insert title here

os

May 29, 2012

CONTENTS

CHAPTER 1

INTRODUCTION

CHAPTER 2

INTRODUCTION

In order to describe the context of the system, we – as a multi project group – will in the following state the motivation of the project, the group of people we are aiming at helping, the technological platform chosen, the used development method, followed by a problem definition, a system description and architecture, and the conducted usability test.

2.1 Motivation

As this is a student report written as part of a learning project, we are required to comply with the study regulation. The main areas of focus, according to the study regulation, are: multi-project management and quality assurance in the form of requirements analysis, requirements management, and testing. The goal is to create a comprehensive software system, across multiple project groups, in order to enhance our competences in analysis, design, implementation, and evaluation of software applications in regards to the system requirements[20].

This project builds on top of a previous project, and is further developed, with the aim of having other students continue the development. The goal of the project, we are building on top of, is to create a touch based tablet system to support children and their guardians in everyday scenarios.

2.2 Target Group

Our target group is both children and their guardians. These guardians have certain needs for special tools and gadgets that help to ease the communication between them and the children.

Five teachers and educators, who work with children, act as customers. They will provide requirements and information about the institutions' way of working to give us an insight into their daily struggles.

2.2.1 Working with Children with ASD

This section is based upon the statements of a woman with ASD [4], explaining what it is like to live with ASD, and an interview with an educator at Birken, a special kindergarten for children (see appendix A.4 for interview notes).

People with ASD are often more visual in their way of thinking. Rather than visualizing thoughts in language and text, they do it in pictures or visual demonstrations. Pictures and symbols are therefore an essential part of the daily tools used by children and the people interacting with them. Also, children can have difficulties expressing themselves by writing or talking, and can often more easily use electronic devices to either type a sentence or show pictures, to communicate with people around them. Another characteristic of children is their perception of time. Some of them simply do not understand phrases like "in a moment" or "soon", they will need some kind of visual indicator that shows how long time they will have to wait.

Different communication tools for children with autism already exist, but many of them rely on a static database of pictures, and often these has to be printed on paper in order to use them as intended. Other tools, such as hourglasses of different sizes and colors, are also essential when working with children, and these tools are either brought around with the child, or a set is kept every place the child might go, e.g. being at an institution or at home.

There exists tools today which helps the guardians in their daily life, although – as stated in Drazenko's quote – none of them are cost-effective enough to be used throughout the institutions. From the quote, it is clear that there is a need for a more cost-effective solution.

> The price of the existing solutions are not sufficiently low such that we can afford to buy and use them throughout the institu-

tion.

- Drazenko Banjak, educator at Egebakken.

2.3 Target Platform

Since we build upon last year's project, we are bound to use the platform they used, which is tablets running the Android operating system.

In this project we have been provided with five Samsung Galaxy Tab 10.1 devices[16]. The firmware on the tablets is version 3.2. This version, as of project start, is the latest stable version available for these specific tablets. [19]

2.4 Development Method

As a part of the study regulation we have been required to use the same development method in each individual group. Two methods have been considered, *XP* (eXtreme Programming) [21], and Scrum [1].

With the knowledge of both XP and Scrum, we decided in the multi project to use Scrum of Scrums, which is the use of Scrum nested in a larger Scrum project [2].

The reason for choosing Scrum of Scrums is that everyone, at all times, will be able to know what the vision of the project is, and how close every group is to achieving their individual goals of the vision.

Another element of the Scrum method is that a close contact with the customers is maintained. This helps keep the product backlog up to date and correctly prioritized. The customers are presented with the vision of the project, as well as showing the latest release when we have meetings with our customers.

We customized Scrum to fit our project. The changes are as follows:

- The sprint length have been shortened to approximately 7 - 14 half days.
- Some degree of pair programming have been introduced.
- There is no project owner because this is a learning project.

- Everyone is attending the Scrum of Scrums meetings.
- The Scrum of Scrums meetings are only held once at sprint planning.

2.5 Problem Definition

The problem statement is as follows:

> How can we ease the daily life for children with ASD and their guardians, while complying with the study regulation?

This problem statement is necessarily vague to allow the individual groups some freedom in their projects, while we maintain the overall structure of the multi project, however there are limiting factors. We are limited by resources and time available, as we are only working on this project for a single semester. However, all work done in this multi project will be passed on to the next line of students, which means we can make a full system design and pass on anything we do not have the time or resources for. This also requires that our work need to be of such quality that it is understandable by students of the same educational level as ourselves.

2.6 System Description

GIRAF is a collection of applications, either fully or partially interdependent, for the Android platform, designed to be used by guardians and children. GIRAF consists of five projects with various degree of interaction. These projects are named Launcher, PARROT, WOMBAT, Oasis, and Savannah. Each of the groups have produced individual products, which are parts of a greater project, GIRAF.

Launcher handles execution of GIRAF apps, and at the same time it provides safety features to ensure that a user that is not authorized to interact with the rest of the system will not be able to do so. When the launcher executes an app, it will provide it with profile information, specifying which child is currently using the app, as well as which guardian is signed in.

PARROT is an app which provides access to pictograms – pictures with associated information such as sound and text – which can be used for communication. PARROT also gives guardians functionality for adding additional pictograms, as well as organizing the pictograms into categories for ease of access, based on the needs of the individual child.

WOMBAT is an app which purpose is to help the children to understand the aspect of time, by visualizing it. WOMBAT provides different ways of displaying time, as well as the possibility to configure the app for the needs of individual children.

Oasis locally stores the data and configuration of the GIRAF platform, and provides an API to access it. The stored data and configurations are synchronized to the Savannah server, if available. In addition, an app is provided for the guardian to access the stored data and configurations.

Savannah provides Oasis with a way to synchronize tablets running GIRAF. Furthermore, a website is provided to ease administration of the synchronized data.

2.7 Architecture

Our System architecture – shown in Figure 2.1 has been designed with simplicity in mind and was greatly inspired by the MVC pattern. This means that the architecture is divided into three layers. The lowest layer is the database where the information is stored. Above this layer is the controller layer which, in the GIRAF platform, is known as Oasis. The controller is responsible for querying the database for information needed in an app and the controller is also responsible for storing information in the database. The last layer is the apps. This division of layers give the GIRAF platform a low cohesion which makes it easier to work with individual parts of the platform independently.

We have chosen to redesign last year's architecture [7] to make it easier to work with. We have simplified the architecture because we feel it is unnecessarily complex.

2.8 Usability Test

As stated in the motivation, quality assurance through testing of the system is required. Therefore a usability test was conducted in order to measure the current usability of the GIRAF platform as a whole, as well as of the individual parts of the platform. Furthermore, the next wave of developers

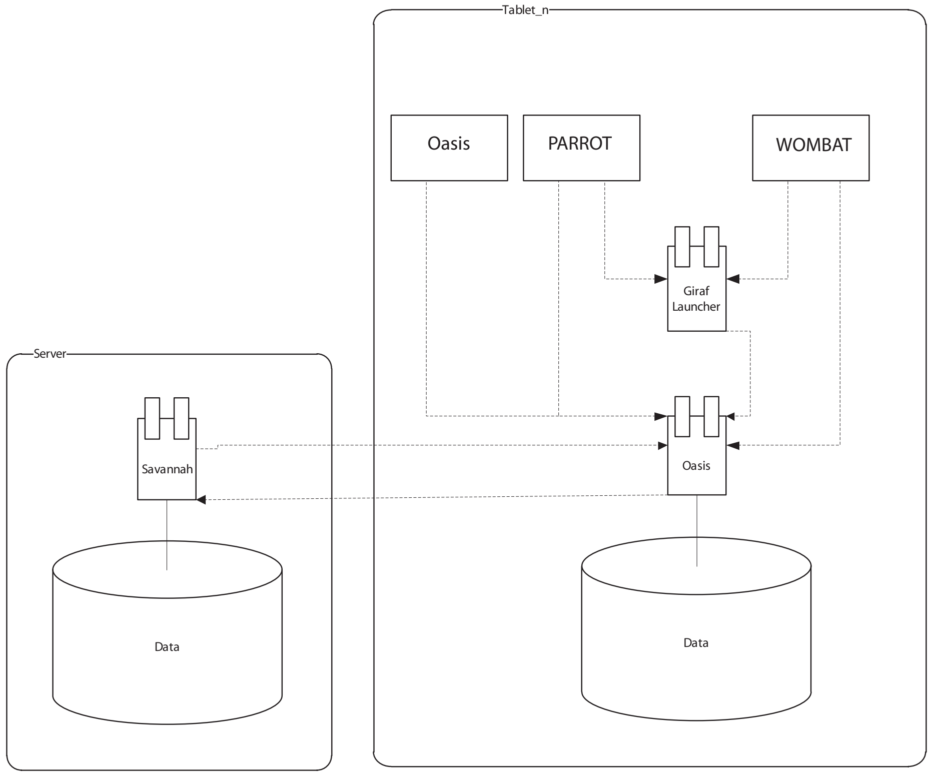


Figure 2.1: The GIRAF architecture

will immediately be able to start correcting the found usability issues.

2.8.1 Approach

The test group consists of the five contact persons. We assess that they, as a test group, are representative. We base this on them being a mix of educators and teachers, with varying computer skills.

They have prior knowledge about the overall idea of the GIRAF platform, and although some of the contact persons had previously informally used some aspects or parts of the system, they had not been exposed to the platform as a whole, and therefore still are of value.

The invitation sent to the test persons can be found in Figure A.5.

The Instant Data Analysis (IDA) method for usability is chosen. A traditional video analysis method could be used, but since IDA is designed for small test groups, this approach is used. [9]

Setup

The usability test is divided into two tests: A test of three user applications, and a test of two administrative applications. The user applications are: The launcher, PARROT, and WOMBAT. The administrative applications are: The Oasis app and the Savannah web application. Each test is assigned a team to accommodate the need to run two tests simultaneously. The teams are made with respect to the criteria of the Instant Data Analysis process. Each team consisted of:

- 1 x Test Coordinator
- 1 x Test Monitor
- 1 x Data Logger
- 2 x Observers

The usability lab at Aalborg University is designed with two rooms for usability testing and a control room to observe and record the tests. The two test chambers are assigned a test each and the control room is used to observe both tests as seen in figure 2.2.

