evaluation group test, show all evaluation groups

Test ID	ST-02
Test name	Evaluation group 2
Purpose	To test if the requirement FR-1.2, Create evaluation group, is met through the Drupal site
Requirements	Running system
Test description	1. Click on "Show all evaluation groups" 2. Click on "Add new evaluation group" 3. Write the name of the new evaluation group 4. Click on "Add" A note confirms that the evaluation group is added, and the details about the evaluation group appear
Test result	PASSED

Table F.2: Evaluation group test, create evaluation group

Test ID	ST-03
Test name	Evaluation group 3
Purpose	To test if the requirement FR-1.3, Update evaluation group, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Choose the evaluation group to be edited 3. Click on "Edit" 4. Edit the name of the evaluation group A note confirms that the evaluation group is updated
Test result	PASSED

Table F.3: Evaluation group test, edit evaluation group

Test ID	ST-04
Test name	Evaluation group 4
Purpose	To test if the requirement FR-1.4, Set evaluation group inactive, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Choose the evaluation group to be deleted 3. Click on "Delete" A note confirms that the evaluation group is deleted
Test result	PASSED

Table F.4: Evaluation group test, delete evaluation group

Test F.5, F.6 and F.7 show the tests done on the evaluation set.

Test ID	ST-05
Test name	Evaluation set 1
Purpose	To test if the requirement FR-1.6, Create evaluation set, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Click on the evaluation group the evaluation set should belong to 3. Click on "Add new evaluation set" 4. Enter the subject of the evaluation set 5. Choose a criteria set 6. Click on "Save" A note confirms that the evaluation set is added, and the details about the evaluation set appear
Test result	PASSED

Table F.5: Evaluation set test, create evaluation set

Test ID	ST-06
Test name	Evaluation set 2
Purpose	To test if the requirement FR-1.7, Update evaluation set, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Click on the evaluation group the evaluation set belongs to 3. Choose the evaluation set to be edited 4. Click on "Edit" 5. Edit the name of the evaluation set 6. Click on "Save" A note confirms that the evaluation set is updated, and the details about the evaluation set appear
Test result	PASSED

Table F.6: Evaluation set test, edit evaluation set

Test ID	ST-07
Test name	Evaluation set 3
Purpose	To test if the requirement FR-1.8, Set evaluation set inactive, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group
Test description	1. Click on "Show all evaluation groups" 2. Click on the evaluation group the evaluation set belongs to 3. Choose the evaluation set to be deleted 4. Click on "Delete" A note confirms that the evaluation set is deleted
Test result	PASSED

Table F.7: Evaluation set test, delete evaluation set

Test F.8, F.9, F.10 and F.11 show the tests done on comment.

Test ID	ST-14
Test name	Comment 1
Purpose	To test if the requirement FR-1.16, List all comments on evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation A list of the comments on the evaluation appear
Test result	PASSED

Table F.8: Evaluation test, list all comments on evaluation

Test ID	ST-15
Test name	Comment 2
Purpose	To test if the requirement FR-1.17, List all comments on revision of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on the desired revision of evaluation A list of the comments on the evaluation appear
Test result	PASSED

Table F.9: Evaluation test, list all comments on revision of evaluation

Test ID	ST-16
Test name	Comment 3
Purpose	To test if the requirement FR-1.18, Add a comment on revision of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Add a comment on the revision of the evaluation 6. Click on "Save" A note confirms that the comment is added
Test result	PASSED

Table F.10: Evaluation test, add a comment on revision of evaluation

Test ID	ST-17
Test name	Comment 4
Purpose	To test if the requirement FR-1.19, Mark comment deleted, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation 5. There exist a comment
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on the desired comment 6. Click on "Delete" A note confirms that the comment is deleted
Test result	PASSED

Table F.11: Evaluation test, mark comment deleted

Test F.12, F.13, F.14, F.15 and F.16 show the tests done on adding files.

Test ID	ST-18
Test name	File 1
Purpose	To test if the requirement FR-1.20, Add file(s) on comment, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation 5. There exist a comment
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Add a comment 6. Click on "+ Add files" It is possible to attach several files, by clicking on "+ Add files" again 7. Choose the desired file and add a description 8. Click on "Save" A note confirms that the comment, and the file is added. And the information about the file appears.
Test result	PASSED

Table F.12: Evaluation test, add file(s) on comment

Test ID	ST-19
Test name	File 2
Purpose	To test if the requirement FR-1.21, Add file(s) on revision of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on "Edit" 6. Click on "+ Add files" It is possible to attach several files, by clicking on "+ Add files" again 7. Choose the desired file and add a description 8. Click on "Save" A note confirms the file is added. The information about the file appears.
Test result	PASSED

Table F.13: Evaluation test, add file(s) on revision of evaluation

Test ID	ST-20
Test name	File 3
Purpose	To test if the requirement FR-1.22, List all files on comment, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation 5. There exist a comment
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on the desired comment The file(s) attached the comment is listed
Test result	PASSED

Table F.14: Evaluation test, list all files on comment

Test ID	ST-21
Test name	File 4
Purpose	To test if the requirement FR-1.23, List all files on evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation The file(s) attached the evaluation is listed
Test result	PASSED

Table F.15: Evaluation test, list all files on evaluation

Test ID	ST-22
Test name	File 5
Purpose	To test if the requirement FR-1.24, List all files on revision of evaluation, is met through the Drupal site
Requirements	1. Running system 2. There exist an evaluation set 3. There exist an evaluation group 4. There exist an evaluation
Test description	1. Click on "Show all evaluation groups" 2. Click on the desired evaluation group 3. Click on the desired evaluation set 4. Click on the desired evaluation 5. Click on the desired revision of the evaluation The file(s) attached to the revision of the evaluation is listed
Test result	PASSED

Table F.16: Evaluation test, list all files on revision of evaluation

Appendix G

User guides

G.1 Drupal

This guide consists of screenshots of each page on the Drupal website. Each screenshot got comments explaining what all the buttons and forms do.

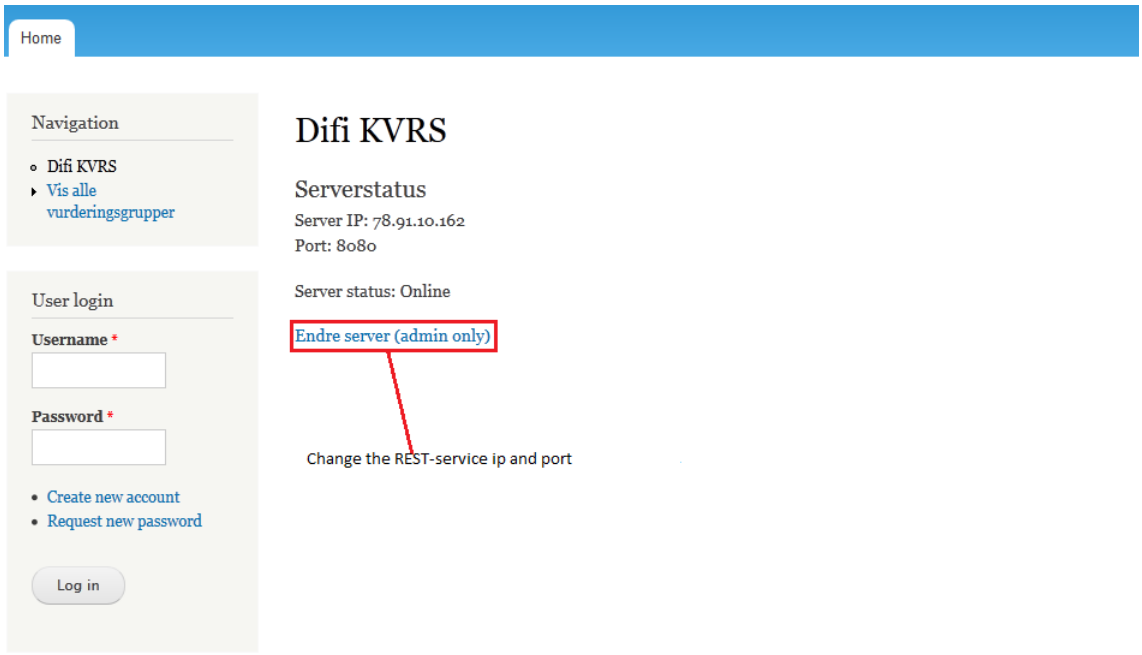


Figure G.1: Page showing server status

Figure G.2: Page showing all evaluationgroups

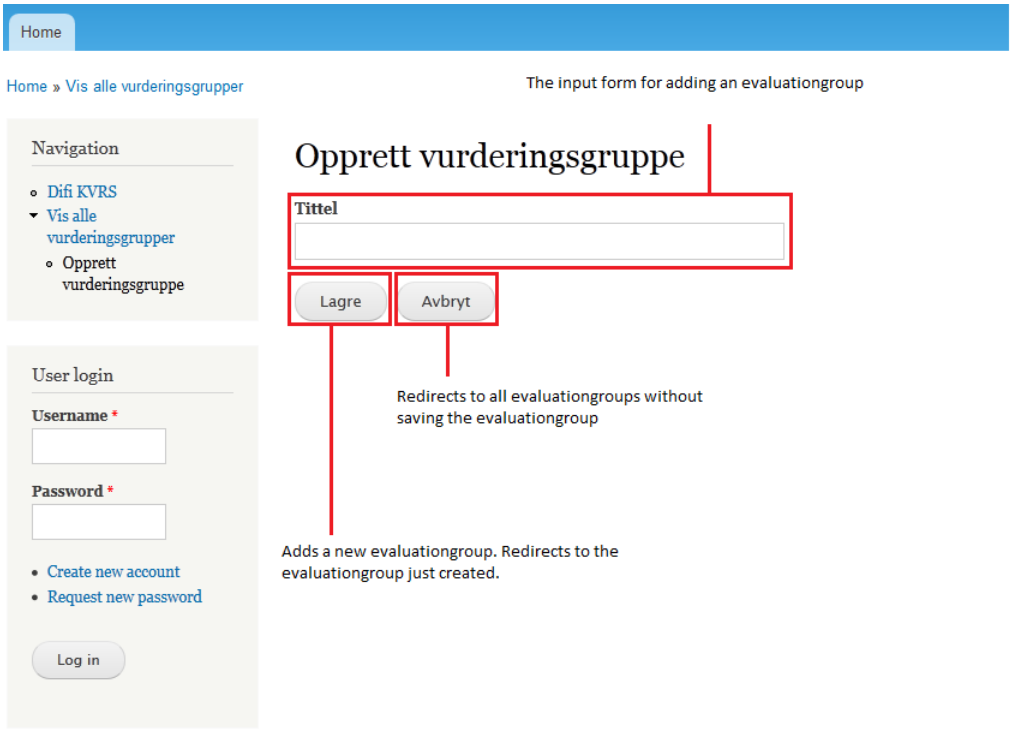


Figure G.3: Page for adding an evaluationgroup

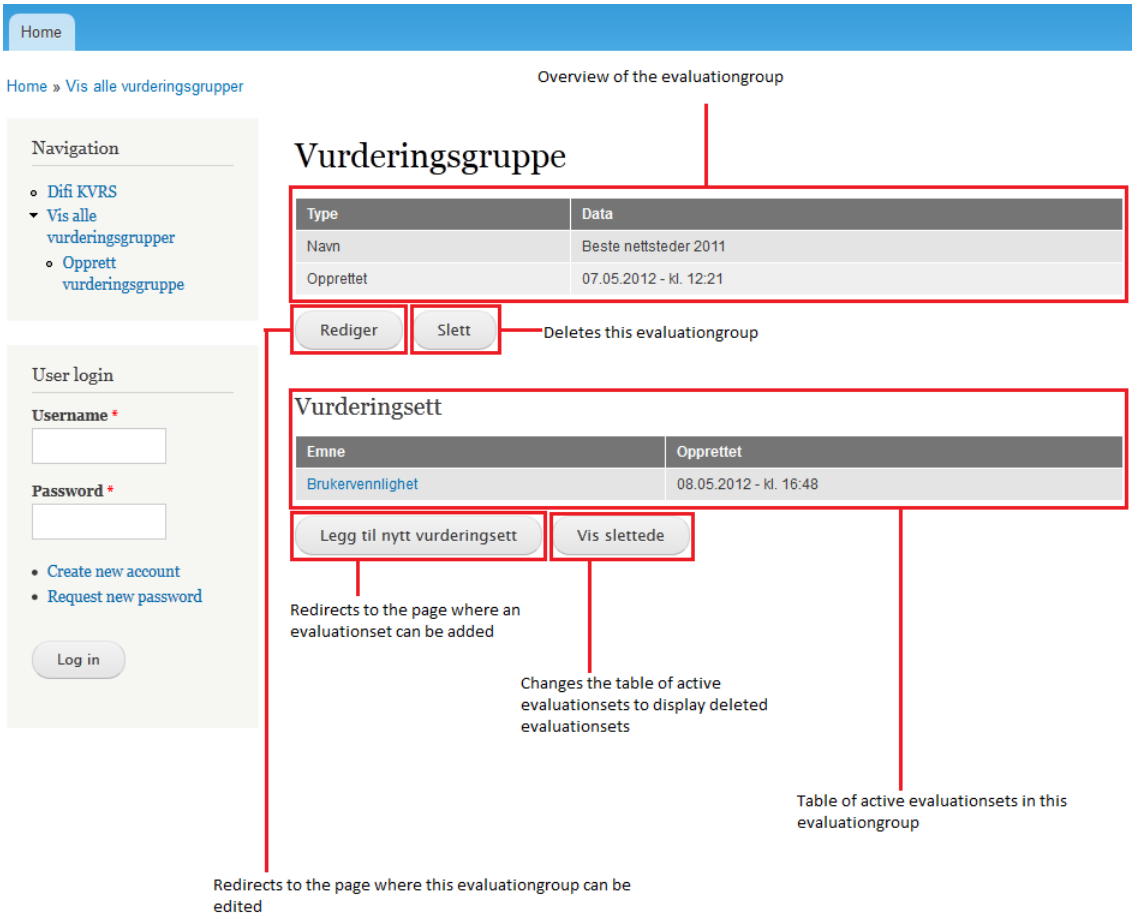


Figure G.4: Page for viewing an evaluationgroup

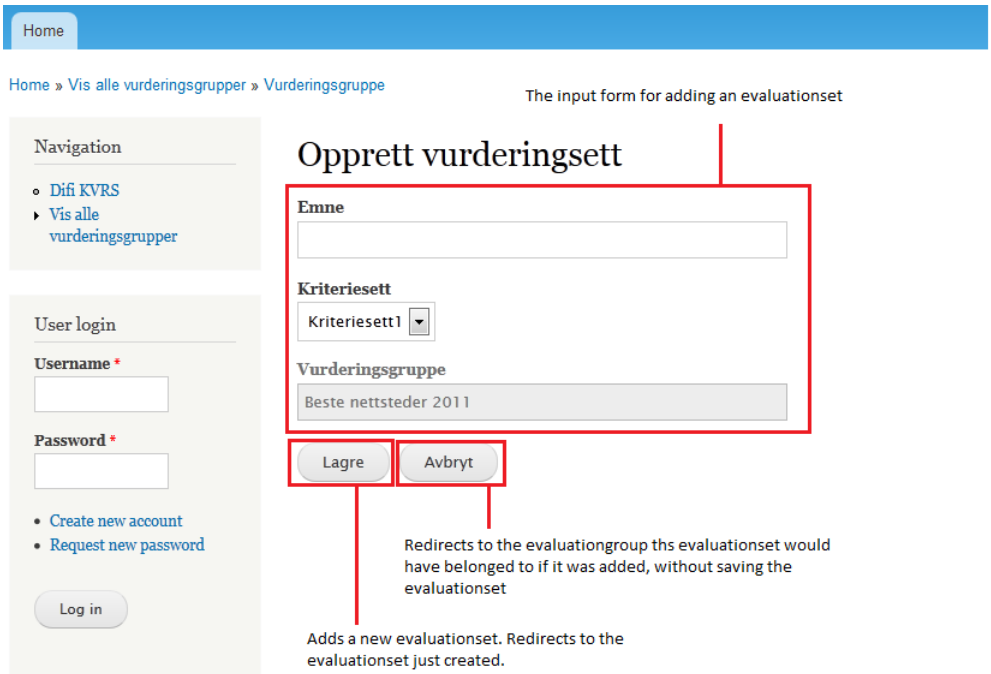


Figure G.5: Page for adding an evaluationset

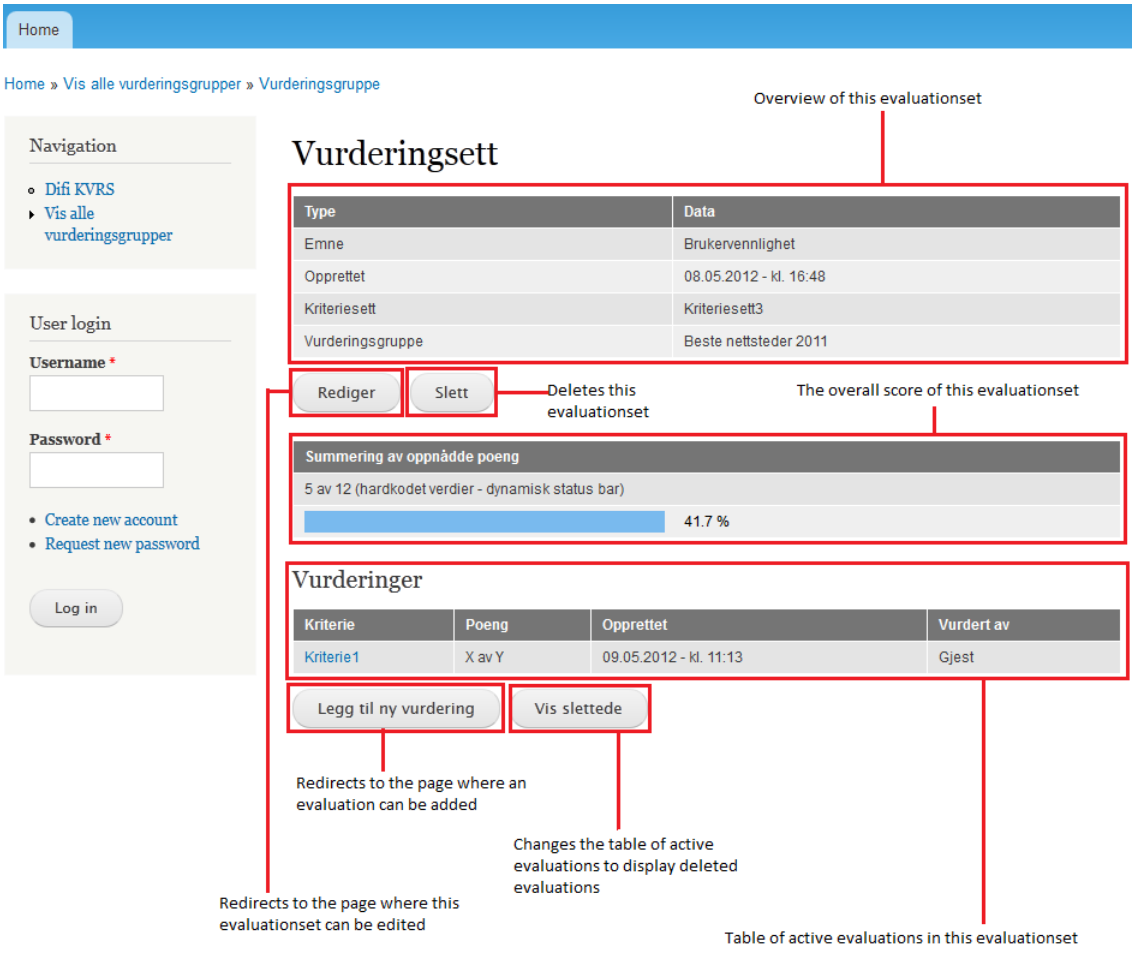


Figure G.6: Page for viewing an evaluationset

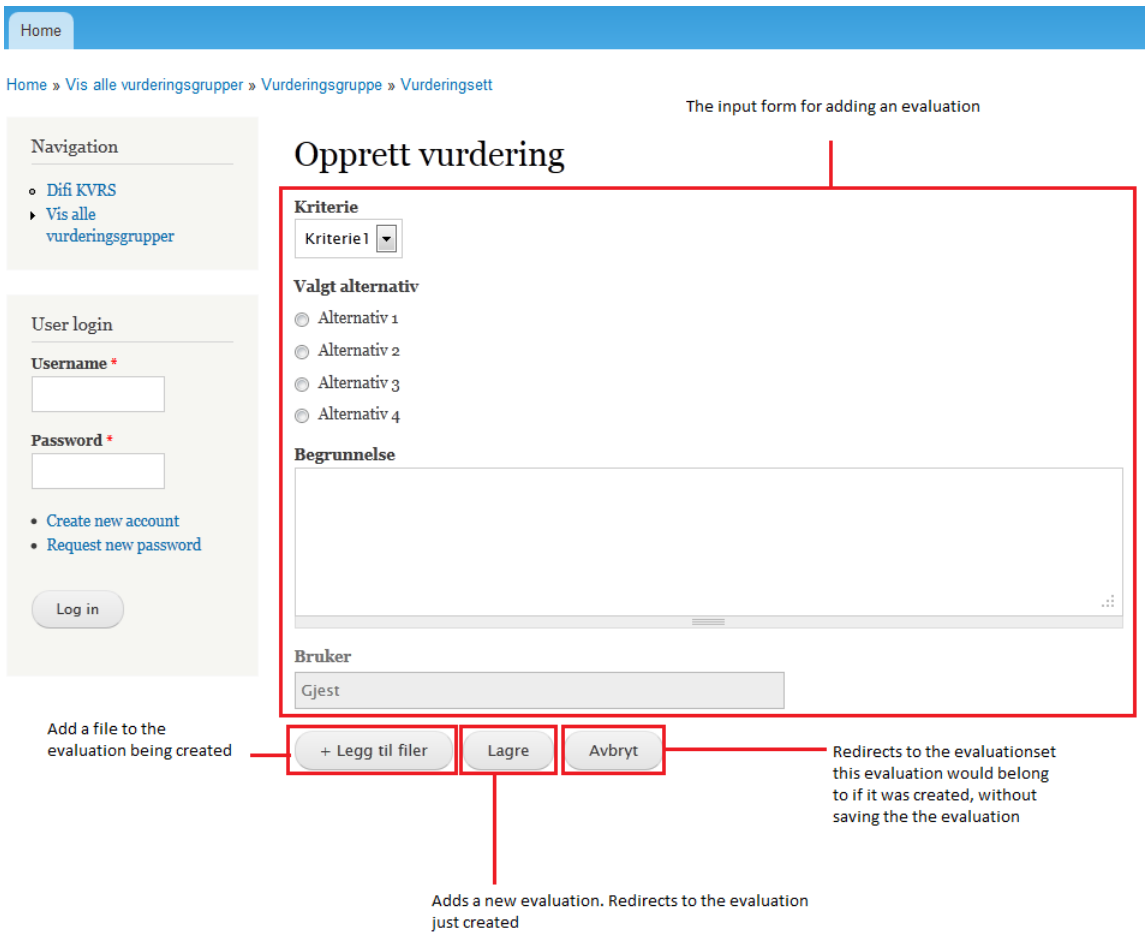


Figure G.7: Page for adding an evaluation

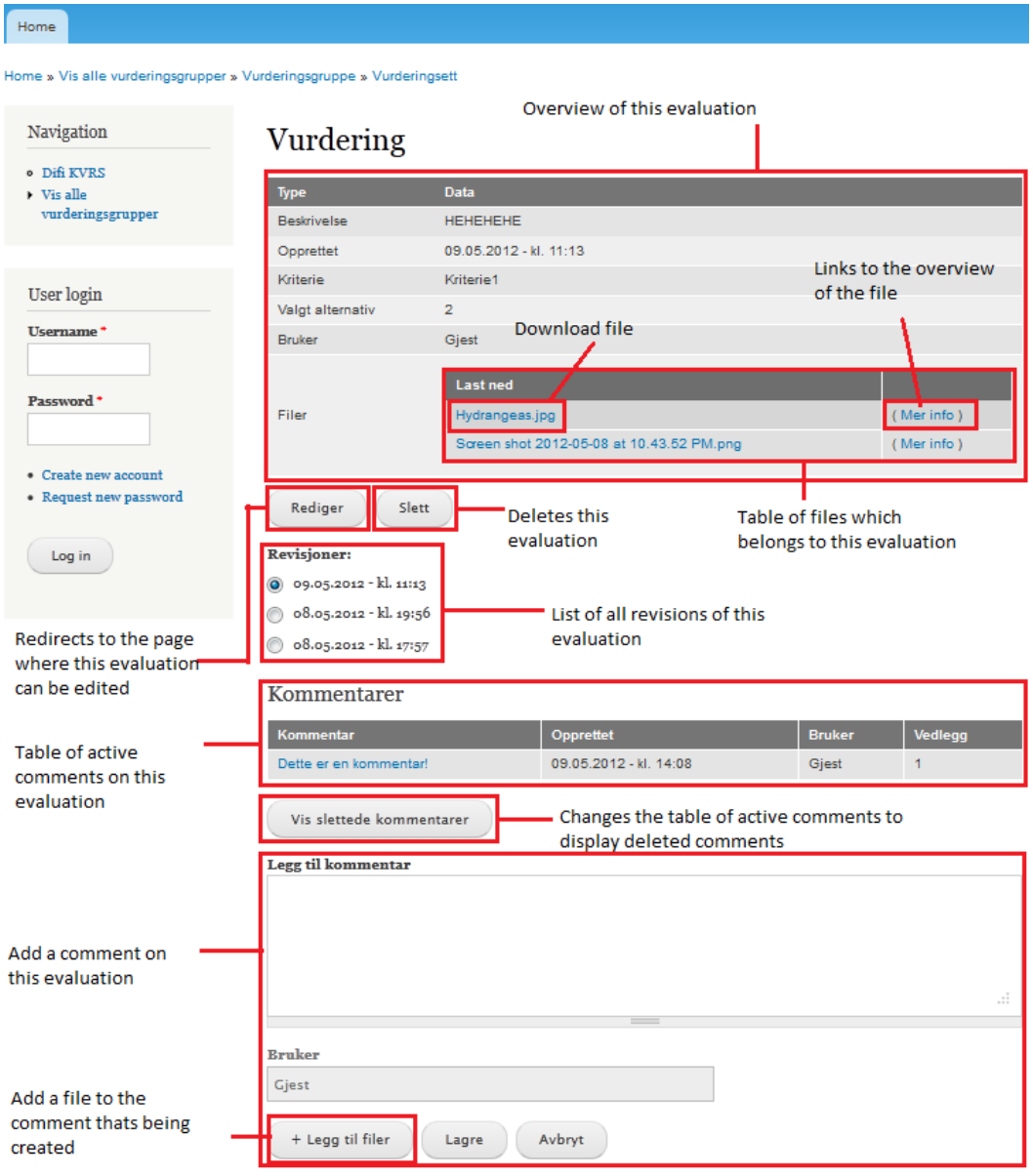


Figure G.8: Page for viewing an evaluation and adding a comment