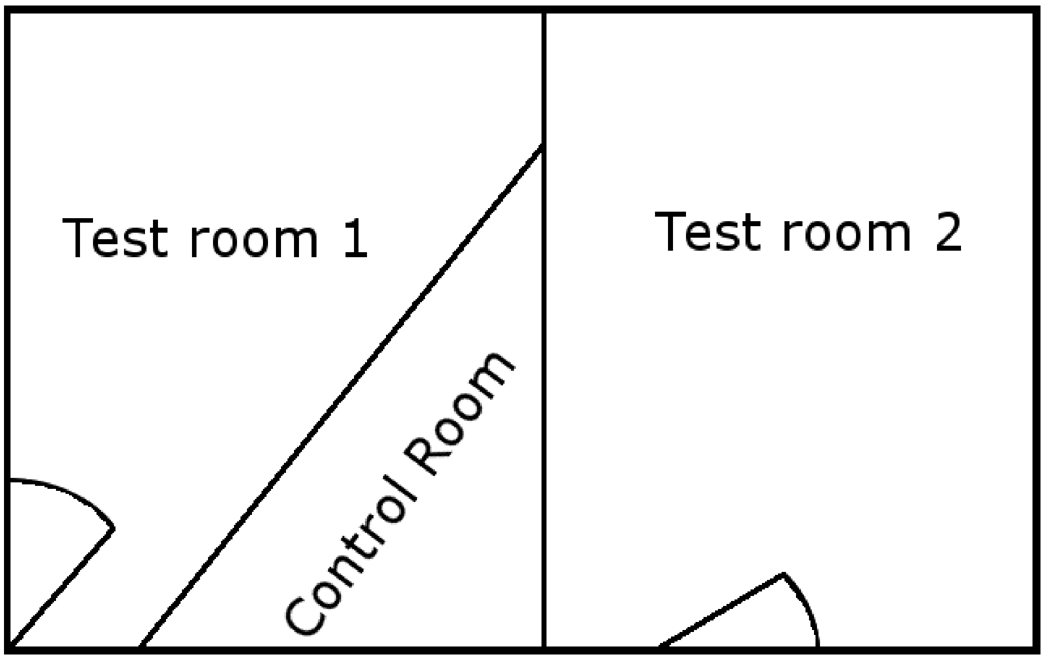


Figure 2.2: An overview of the usability lab at Cassiopeia, Department of Computer Science, Aalborg University.

Execution

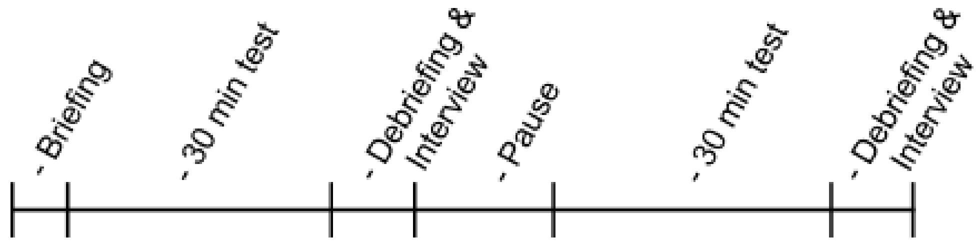


Figure 2.3: The schedule of the usability test.

The tests are conducted according to the schedule in Figure 2.3.

Briefing, debriefing, and questionnaire documents can be found in **??**, and the results of the test can be found in **??**.

CHAPTER 3

AGILE DEVELOPMENT/DEVELOPMENT METHOD

3.1 Theory

One of the goals of this semester is to show that we have knowledge in choosing and implementing different work methods and apply the method that we find most suited for our project. In order to do this one must first know the different methods before he/she will be able to choose the best suited.

Different work methods can be put into two categories: Traditional and Agile. The difference of traditional and agile methods are what they emphasizes the most.

Traditional methods puts big emphasize on analyzing the problem and create documentation of the analysis. The structure of a traditional method is therefore often split into phases, for instance the analysis phase or the implementation phase. Once a phase is done you move to the next phase. An example of this is the waterfall method. [put some example]

Agile methods on the other hand puts emphasis on analysis and documentation in a different way. Most agile methods utilizes an iteration driven work style. This means that you do all the phases over and over. This also means that you do not do the whole project at once. Instead you focus on getting a part of the system to work and then as more iterations are completed you add on to the system until the system is ready to be released. An example of this is the SCRUM method. [put some example]

Both approaches have strength and weaknesses. The traditional method features lots of documentation and analysis and so it is easier for others to understand if they are to develop further on the system. It also create a great overview of the whole system, and thus it is easier to estimate how long time it will take before the system is ready to ship. This makes traditional methods well suited for large projects. The agile method on the other hand implements the iteration driven approach. One of the great features of this approach is that you can correct errors and include things you might have forgotten relatively easy, because you just do it in the next iteration.

One of the big drawbacks of the traditional method is that you cannot just go back a step, if you realize you have forgotten something or made an error. This is very costly because you have to do lots of steps over again. Another big issue with the traditional method is that the customers of the system might not always know what they want or need at the start of the development, but this is where you do all the analysis. The Agile method excels at this as you can present to your customers your current build and receive feedback that you might not have considered.

The Work method that we find best suited for our project is an agile method. This is because in this semester we are working together with educators from institutions working with children with ASD. They will have requirements for our system so it is important that we are taking them into the project. It is also important that we choose a development method that have tools for managing bigger project groups as we are working as part of a bigger team to develop a system.

We have considered These following agile work methods:

- Scrum
- Xp

In short Scrum is a method that emphasizes a self-directed and self-organizing team, each iteration is client driven meaning that the clients provides the requirements and features will be prioritized according to the clients needs.

XP emphasizes programming and testing. This means that there are very little documentation of the system other than the code. A simple design is preferred and the code is refactored with high frequency. This method requires high discipline since most planning is done orally.

3.2 Scrum implementation

In our project it was required that all groups used the same development method to keep things simple. This means that the chosen development method may not be perfectly suited for every group, so while we have chosen the Scrum method we have also made some adjustments to fit it smoother for our group.

The key points of the adjustments are as follow

- The daily scrum meeting
- The customer involvement
- The sprint length
- Incorporating key elements from Xp method

We, in our group, have not utilized the daily scrum meeting that is very typical and defining for the scrum method. The reason for this is that we are already sitting together in the same room. We also know each other well because we have worked together in earlier projects. If we felt like we needed to know anything about the other team members progress we could just ask. We simply did not feel the need to waste time every morning by stating the obvious. We see the daily scrum meeting as a tool for communication. We agree that communication is important to make a great product but we have used alternative ways to communicate.

Customer involvement have been a big focus in the multi project. However our group has been in a different situation than the others because our product has been the server which has little to no interest for the customers. Our biggest requirement providers has been the other groups. This has meant that we have not had a release after each sprint that was shown to the customer for feedback. This was only done once and only for the web interface. As mentioned in section [] a usability test was carried out to get a little more feedback from the customers.

The sprint length has been modified because the "number of days" interval would be misleading because we have some days with lectures and others without. Instead we have define the term "half days" which covers either from 8.00 to 12.00 or from 12.00 to 16.00. This interval makes it possible to use days, with lectures covering up half the day, as working days. The sprint length has been dynamic and has been decided at each sprint start. A typical sprint length could be 14 half days which means the next 10 half days to occur. This could be a week without any lectures, or two weeks with lectures.

There is a specific element associated with the Xp method that we felt the need to incorporate into Scrum, being the pair programming. We have incorporated this element because we find it well suited for programming while leaning. Because we had to learn a new programming language, Java servlet, that none of us had ealier experience in, pair programming helped us a lot in the beginning to quickly catch on.

CHAPTER 4

SAVANNAH

4.1 Requirements

Initial Requirement gathering

Requirement gathering was done in the multi project group in the very early stages of the project, but has also been updated as we have been progressing and showing our work to our customers. In the initial stages we did semi-structured interviews of our customers, where the primary focus was understanding our target group and exploring any tools they currently have access to, and allowing the customers to present their own ideas and visions of the project.

From these interviews we created an initial list of requirements for the multi project. three interviews were done: Mette and Kristine from Birken, a kindergarten for kids with special needs , Drazenko from the school Egebakken, and Tove from Tale Instituttet,the speech center. From the individual interviews we gathered a list of their individual ideas and visions and made a requirements list which groups later could refer back to.

- Mette And Christina

    Customizable software

    Ease of use, compared to current physical tools

    Emphasize visual stimuli

Continuous stimuli

- Drazenko

  Customizable software

  Visual abstract concept

  Emphasize visual stimuli

  Unambiguous

  Consistency/Structure

- Tove

  Customizable software

  Engaging/Entertaining software

  Authentic/proper feedback or behavior

From the list of requirements of the individual customers, we identified the requirements which were common for all of them, and made a list of requirements which should be common for all groups in the multi project.

- Customizable software

  Some way to distinguish unique users on the same tablet is required, since people with autism have very different needs and have very different perceptions of the world.

  Customization should trump feature bloat.

- Emphasize visual stimuli

  Autistic people are visually over stimulated.

- Authentic/proper feedback or behavior

These are all requirements set by our customers, however, since this system is dealing with potentially sensitive personal data, it is also a requirement that before it can be deployed in a real world situation, that all transmissions are done via a secure connection.

Continuous requirement gathering

As an integral part of Scrum, we have had continuous requirements gathering. For Savannah the continuous requirements gathering come from discussing the needs of the other project groups of the database, as well as talks with the customers during the project.

During the project we have gathered these requirements:

- PARROT

  Linking audio and images in the database

- Launcher

  QR support and storage in the databas

- Oasis

  Updating of database schema in cooperation with Oasis

  Retrieval styles: Full profile and children associated with a specific guardian.

- Customers

  Minor changes to the web interface

The majority of requirements for Savannah has come through cooperation with Oasis during the design phase of the database, which was a running process that ended at the beginning of the 4th sprint where the database was frozen. Any requirements that may have arrived beyond this point was added to the server backlog for future development.

4.2 Database

To be able to have profile control in the system, a way to keep track of the various profiles are required. A MySQL database has been designed and implemented to handle this. This section describes the process of the database creation.

4.2.1 Design

The database is designed in MySQL 5.1.61, and resembles the local database created by the “Admin-group” very closely – the only difference is, that the Savanah database has two extra fields in the “AuthUsers” table: username and password. The reason for this is, that while all apps running on the android platform uses QR-codes for user authentication, this is not a fitting choice for the webinterface; it would be more complicated to scan the QR-code form a webcam to login, than to simply use a username and password combination.

Requirements

The requirements for the database has been provided by the app groups, and are as follows:

- All users must be able to login with a QR-code
- Each department must also be able to login
- All users must be linked to at least one department
- All guardians must be able to be linked to at least one child.
- All parents must be able to be linked to at least one child.
- All users must be able to have access all the apps
- All users must be able to access their own pictures, and all public pictures
- All departments must be able to access all public pictures
- All pictures needs to be able to be linked to different tags
- Pictures must be able to be linked with audio and vice versa
- A department must be able to have a subdepartment[1]

[1]Example: “Birken” has two departments, at two different addresses, both these subdepartments needs to be linked with “Birken” as a superdepartment

Diagram

A diagram has been made to get a overview of the different tables and their relations. This diagram is designed in DIA, which is a GTK+ based diagram creation program[10], and can be seen in Figure 4.1.