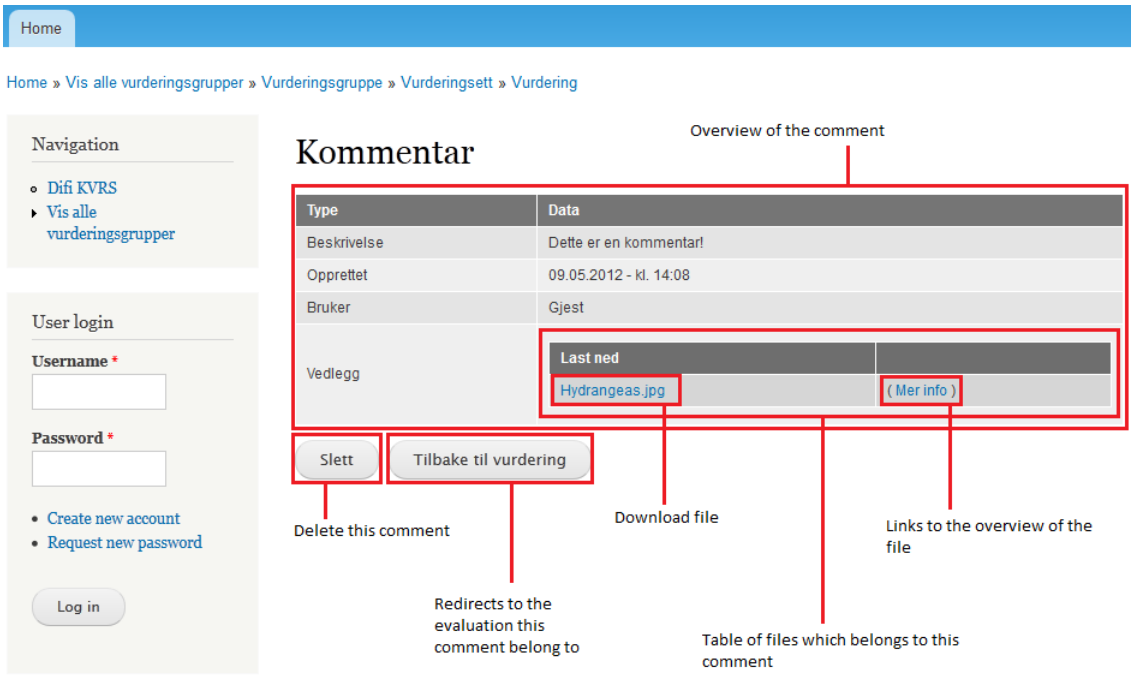


Figure G.9: Page for viewing a comment

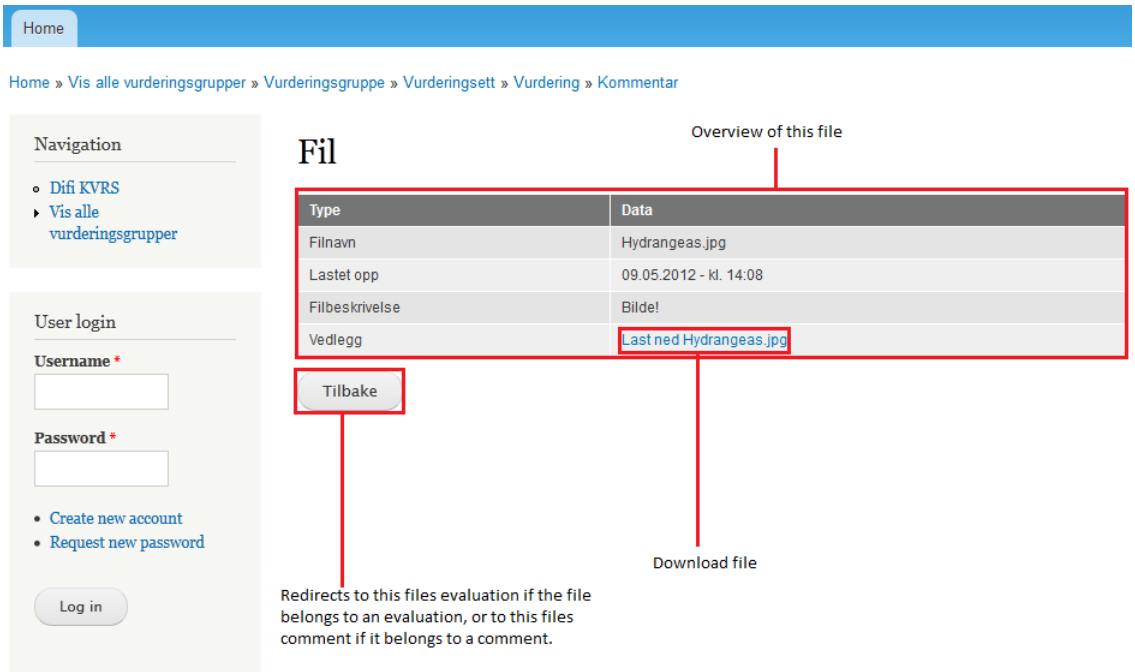


Figure G.10: Page for viewing a file

G.2 REST-service

This guide gives an overview of all the valid queries which the REST-service offers. The queries are split into groups based on what type of entity that is handled during the processing of the query. The queries are also listed based on the HTTP method it uses (GET, PUT or DELETE). The parts of the urls that is marked with [integer] is suppose to be replaced with an integer value, as well as the [boolean] should be replaced with either true or false. Setting the boolean parameter to true will make the service return active entities, false will return inactive entities.

- **Evaluation group**
  - **GET**