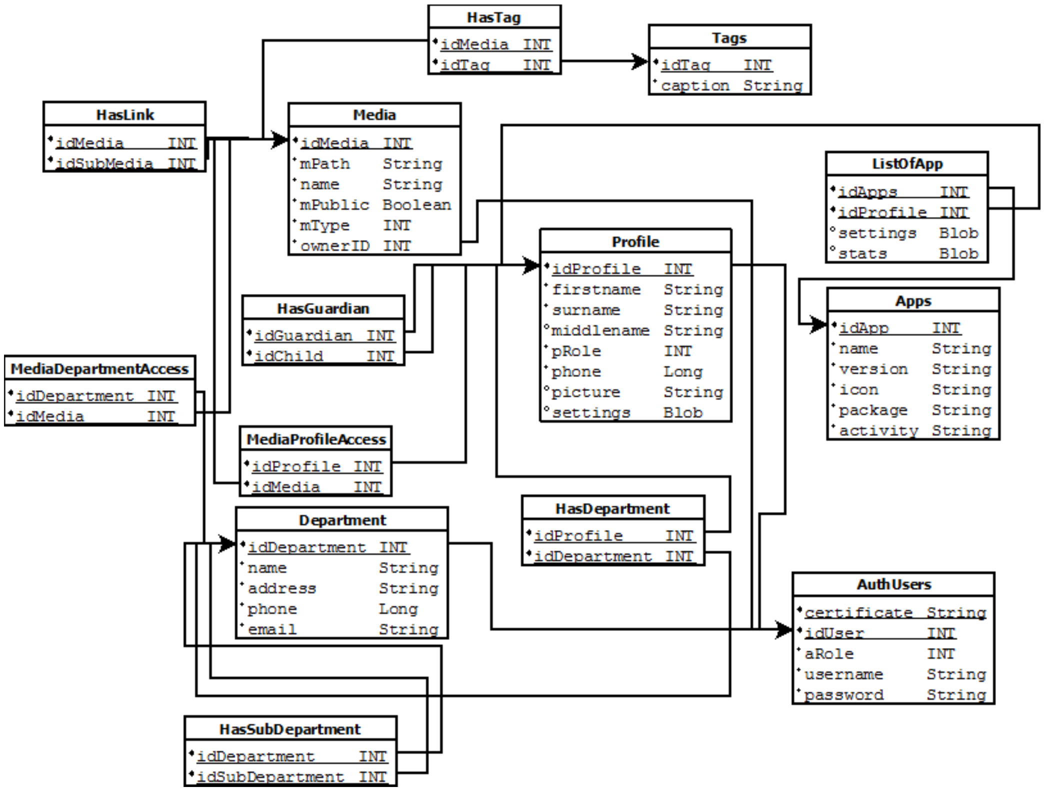


Figure 4.1: Dia diagram of the database

Although the diagram in Figure 4.1 does not meet the standard of database diagrams, it does provide a good view of the different tables, and the relations: Foreign keys points to where the value is fetched form. This crude diagram shows that be having the “idUser” from the “AuthUsers” one is able to access all other information for that specific user.

Normal form

To prevent modification anomalies in the database, it can be normalized. This becomes important when a table is dealing with functional dependencies. The solution is to remove the dependicies form the table to another (e.g. split a table into two). An example of this is shown in Table 4.1 and Table 4.2.

Sales		
Customer_ID	Product	Price
1001	Laundry detergent	12
1007	Toothpaste	3
1010	Chlorine bleach	4
1024	Toothpaste	3

Table 4.1: Non-normalized database[18, p. 114]

The problem in Table 4.1 is, that if customer 1001 is removed form the database, not only his products are removed, but the fact that Laundry detergent costs 12$ is also lost. A way to cope with this, is to split the table into two tables, as seen in Table 4.2

CUST_PURCH	
Customer_ID	Product
1001	Laundry detergent
1007	Toothpaste
1010	Chlorine bleach
1024	Toothpaste

PROD_PRICE	
Product	Price
Laundry detergent	12
Toothpaste	3
Chlorine bleach	4

Table 4.2: Normalized database example[18, p 115]

As seen in Table 4.2 the table "CUST_PURCH" now only deals with the customer, and thus it is no possible to remove a customer form the system, without loosing the data about the product.

There are several degrees of normal form for a database, here the focus is on the first three (1st, 2nd and 3rd normal form).

First normal form (1NF)

The example in Table 4.1 and Table 4.2, is an example of 1NF. To achieve this the following rules must apply[18, p. 116]:

- Each cell (intersection of a row and a column) of the table must have only a single value.

- Each column must have a unique name.
- No two rows may be identical (that is, each row must be unique).
- Each column contains data for a single attribute of the thing it's describing.

As seen in Figure 4.1 the Savannah database is follows the rules for 1NF, and thus is 1NF.

Second normal form (2NF)

For the database to be in 2NF, if a table does contain a composite key, all other attributes in the table must be depended on the entire key. Furthermore [18, p. 117] states:

> ...every relation that is in 1NF with a single attribute key is automatically in second normal form.

As this is the case in the Savannah database it is in 2NF.

Third normal form (3NF)

For a database to be en 3NF, it is required that there are no transitive dependency[2]. As seen in Figure 4.1 does live up to this definition. There are no transitive dependencies within any of the tables. A good example of this is the handling of a "Profile"'s access to "Apps": When the app is removed from the "Profile" it is done in the "ListOfApps" table, and thus only the relation is removed, and the specific app still is available in the system. This makes the Savannah database to be in 3NF.

Rules

To provide the security needed in the system, a few rules need to apply for the databse:

- The "Profile"→"AuthUsers" relation must be one-to-one, as one user form the "AuthUsers" can only have one profile in the system

[2]A transitive dependency occurs when one attribute depends on a second attribute, which depends on a third attribute. Deletions in a table with such a dependency can cause unwanted information loss. [18, p. 118]

- The "Department"→"AuthUsers" relation must be one-to-one, as one department form the "AuthUsers" can only be one department in the system
- The "idUser" in "AuthUsers" must be unique.
- The "username" in "AuthUsers" must be unique
- It must be possible to distinguish between users and departments in the "AuthUsers" table
- It must be possible to distinguish between children, parents and guardians in the "Profile" table

These rules will be applied in a mix between SQL script and software level in **REF TIL SECTION**

4.2.2 Implementation

The implementation of the MySQL database is done in MySQL Workbench 5.2.40 which is a GUI tool that provides a large set of features for various purposes.

Due to the dependencies of the various tables, the order of how the need to be created is quite strict. The code to create the tables is found in the appendix section A.1. To make sure the "username" and "userID" follows the previously stated rules, both fields has been made `UNIQUE NOT NULL` and the "userID" furthermore has `AUTO_INCREMENT` as seen in Listing A.1. This makes sure there can be no one with the same "username" or "userID" and the "userID" will automatic increase when new data is inserted. This also guarantees a one-to-one relation between both "Profile" and "AuthUsers" and "Departments" and "AuthUsers". To be able to distinguish between "Department" and "Profiles" a filed called `aRole` is used. This is a int and will hold a number, which will be used at software level to do the distinguishing. The same applies for the distinguishing in "Profile" (code shown in Listing A.4), here a field called `pRole` holds an int, which again will be used at software level.

The MySQL Workbench provides the functionality to create a ERR diagram form an existing database, this diagram is shown in Figure 4.2. The diagram is con completely as MySQL workbench creates it, the real is shown in the appendix section A.2. But this is a error in the software; as seen in the diagram, the tool generates the "Profiles"→"AuthUsers" as a one-to-many relation. This however, is not possible since the "idUser" field in "AuthUsers" is unique (as seen in Listing A.1), the same goes for both "Departments" and "Medias".

To make the deletion of data easy, many of the filed in the tables has the constraint `ON DELETE CASCADE`. This can be "dangerous", as side-effects can result in unintended data will be deleted. To makes sure no unintended data will be deleted, the software level implementation needs to warn the user when deleting data. Furthermore an analysis has been made to argue for which fileds can have this constraint. The following tables and fields has the cascade constraint:

```
‘‘Profiles’’.’’idProfile’’,
‘‘ListOfApps’’.’’idProfile’’,
‘‘HasDepartment’’.’’idProfile’’,
‘‘HasGuardian’’.’’idGuardian’’,
‘‘HasGuardian’’.’’idChild’’,
‘‘Media’’.’’OwnerID’’,
‘‘HasTag’’.’’idMedia’’,
‘‘HasLink’’.’’idMedia’’,
‘‘HasLink’’.’’idSubMedia’’,
’’MediaProfileAccess’’.’’idProfile’’.
```

Use case

An use case of the constraint is:

"A user "Jesper" wishes to delete his entire profile"

What will happen is:

1. A deletion of the "userID" in "AuthUsers" is executed
2. The relation between "AuthUsers" and "Profiles" will delete the profile
3. The relation between "Profiles" and "HasGuardian" will delete all fields where the "userID" is either "idGuardian" or "idChild"
4. The relation between "AuthUsers" and "Media" will delete all fields where "idUser" is the owner

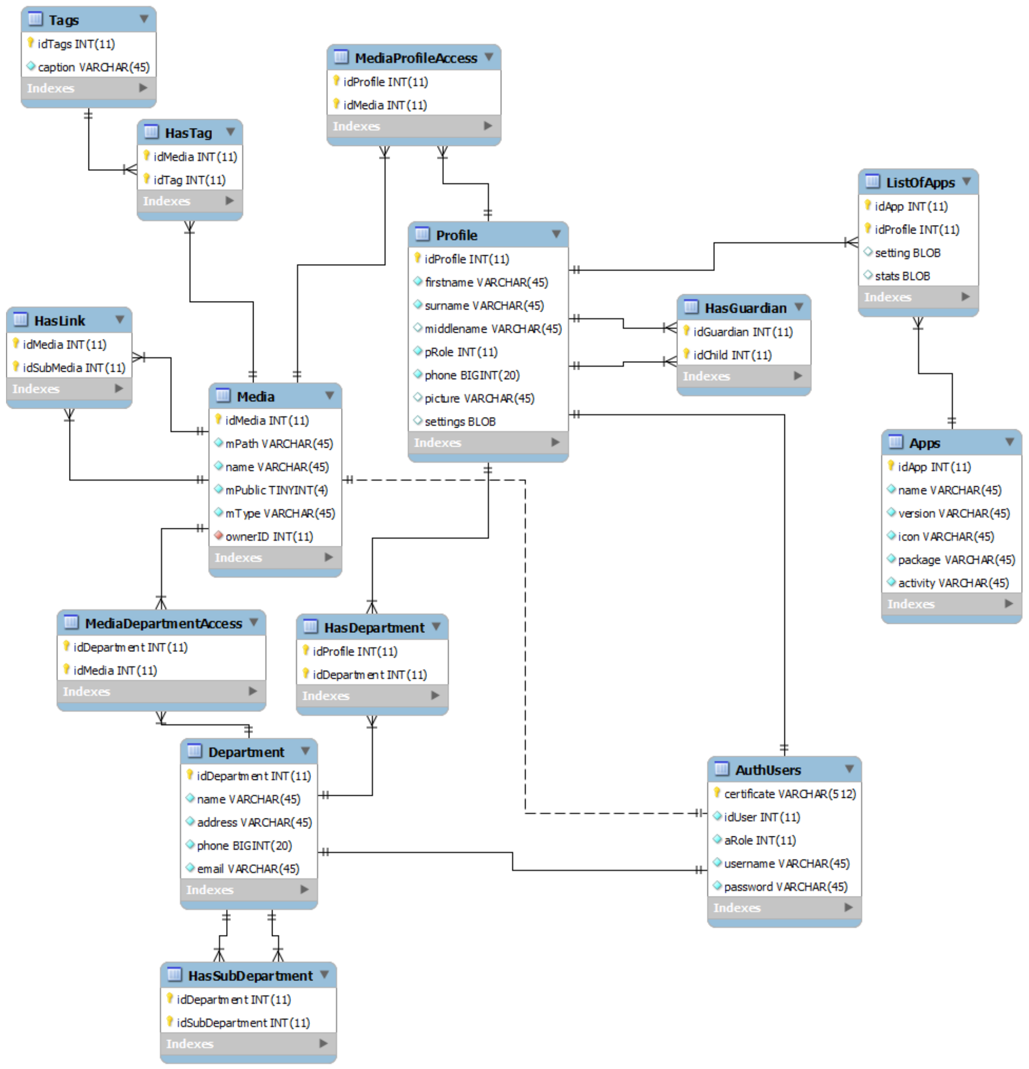


Figure 4.2: The ERR diagram

5. The relation between "Media" and "HasTag" will delete all fields where "HasTags"."idMedia" equals "idMedia"

6. The relation between "Media" and "HasLink" will delete all fields where "HasLink"."idMedia" or "HasLink"."idSubMedia" equals "idMedia"

4.2.3 Test

4.3 Web interface

4.3.1 Programming language (Jesper)

To be able to provide the desired functionality form the web interface, the web pages must be dynamic and able to interact with the database. There are several different programming languages to choose from. Some of the most common are ASP.net, PHP and Java servlets. Due to the fact that ASP.net is not open source [14], and the project is, this is not a feasible choice. This section will do a deeper analysis of PHP and Java servlets, to argue for the best choice of programming language.

PHP

PHP originally emerged in 1994 when creator Rasmus Lerdorf wrote a simple set of Common Gateway Interfaces (CGI)[3] to track visits on his online resume, and decided to name it "Personal Home Page Tools" (PHP Tools). As more functionality was desired, he rewrote PHP Tools to fit his demands, and in mid 1995 he released the source code to the public.
In 1997 version 3.0 of PHP was released, and the name changed form "Personal Home Page Tools" to the recursive acronym "PHP: Hypertext Preprocessor" as which it is known today. This is the first version that colsely resembles PHP as it exists today [12].

Key functionality

The following list is a segment of functionality found in PHP 5.4.3. This is based on [11].

[3]The Common Gateway Interface (CGI) allows an HTTP server and a Computer-generated Imagery script to share responsibility for responding to client requests [3].

Operating system PHP is supported by all major operating systems, including Linux, many UNIX variants, Microsoft Windows, MAC OSX etc.

Output PHP provides the functionality to output not only HTML, but also images, PDF files and Flash movies, all generated on the fly

Databases PHP supports a wide range of different databases, including but no limited to MySQL, SQLite and PostgresSQL. The entire list of supported databases can be found at `http://www.php.net/manual/en/refs.database.php`.

Protocols PHP can use other service protocols than HTTP, and it is possible to open raw network sockets.

Requirements

To be able to run PHP on a web server, this needs to support PHP. Php.net recommends using Apache web server, and and for database control MySQL [13].

Java Servlets

The first version of Java servlets was created by Sun Microsystems in mid 1997, and in December 2009 version 3.0 was released[22]. It was developed to use the advantages of Java to solve the problems of performance, scalability and security in CGI[5].

Key functionality

The following list is a segment of functionality found in servlets. This is based on [6].

Portability Because servlets are written in Java, they are platform independent.

Power Servlets can use the full functionality of core Java APIs; this includes URL access, multithreating, image manipulation, database connectivity etc.

Efficiency When a servlet is loaded, in remains in the servers memory as a single object instance, this makes it able to handle requests almost immediately.

Safety Due to servlets being written in Java, they inherit the strong type safety.

Requirements

To be able to run servlets on a web server, this needs to support Java servlet. Apache has made a service to allow the execution of servlets, called Tomcat.

Comparison between PHP and servlets

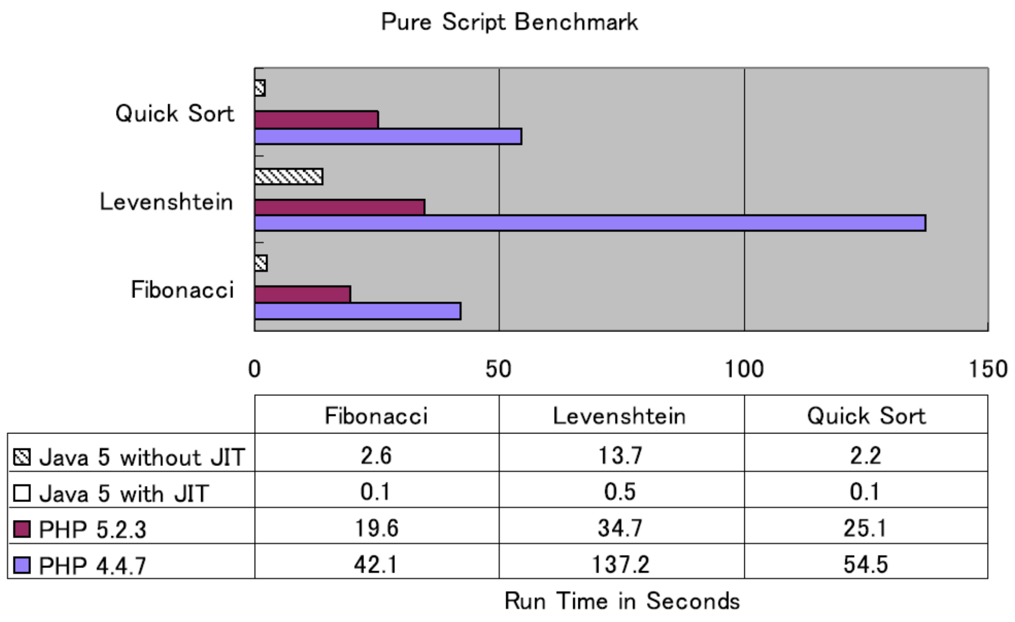
When comparing PHP and servlets, the focus is put on efficiency and number of active sessions in both languages, as this will be the main requiremtns for the web interface. The following is based on [17] - a research paper by “IBM Tokyo Research Laboratory”.

Efficiency

The web interface must be able to run fast and smooth, even on high load. A pure benchmark test is found in Figure 4.3. The test calculates a quicksort which sorts 100 integers, A Levenshtein alogithm to measure the similarity between two strings of 56 characters and a Fibonacci execution which calculates the 15th values, with two random starting values. The setup is:

> We compared the total run time of executing each test 10,000 times with each engine. We also executed each benchmark an additional 10,000 times as a warm-up, before the measured test. This prevents Java just-in-time compilation overhead from impacting the score in the Java tests. We ran the experiment on an Intel Pentium 4 CPU at 3.40 GHz with 3GB RAM Memory, with the Linux 2.6.17 kernel. [17, p. 167]

As seen from Figure 4.3 servlets are in all test faster, both with and without the Just In Time (JIT) compilation.



	Fibonacci	Levenshtein	Quick Sort
▨ Java 5 without JIT	2.6	13.7	2.2
□ Java 5 with JIT	0.1	0.5	0.1
🟥 PHP 5.2.3	19.6	34.7	25.1
🟪 PHP 4.4.7	42.1	137.2	54.5

Figure 4.3: A pure script benchmark calculating the average time of 1000 [17]

Sessions

The web interface must be able to handle many active sessions at the same time. A test of this found in [17, p. 173], is shown in Figure 4.4.

> [The figure]... shows the maximum performance for each configuration and scenario, as determined by the maximum number of simultaneous sessions (e.g., users) which can be supported with acceptable Quality Of Service as defined by SPEC [17, p. 173].

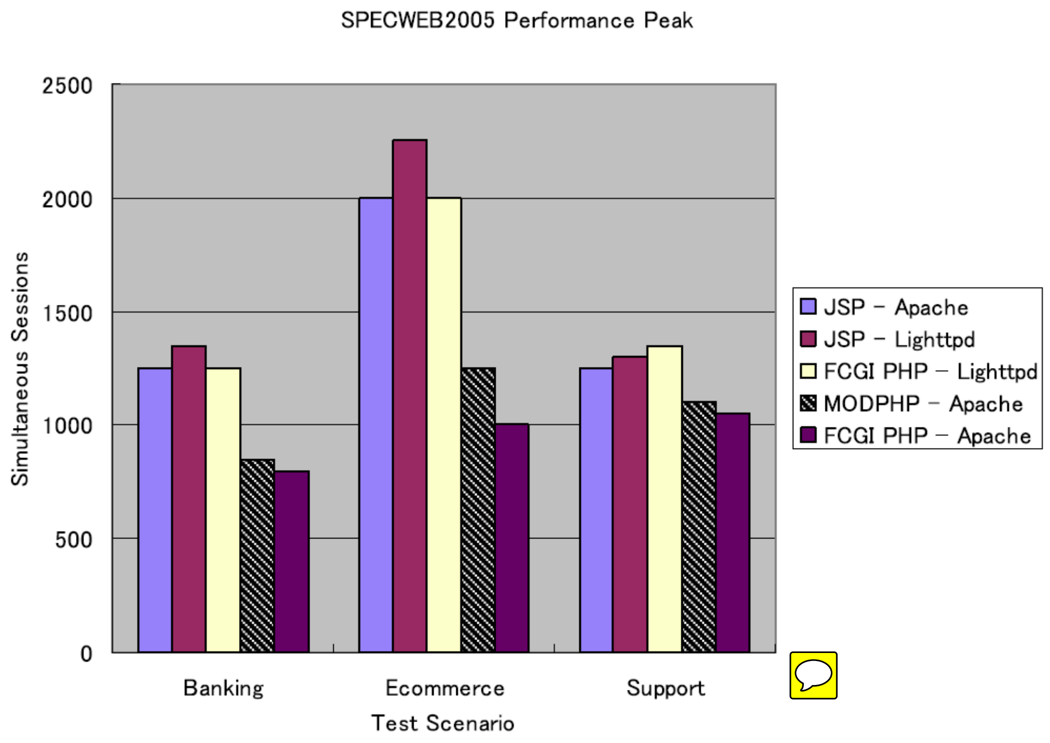


Figure 4.4: TROLOLOLO

Figure 4.4 shows that in 2 out of 3 test cases JSP is able to handle more active users than PHP.

Even though the servlets genually is more powerfull than PHP, this only comes to show, when the servler load is high (around 750 active sessions at the same time). Eventhough the efficiency of JSP is better than PHP [17] sums up the descussion quite well:

> When implementing a web server system which will never experience high load, or in which performance, throughput, and reliability under high load is not an issue, then the use of any of the analyzed languages or web servers will achieve similar performance results. If outstanding performance and throughput is the primary goal, then the use of JSP over PHP is advisable. However, if a 5-10% difference in throughput and performance is acceptable, then the implementer of a web system can achieve similar results using either PHP or JSP. In which case, other requirements such as developer language familiarity and programming efficiency, maintainability, security, reliability, middleware compatibility, etc. would be the deciding factors [17, p. 181].

We have chosen to implement the web interface in servlets, due to the facts that we have a lot of experience in Java already and the Andoid apps are developed in Java, so choosing this seems fitting.

4.3.2 Implementation

4.3.3 Test

Our usability test was carried out using the instant data analysis (IDA) method to process our results. The results are as shown in Table 4.3

Error	Frequency	Category
Create a profile was difficult and counter intuitive	3	1
The home button was only on some sites and confuses the user	5	1
/imagenull would magically appear for no reason when a user was created	3	0
couple of HTTP errors during the test	5	2
Access to apps caused confusion, the user could not figure out what it was used for	5	0
Choose file was not further specified and caused confusion	3	0
UI size was not filling up the whole screen which some of the persons found inconvenient	2	0
QR code missing home path	3	0
Add guardian to child was counter intuitive	3	2
Edit profile made the users believe that they had to change password every time	2	1
When logging in on child's profile as guardian the dropdown menu was confusing	4	1
The user was confused of what to do after login	1	0
Edit/Add picture was confusing	3	1
The phone number was required but this is not the situation in reality	5	2
Missing caption for choosing role causes confusion	1	0
The test subject was unsure which tags responded to which checkbox	2	2
Confusing, too many options	1	1
The test subject tried to login as one of the predefined users because they thought it was their profile	2	0
The user lost the overview	5	1
The user was not sure which pictures were public when asked to delete a public picture	1	0
When deleting pictures, the tags in brackets caused confusion and the text fields implied that you could write in them which is not the case	1	0
The test subject was unsure how to get back from the audio page	1	0

Table 4.3: Results from usability test

To sum up 11 errors were defined as cosmetic, 7 as serious and 4 as critical.