    - **vurderingsgruppe/alle**

      Returns all evaluation groups in the system.
    - **vurderingsgruppe/alle?aktiv=[boolean]**

      Returns all active or inactive evaluation groups based on the parameter.
    - **vurderingsgruppe/[integer]**

      Returns the evaluation group with the same primary key as the integer given in the url.
  - **PUT**
    - **vurderingsgruppe/**

      This query either updates or adds a new evaluation group. If the json string contains an id it will update the evaluation group with the same primary key as the id, else a new evaluation group will be added. Returns the new/updated evaluation group.
  - **DELETE**
    - **vurderingsgruppe/[integer]**

      This query will set the evaluation group, with the given integer as primary key, inactive. Returns this evaluation group.
- **Evaluation set**
  - **GET**
    - **vurderingsett/alle**

      Returns all evaluation sets in the system.
    - **vurderingsett/alle?aktiv=[boolean]**

      Returns all active or inactive evaluation sets based on the parameter.

* **vurderingsett/alle?aktiv=[boolean]&vurderingsgruppe=[integer]**

  Returns all active or inactive evaluation sets with evaluation group id equal to the parameter.

* **vurderingsett/[integer]**

  Returns the evaluation set with the same primary key as the integer given in the url.

- **PUT**

* **vurderingsett/**

  This query will either update or add a new evaluation set. If the JSON string contains an id it will update the evaluation set with the same primary key as the id, if not a new evaluation set will be added. Returns the new/updated evaluation set.

- **DELETE**

* **vurderingsett/[integer]**

  This query will set the evaluation set, with the given integer as primary key, inactive. Returns this evaluation set.

- **Evaluation**

- **GET**

* **vurdering/alle**

  Returns all evaluations in the system.

* **vurdering/alle?aktiv=[boolean]**

  Returns all active or inactive evaluations based on the parameter.

* **vurdering/alle?aktiv=[boolean]&vurderingsett=[integer]**

  Returns all active or inactive evaluations with evaluation set id equal to the parameter.

* **vurdering/[integer]**

  Returns the evaluation with the same primary key as the integer given in the url.

* **vurdering/revisjonsgruppe/[integer]**

  Returns all evaluations with "revisionOf" attribute equal to the integer given in the url.