The critical errors where that the page broke down with an error which is an indication of programming errors which is clearly a show stopper. another critical error was that you had to include a phone number for a child while it should be optional. This is a requirement that has not been implemented because we have not gotten the request before the test. When prompted to add a guardian to a child there was great confusion of how to do that. The reason is mainly because it is not placed in an intuitive place. To add a guardian you go to edit profile, choose the child and add it in the list box.

This was not clear to the test subjects and they had to get help from the test monitor. The last critical error was the tags selection. The problem was that the user did not know if a checkbox was assigned to the caption left or right for it. While this seems as a cosmetic or serious problem at worst the system decided to crash in one of the tests. We have not been able to recreate this crash so we have rated the problem as critical although the system might not crash.

The rest of the problems found in the test was primarily things missing, bad layout or misplaced functionality. Missing things include "back" or "home" options from certain pages or captions that tells the user what certain fields do when filling out formulas. Bad layout is the main reason for confusion when using the system, this could be too much information presented to the user at once. Misplaced functionality includes the placement of the profile list on the main page which a lot of users thought had to do with editing and adding profiles leading to big confusion.

4.4 The Server

The following chapter concerns the design and implementation of the server side software for savannah and the sw6ml language.

4.4.1 Architechture and Design

Architecture

An overall design and architecture was created in the early sprints of the project. The server side software is different from the rest of the software made in the multi project group, in that the the customers will never actually see it in action, it works perfect if they never notice it is there. Almost all requirements from our customers concern the user interface and general feel of the apps for the mobile devices. In regards to the server software it is the requirements of the multi project that are of interest, in particular the oasis group, as their responsibility is to link the rest of the world to us.

The giraf system is designed in such a way that it deploys with two databases: Savannah - our project, a global database for a full deployment unit, and Oasis - or localDB, a local database which exists only on the mobile devices, which has an almost identical schema to Savannahs database. Rather than having Oasis query the global database directly, it was decided to implement access to both the databases through a software layer, as shown in Figure 4.5. The pros and cons of this seemingly redundant software layer was considered, see Table 4.4

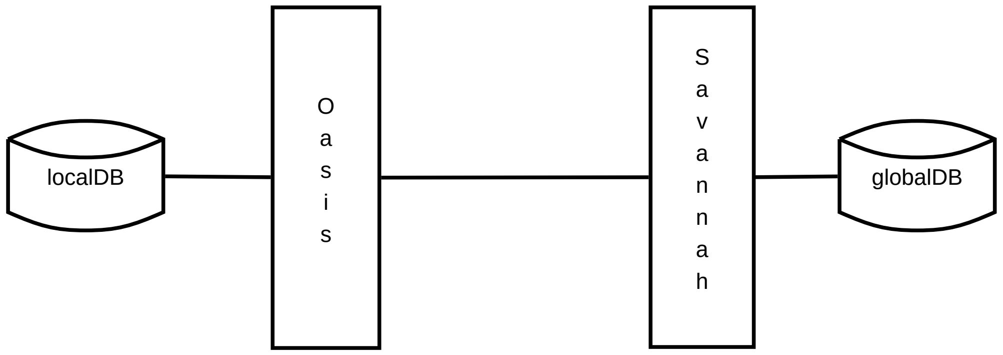


Figure 4.5: Software layers

Pros	Cons
More flexibilty	higher complexity
Independent DB updates	Lower performance

Table 4.4: pros and cons of an extra software layer between the databases

Having an extra software layer between the databases means we can alter the global and local databases independently of each other. By providing software with methods for extracting specific details from Savannah, like full profiles, oasis does not have worry about Savannahs internal database schema. The downside of this is that the complexity inevitably will be higher, and performance will be lower, the performance issue is not critical though, since the perceived performance of the system will be dominated by the bandwidth of the mobile devices. The communication between the software layers will be facilited by the sw6ml XML language, presented in section 4.4.1

Savannah has a three layered internal architecture, show in Figure 4.6.

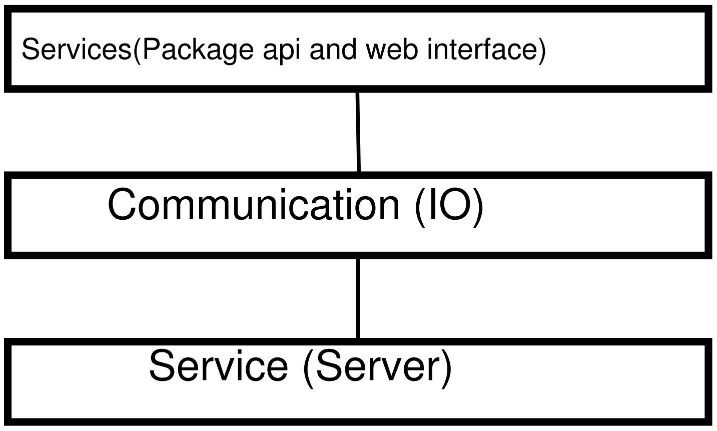


Figure 4.6: Savannahs architecture

A short description of each layers responsibilities follows:

Services The services layers consists of the services that we provide to external users. We have considered two services to include, which are an API for creating transmissions packages that savannah understand, and the web interface discussed in section 4.3.

Communication The communication layer consists of the IO package of the server side software, which handle retrieval and responding.

Service The service layer consists of event handling, query handling and building of sw6ml documents.

Design

The overall server software is designed around a producer-consumer pattern, where Oasis acts as the producer through savannah's IO layer. Request or commit packages send to savannah will be processed by the `IOHandler` which will add an event to the `EventQueue`. The consumer is the server itself, which will remove events from the queue and process its content, being it a request or a commit. Using a queue based design was chosen to for sake of simplicity, the project is a part of learning process and building a server with concurrent event handling was down prioritized versus a simpler server, which would allows us to gain experience and still meet the requirements of the study regulations. A mock overview of the design of the server can be seen in Figure 4.7, this diagram represents the general idea and flow of the server and should not be understood as formal technical specification.

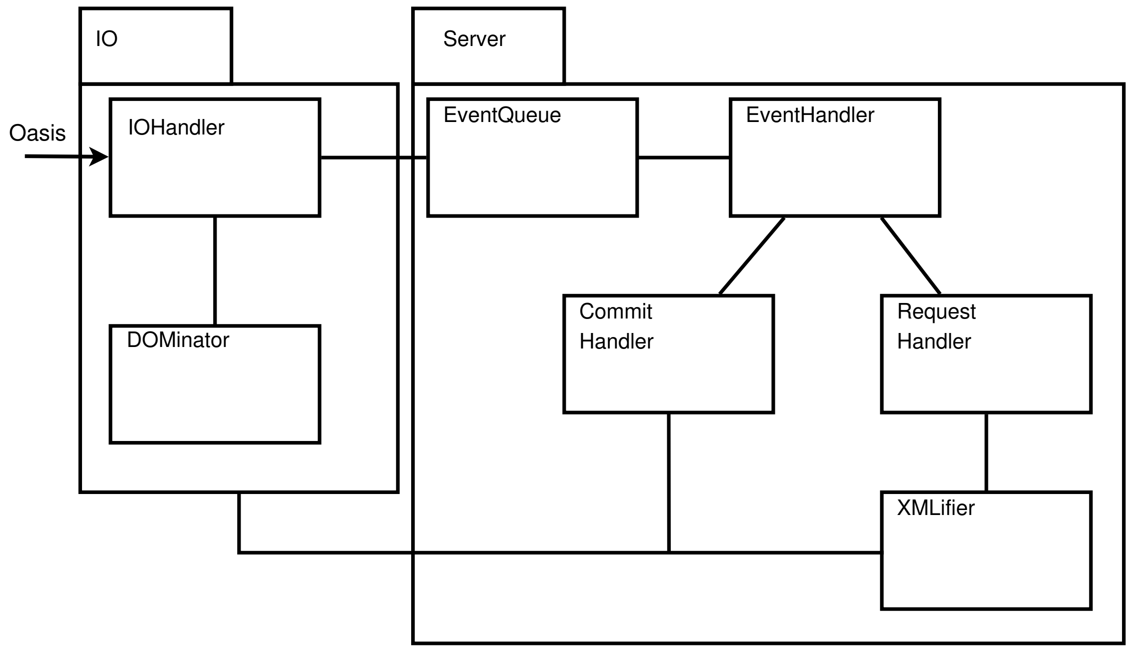


Figure 4.7: Savannnah server Mockup

sw6ml

In order to facilitate consistent data transfer between the global database and localDB, we have designed an XML language which mimics the schema of the database. We have chosen to use XML as it is a recognized standard, with a wide array of tools available, in particular JDOM[23]. JDOM is a light weight implementation of SAX and the Data Object Model(DOM) for java, which allows seamless integration of XML, with support for XPath and XSLT. In section 4.4.1 a short usage documentation for sw6ml is provided. sw6ml is defined with XML schema.

Language Design

The sw6ml language consists of a number of primary elements which reflect the tables in the database, each of these elements accepts any number of ENTRY elements that tell the server what action is should do with the incomming data. in Listing 4.1 a short example of legit sw6ml syntax for adding a row to the AuthUsers table and deleting a row with the idUser table attribute equal to 2 is shown. The `<AuthUsers>..content..</AuthUsers>` element identify the table on which we want to make changes, the following `<Entry>..content..</Entry>` elements define what row and what kind of action, through the `action="foo"` xml attribute, should be done on the row.

```
<AuthUsers>
    <Entry action="create">
       < certificate  type="string">This is a certificate</certificate>
       <idUser type="int">1</idUser>
       <arole type="int">1</arole>
       <username type="string">mette</username>
       <password type="string">obfuscated</password>
    </Entry>
    <Entry action="delete">
       <idUser type="int">2</idUser>
    </Entry>
 </AuthUsers>
```

Listing 4.1: Example of sw6ml syntax

The `action="foo"` xml attribute has four legal types corresponding to the CRUD profile actions, create, read, update, and delete, shown in Listing 4.2. Contained in the `<Entry action="crud_type‘">..content..</Entry>` element is a series of 0 or more elements corresponding to the table schema of, in this case, the AuthUsers table. It takes zero or more of the table attributes in a row, since not all table attributes are needed for all actions, as an example delete only requires the unique identifier of the table and updates will only need the unique identifier and the table attribute being updated.

```
<xs:simpleType name="crud">
 < xs:restriction  base="xs:string">
   <xs:enumeration value="create"/>
   <xs:enumeration value="read"/>
   <xs:enumeration value="update"/>
   <xs:enumeration value="delete"/>
 </ xs:restriction >
</xs:simpleType>
```

Listing 4.2: sw6ml crud simple type

Documentation

To use the current version of sw6ml and savannah together it is essential to know the elements required from the different crud types. while XML schema provides advanced features for dynamic languages, sw6ml int its current version, is a primitive language consisting of simple types and sequences. this is however all that is needed, with a few assertions on the format server side.

Listing 4.3 show the formal structure of a sw6ml document.

```
<sw6ml>
  <table_element_1>
    <Entry action="crud_type">
      <table_element_1_attribute_1/>
        ...
      <table_element_1_attribute_n/>
    </Entry>
  </table_element_n>
  ...
  <table_element_n>
    <Entry action="crud_type">
      <table_element_n_attribute_1/>
        ...
      <table_element_n_attribute_n/>
    </Entry>
  </table_element>
</sw6ml>
```

Listing 4.3: root and table elements

following is a short description of the setup of the `<Entry>` element for each crud type.

create creates a new row in the database: All attributes from the database schema is required, if no value exists, use null. See Listing 4.1 for an example.

update Updates a field in a row, Required in this order: Unique identifier of the row, Attribute to be updated. See Listing 4.4 for an example.

delete Deletes a row in the able: Only the unique identier is required. See Listing 4.1 for an example.

read Read is only used in xml which is send back on a request to the server, and thus require no special formatting.

Notice the `<row_attribute type="bar">..</..>` xml attribute, legal types are `string` or `int`, if the data type of the row attribute is a string or any other type requiring apostrophes in a sql query, use `string`, for anything else use `int`.

```
<Entry action="update">
  <unique_identifier type="bar">foo</unique_identifier>
  <attribute_to_be_updated type="bar">newValue</
      attribute_to_be_updated>
</Entry>
```

Listing 4.4: "sw6ml Update syntax example

The sw6ml schema can be found on the project repository together with a valid sw6ml document, Full path: `http://code.google.com/p/sw6-2012/source/browse/random_group_stuff/Group_server/xml/sw6_schema.xsd` and `http://code.google.com/p/sw6-2012/source/browse/random_group_stuff/Group_server/xml/sw6_example.xml`

4.4.2 Implementation

Input handling

When the server starts it will execute a method called `listen()` that listens for any connections, see Listing 4.5. Whenever any `java.net.Socket` connects to the server, the `listen()` method will create a new `CommunicationThread`. This thread will then read the information in the `Socket`'s `java.io.InputStream` and depending on the connection type (commit event, request event or ping) it will deal with it appropriately.

```
private void listen() {
        try {
                //Initiates  a ServerSocket
                System.out.println("Initiating serversocket !");
                this.serverSocket = new ServerSocket(Configuration.
                    PORT);
                System.out.println("Initiation complete");

        }       catch (IOException io) {
            System.err.println("Could␣not␣create␣ServerSocket␣-_-");
        }
        System.out.println("Starting␣to␣listen␣!");
        //Loop that continues to look & accept connections
        //on the ServerSocket
        while (true) {
            try {
                //Making a connections
                Socket con = this.serverSocket.accept();
                System.out.println("Connection␣accepted␣-␣Socket:␣" + con);

                //Saving the connections and a corresponding
                //DataOutputStream for later use
                this.connections.put(con, new DataOutputStream(con.getOutputStream()));

                //Starts a new Thread used for communication
                Thread comThread = new CommunicationThread(con, this.folder);
                comThread.start();

            }       catch (IOException e) {
                System.out.println("Could␣not␣accept␣connection␣!");
            }
        }
}
```

Listing 4.5: Method: `listen()`

In the following we will explain how the different types of connections are processed by the system.

Ping

Diagram ledger: (see Figure 4.8)
The uppermost port is used to indicate input and the lowermost port is used

to indicate output. The arrows indicate which way the information is forwarded in the system. Any arrows leading to and fro a component, and not from an input or output port, are to be executed first.

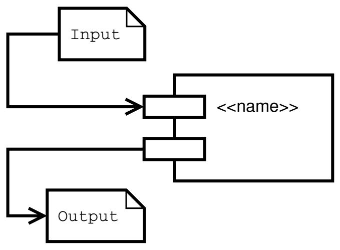


Figure 4.8: Ledger for diagrams

Figure 4.9 is a model of how a ping is process by the server. The `Connection` component sends output, in this case a ping event, to the server. In the server this is received by the `IOHandler`, which creates a new `CommunicationThread`. The `CommunicationThread` will then use the `TransmissionHandler` to determine if the input is of the type commit, request or ping. In this case the input is of the type ping, and it will just directly respond to the caller (connection). More information on these classes can be found in Figure A.2, Figure A.3, and Figure A.4.

The reason for this implementation is that Java only supports two types of sockets: stream based (TCP – `java.net.Socket` and `java.net.ServerSocket`) and datagram based ones (UDP – `java.net.DatagramSocket` and `java.net.MulticastSocket`)[15][8]. However an implementation of ping would require the use of ICMP(Internet Control Message Protocol) ping, and since this is not possible we have implemted our own ping-like function.

Commit and Request

For a connection of type commit or request, the procedure is almost the same. However, when the `CommunicationThread` has been created and its `TransmissionHandler` has determine the type of the connection, it will opposite for the ping, create a new event corresponding to the type and send it to the `EventHandler`, see Figure 4.10. First when the `EventHandler` is done processing the event, will the server respond to the connector. To see how the `EventHandler` processes the given queries see section 4.4.2.

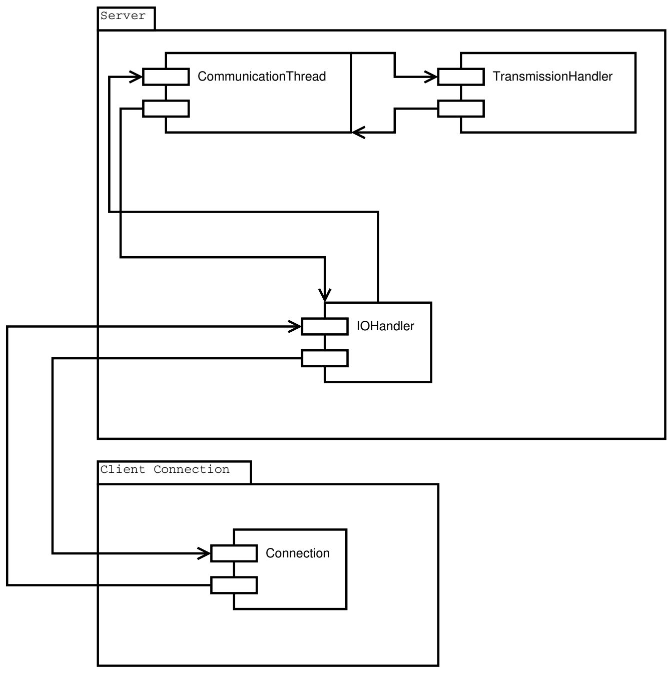


Figure 4.9: A diagram illustrating how a ping is processed.

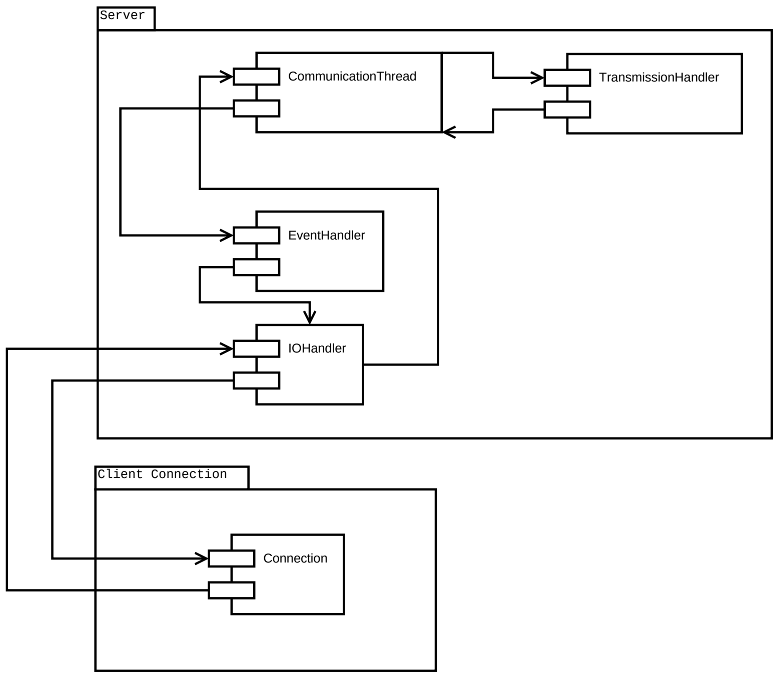


Figure 4.10: A diagram illustrating how a request or a commit event is processed.

Queue and Query handling

4.4.3 Test

JUnit

Other test

CHAPTER 5

RECAPITULATION

5.1 Conclusion

5.2 Future work

5.3 Multi project

APPENDIX A

APPENDIX

A.1 MySQL code

```

CREATE TABLE `04`.`Apps` (

`idApp` INT NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL ,

`version` VARCHAR(45) NOT NULL ,

`icon` VARCHAR(45) NOT NULL,

`package` VARCHAR(45) NOT NULL,

`activity` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idApp`) );
```

Listing A.2: Create Apps

```
CREATE TABLE `04`.`Tags` (
  CREATE TABLE ‘04‘.‘AuthUsers‘ (

  ‘certificate‘ VARCHAR(512) NOT NULL ,

  ‘idUser‘ INT NOT NULL AUTO_INCREMENT ,

  ‘aRole‘ INT NOT NULL,

  ‘username‘ VARCHAR(45) UNIQUE NOT NULL,

  ‘password‘ VARCHAR(45) NOT NULL,

  PRIMARY KEY (‘certificate‘) ,

  UNIQUE INDEX ‘idUser_UNIQUE‘ (‘idUser‘ ASC) );
```

Listing A.1: Create AuthUsers

```

  ‘idTags‘ INT NOT NULL AUTO_INCREMENT,

  ‘caption‘ VARCHAR(45) NOT NULL ,

  PRIMARY KEY (‘idTags‘) ,

  UNIQUE INDEX ‘caption_UNIQUE‘ (‘caption‘ ASC) );
```

Listing A.3: Create Tags

```
CREATE TABLE ‘04‘.‘Profile‘ (

  ‘idProfile‘ INT NOT NULL ,

  ‘firstname‘ VARCHAR(45) NOT NULL ,

  ‘surname‘ VARCHAR(45) NOT NULL ,

  ‘middlename‘ VARCHAR(45) NULL ,

  `pRole` INT NOT NULL ,

  `phone` BIGINT NOT NULL ,

  `picture` VARCHAR(45) NULL ,

  `settings` BLOB NULL ,

  FOREIGN KEY (`idProfile` )

  REFERENCES `04`.`AuthUsers` (`idUser` ) on delete cascade,

  PRIMARY KEY (`idProfile`) );
```

Listing A.4: Create Profile

```
CREATE TABLE `04`.`ListOfApps` (

  `idApp` INT NOT NULL ,

  `idProfile` INT NOT NULL ,

  `setting` BLOB,

  `stats` BLOB,

  FOREIGN KEY (`idApp` )

  REFERENCES `04`.`Apps` (`idApp` ),

  FOREIGN KEY (`idProfile` )

  REFERENCES `04`.`Profile` (`idProfile` ) ON DELETE CASCADE,

  PRIMARY KEY (`idApp`,`idProfile`) );
```