- **PUT**

* **vurdering/**

  This query will always add a new evaluation. If the JSON string contains an id it will set the evaluation with that id to inactive, and a new evaluation will be added. The new evaluation's "revisionOf" attribute will be a copy of the old evaluations "revisionOf" attribute. If the JSON string does not contain an id, a new evaluation will be added and it's "revisionOf" attribute will be a copy of its new primary key (id). Returns the new evaluation.

  - **DELETE**
    - **vurdering/[integer]**

      This query will set the evaluation, with the given integer as primary key, inactive. Returns this evaluation.

- **Comment**
  - **GET**
    - **kommentar/alle**

      Returns all comments in the system.
    - **kommentar/alle?aktiv=[boolean]**

      Returns all active or inactive comments based on the parameter.
    - **kommentar/alle?aktiv=[boolean]&vurdering=[integer]**

      Returns all active or inactive comments with evaluation id equal to the parameter.
    - **kommentar/[integer]**

      Returns the comment with the same primary key as the integer given in the url.
  - **PUT**
    - **kommentar/**

      This query will either update or add a new comment. If the JSON string contains an id it will update the comment with the same primary key as the id, if not a new comment will be added. Returns the new/updated comment.
  - **DELETE**
    - **kommentar/[integer]**

      This query will set the comment, with the given integer as primary key, inactive. Returns this comment.

- **File**
  - **GET**
    - **fil/[integer]/vis**

      Returns the file with the same primary key as the integer given in the url.
    - **fil/alle?vurdering=[integer]**

      Returns the metadata of the files with evaluation id equal to the parameter.
    - **fil/alle?kommentar=[integer]**

      Returns the metadata of the files with comment id equal to the parameter.

- **PUT**
  - **fil/filopp**

    This query will either update or add a new file. If the JSON string contains an id it will update the file with the same primary key as the id, if not a file comment will be added. Returns the new/updated file's metadata.
  - **fil/kopierfil/[integer]?vurdering=[integer]**

    This query makes a copy of the file with primary key equal to the given integer in the url. The new files evaluation id gets set to the integer taken as the last parameter. Returns the new file's metadata.
- **DELETE**
  - **fil/[integer]**

    This query will remove the file with primary key equal to the id given in the url. Returns the deleted file.

- **Option**
  - **GET**
    - **alternativ/[integer]**

      Returns the option with primary key given in the url.

G.3 Installation

1. Install glassfish (http://glassfish.java.net/)
2. Install mysql driver for java on the glassfish server (http://dev.mysql.com/downloads/connector/j/).
3. Add a jdbc resource in glassfish named jdbc/kvrs-vurdering.
4. Deploy the WAR file onto the glassfish server.
5. Install drupal server http://drupal.org.
6. Add the product's drupal modules to you drupal server's modules folder.
   Preferred path is "../sites/all/modules".
7. Set the ip and port of the REST-service by logging in to drupal and browsing
   "http://[drupal-server-ip]:[port]/drupal/main#overlay=admin/config/kvrs/settings".

8. Testing drupal

   (a) Go in to drupal's GUI.

   (b) Click modules at the top bar.

   (c) Find "Testing" (simple test) and enable that module by checking the box and clicking save.

   (d) Click configuration.

   (e) Choose "Testing" in the "Development" section.

   (f) Check KVRS.

   (g) Click run tests.

Appendix H

Evaluation from the customer

Direktoratet for forvaltning og IKT, og tidligere som Norge.no, har i en årrekke arrangert Kvalitetskonferansen, en konferanse hvor de beste nettstedene i offentlig sektor kåres. Dette er et synlig virkemiddel som bidrar til bedre nettsider og tjenester mot publikum, og man ser tydelig hvordan kriteriesett som er utviklet av Difi får følger for utvikling og anskaffelse av nettsider og tjenester i den enkelte kommune og etat.

I alle år har vi brukt et internt fagsystem som vi kaller for KVRS, og selv om verktøyet fungerer helt fint ser vi behov for nytt verktøy på området. Vi så dette som en flott oppgave for studenter i dette faget, og vi delte systemet i to oppgaver, en for kriteriesett, og en for vurdering. Denne gruppen har laget systemet for vurdering, et system som har krav om å kunne kommunisere både via REST-grensesnitt for fremtidige helautomatisert test på enkelte kriterier, men også være enkelt å bruke for de mange konsulentene vi leier inn for gjennomføring av evalueringene. Vi ser også stort potensiale for produktet i forbindelse med etablering av kompetansesenter for universell utforming i Difi.

Demonstrasjon og kodegjennomgang viser at vi har fått levert et produkt som så å si er klart til produksjonssetting. Produktet oppfyller alle våre ønsker og behov, som er kommunisert til studentene. Det har tidvis vært vanskelig å finne tid til å følge opp prosjektet, men siden kravspesifikasjonen er uendret fra dag én ser det ikke ut til at dette har gått ut over produktet.

Produksjonssettingstidspunkt for produktet er ikke fastsatt enda, men det ventes å være i produksjon i god tid før evalueringene skal gjennomføres på våren (Kvalitetskonferansen har fått ny syklus i år, og vil komme neste gang på våren 2013). Vi er godt fornøyd, og systemeierne ser frem til å ta i bruk produktet.