Listing A.5: Create ListOfApps

```
CREATE TABLE ‘04‘.‘Department‘ (

  ‘idDepartment‘ INT NOT NULL ,

  ‘name‘ VARCHAR(45) NOT NULL ,

  ‘address‘ VARCHAR(45) NOT NULL ,

  ‘phone‘ BIGINT NOT NULL ,

  ‘email‘ VARCHAR(45) NOT NULL ,

  FOREIGN KEY (‘idDepartment‘ )

  REFERENCES ‘04‘.‘AuthUsers‘ (‘idUser‘ ),

  PRIMARY KEY (‘idDepartment‘) );
```

Listing A.6: Create Department

```
CREATE TABLE ‘04‘.‘HasDepartment‘ (

  ‘idProfile‘ INT NOT NULL ,

  ‘idDepartment‘ INT NOT NULL ,

  FOREIGN KEY (‘idProfile‘ )

  REFERENCES ‘04‘.‘Profile‘ (‘idProfile‘ ) ON DELETE CASCADE,

  FOREIGN KEY (‘idDepartment‘ )

  REFERENCES ‘04‘.‘Department‘ (‘idDepartment‘ ),


  PRIMARY KEY (‘idProfile‘, ‘idDepartment‘) );
```

Listing A.7: Create HasDepartment

```
CREATE TABLE `04`.`HasGuardian` (

  `idGuardian` INT NOT NULL ,

  `idChild` INT NOT NULL ,

  FOREIGN KEY (`idGuardian` )

  REFERENCES `04`.`Profile` (`idProfile` ) on delete cascade,

  FOREIGN KEY (`idChild` )

  REFERENCES `04`.`Profile` (`idProfile` ) on delete cascade,

  PRIMARY KEY (`idGuardian`,`idChild`) );
```

Listing A.8: Create HasGuardian

```
CREATE TABLE `04`.`HasSubDepartment` (

  `idDepartment` INT NOT NULL ,

  `idSubDepartment` INT NOT NULL ,

  FOREIGN KEY (`idDepartment` )

  REFERENCES `04`.`Department` (`idDepartment` ),

  FOREIGN KEY (`idSubDepartment` )

  REFERENCES `04`.`Department` (`idDepartment` ),

  PRIMARY KEY (`idDepartment`, `idSubDepartment`) );
```

Listing A.9: Create HasSubDepartment

```
CREATE TABLE `04`.`Media` (

  `idMedia` INT NOT NULL AUTO_INCREMENT,

  `mPath` VARCHAR(45) NOT NULL ,

  `name` VARCHAR(45) NOT NULL ,

  `mPublic` TINYINT NOT NULL ,

  `mType` VARCHAR(45) NOT NULL ,

  `ownerID` INT NOT NULL ,

  FOREIGN KEY (`OwnerID` )

  REFERENCES `04`.`AuthUsers` (`idUser` )  ON DELETE CASCADE,

  PRIMARY KEY (`idMedia`) );
```

Listing A.10: Create Media

```
CREATE TABLE `04`.`HasTag` (

  `idMedia` INT NOT NULL ,

  `idTag` INT NOT NULL ,

  FOREIGN KEY (`idMedia` )

  REFERENCES `04`.`Media` (`idMedia` ) on delete cascade,

  FOREIGN KEY (`idTag` )

  REFERENCES `04`.`Tags` (`idTags` ),

  PRIMARY KEY (`idMedia`, `idTag`) );
```

Listing A.11: Create HasTag

```
CREATE TABLE ‘04‘.‘HasLink‘ (

  ‘idMedia‘ INT NOT NULL ,

  ‘idSubMedia‘ INT NOT NULL ,

  FOREIGN KEY (‘idMedia‘ )

  REFERENCES ‘04‘.‘Media‘ (‘idMedia‘ ) on delete cascade,

  FOREIGN KEY (‘idSubMedia‘ )

  REFERENCES ‘04‘.‘Media‘ (‘idMedia‘ ) on delete cascade,

  PRIMARY KEY (‘idMedia‘, ‘idSubMedia‘) );
```

Listing A.12: Create HasLink

```
CREATE TABLE ‘04‘.‘MediaProfileAccess‘ (

  ‘idProfile‘ INT NOT NULL ,

  ‘idMedia‘ INT NOT NULL ,

  FOREIGN KEY (‘idProfile‘ )

  REFERENCES ‘04‘.‘Profile‘ (‘idProfile‘ ) ON DELETE CASCADE,

  FOREIGN KEY (‘idMedia‘ )

  REFERENCES ‘04‘.‘Media‘ (‘idMedia‘ ) ON DELETE CASCADE,

  PRIMARY KEY (‘idProfile‘, ‘idMedia‘) );
```

Listing A.13: Create MediaProfileAccess

```
CREATE TABLE ‘04‘.‘MediaDepartmentAccess‘ (

  ‘idDepartment‘ INT NOT NULL ,

‘idMedia‘ INT NOT NULL ,

FOREIGN KEY (‘idDepartment‘ )

REFERENCES ‘04‘.‘Department‘ (‘idDepartment‘ ),

FOREIGN KEY (‘idMedia‘ )

REFERENCES ‘04‘.‘Media‘ (‘idMedia‘ ),

PRIMARY KEY (‘idDepartment‘, ‘idMedia‘) );
```

Listing A.14: Create MediaDepartmentAccess

A.2 ERR diagram

A.3 Class-diagrams

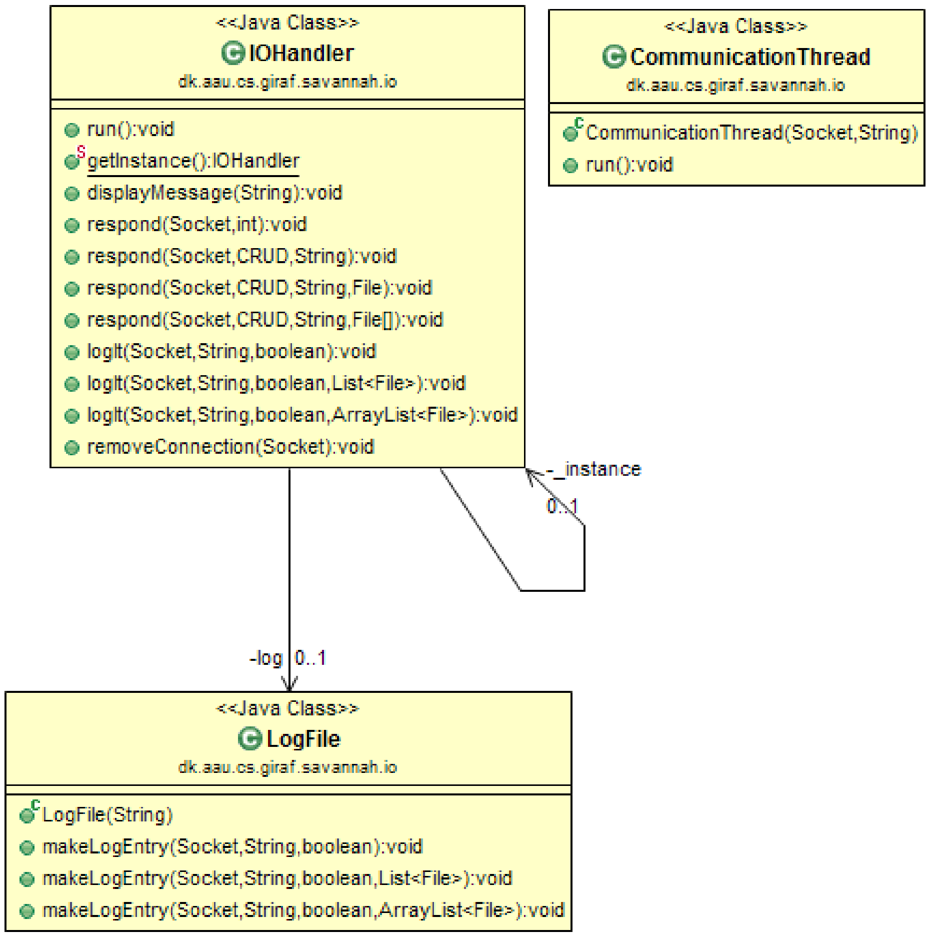

<<Java Class>>
IOHandler
dk.aau.cs.giraf.savannah.io
run():void
getInstance():IOHandler
displayMessage(String):void
respond(Socket,int):void
respond(Socket,CRUD,String):void
respond(Socket,CRUD,String,File):void
respond(Socket,CRUD,String,File[]):void
logIt(Socket,String,boolean):void
logIt(Socket,String,boolean,List<File>):void
logIt(Socket,String,boolean,ArrayList<File>):void
removeConnection(Socket):void
<<Java Class>>
CommunicationThread
dk.aau.cs.giraf.savannah.io
CommunicationThread(Socket,String)
run():void
-_instance
0..1
-log
0..1
<<Java Class>>
LogFile
dk.aau.cs.giraf.savannah.io
LogFile(String)
makeLogEntry(Socket,String,boolean):void
makeLogEntry(Socket,String,boolean,List<File>):void
makeLogEntry(Socket,String,boolean,ArrayList<File>):void

Figure A.2: Class-diagram of the `IOHandler` class

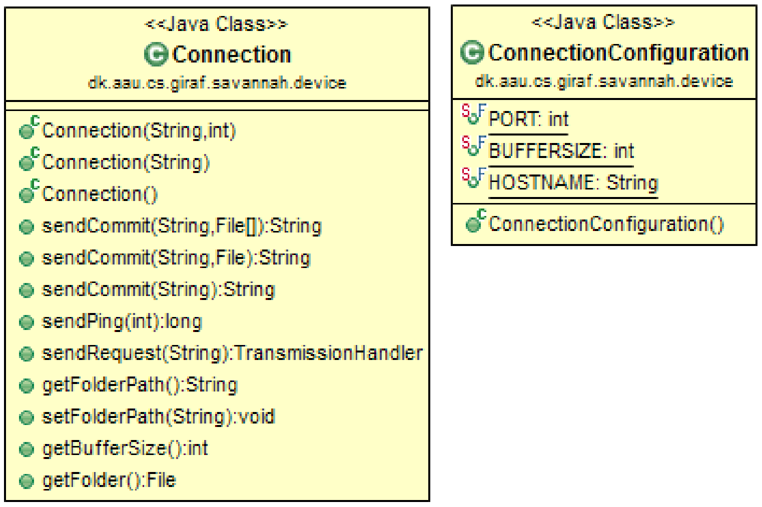

<<Java Class>>
Connection
dk.aau.cs.giraf.savannah.device
Connection(String,int)
Connection(String)
Connection()
sendCommit(String,File[]):String
sendCommit(String,File):String
sendCommit(String):String
sendPing(int):long
sendRequest(String):TransmissionHandler
getFolderPath():String
setFolderPath(String):void
getBufferSize():int
getFolder():File
<<Java Class>>
ConnectionConfiguration
dk.aau.cs.giraf.savannah.device
PORT: int
BUFFERSIZE: int
HOSTNAME: String
ConnectionConfiguration()

Figure A.3: Class-diagram of the `Connection` class

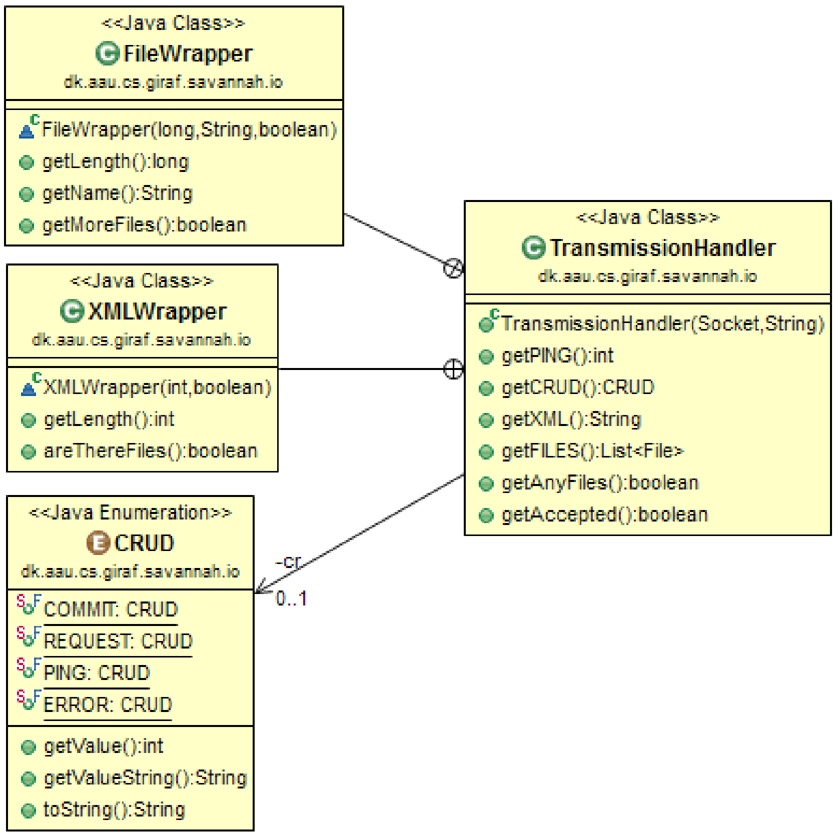


Figure A.4: Class-diagram of the `TransmissionHandler` class

A.4 Notes from Interview

This is notes from an interview with Mette Als Andreasen, an educator at Birken in Langholt, Denmark.

Når tiden løber ud (kristian har tage et billede):
Færdig - symbol
Gå til skema - symbol
Taget fra boardmaker

Kunne være godt hvis man kunne sætte egne billeder ind som start/stop symboler.

Rød farve = nej, stop, aflyst.

De har sådan et ur på 60 minutter hvor tid tilbage er markeret med rød, og så bipper den lige kort når den er færdig.
Det ville være fint hvis de kunne bruge sort/hvid til dem der ikke kan håndtere farver, men også kan vælge farver.

Stop-ur:
en fast timer på 60 minutter + en customizable som ikke ser helt magen til ud, som f.eks, kan være på 5, 10 eller 15 minutter for en hel cirkel.

timeglas:
skift farve på timeglassene, men ikke nødvendigvis gøre dem større. Kombinere med mere/mindre sand. Eventuelt kombinere med et lille digitalt ur, til dem der har brug for det, skal kunne slåes til og fra.

Dags-plan:
ikke særlig relevant til de helt små og ikke særligt velfungerende børn. Men kunne være rigtig godt til de lidt ældre.
En plan går oppefra og ned, og hvis der så skal specificeres noget ud til aktiviteterne, så er det fra venstre mod højre ud fra det nedadgående skema.

Til parrot:
Godt med rigtige billeder af tingene, som pædagogerne selv kan tage, eventuelt også af aktiviteter, så pedagogerne kan have billeder af aktiviter som de kan liste efter skeamet.

Der var mange skemaer rundt omkring, og der henviser det sidste billede i rækken til næste skema, som hænger f.eks. på badeværelset eller i garderoben.

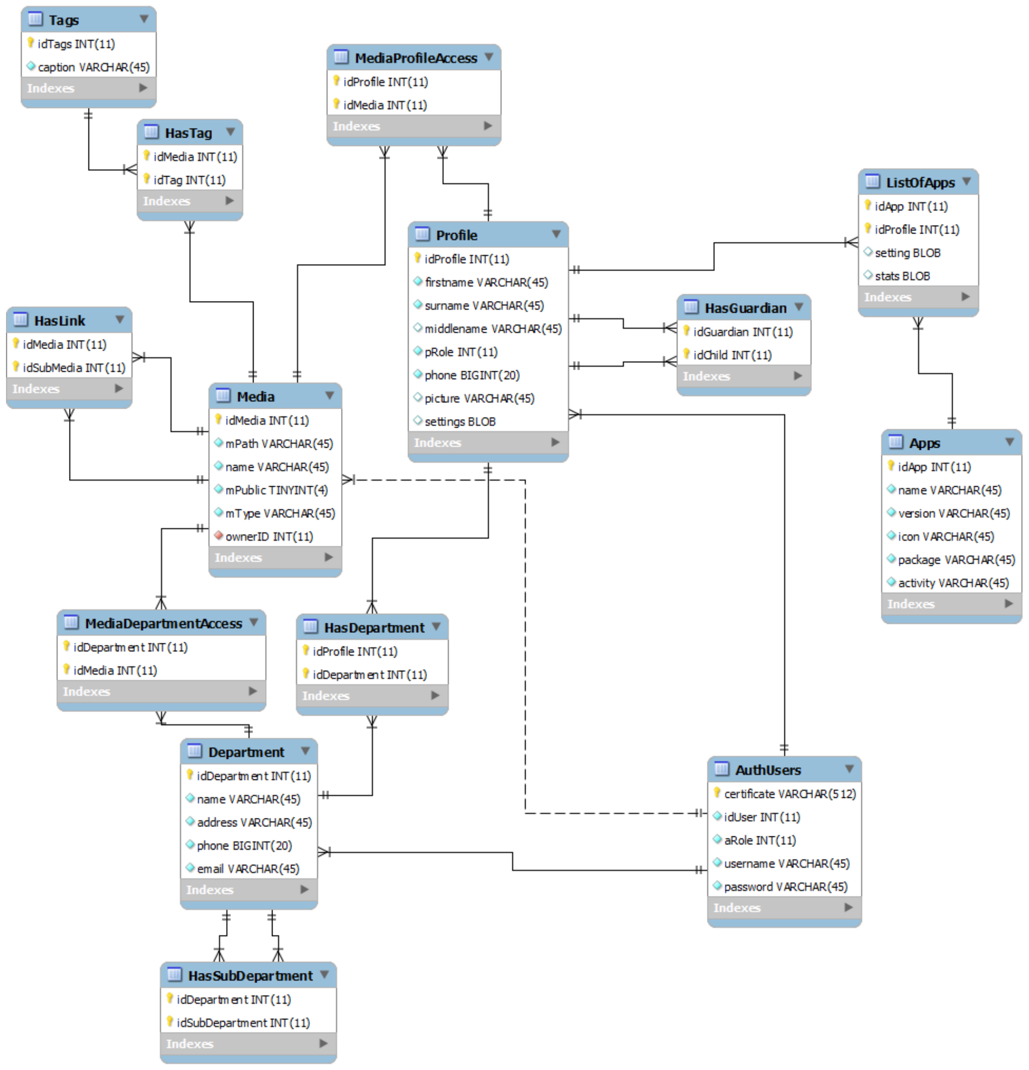
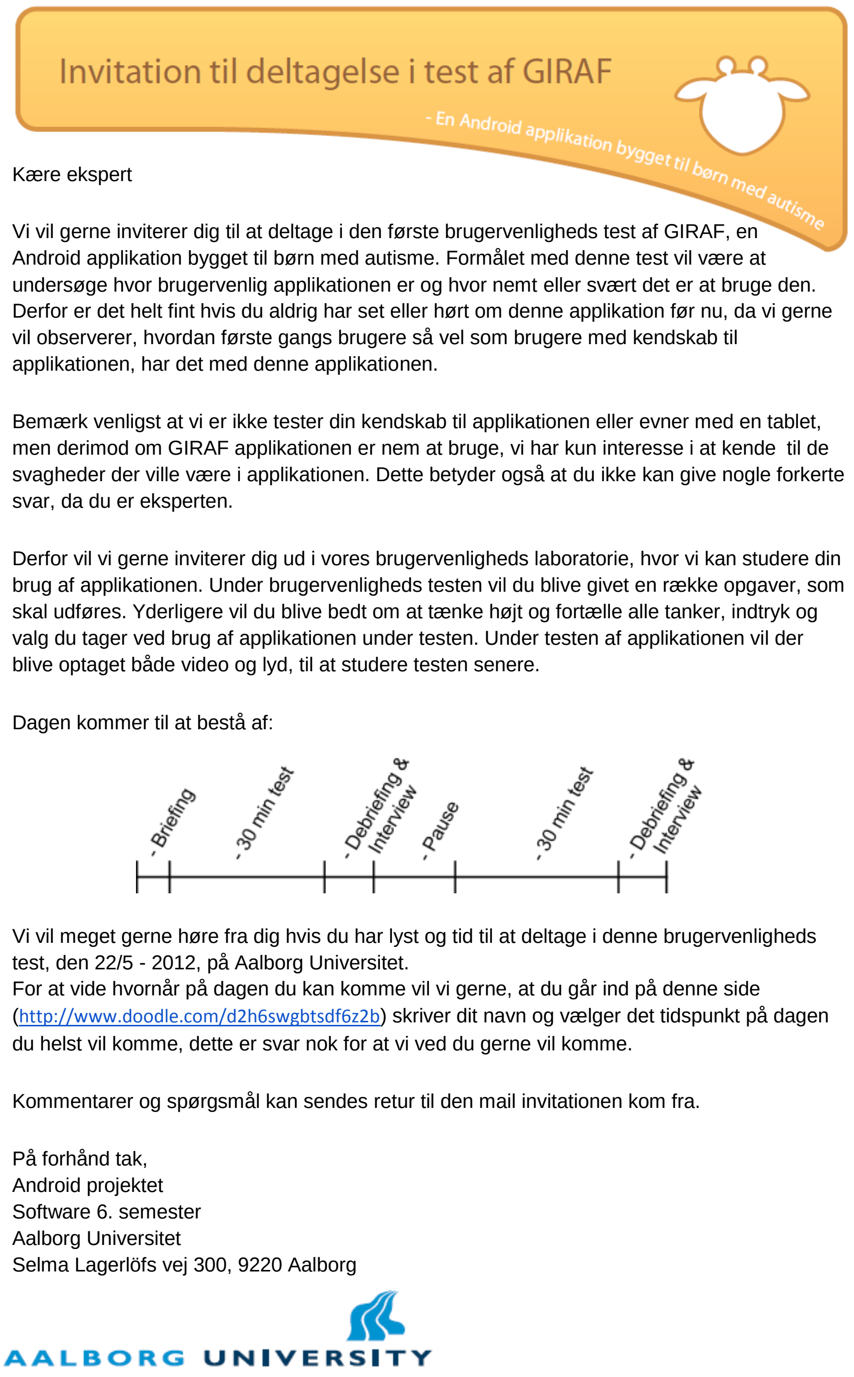


Figure A.1: The ERR diagram as MySQL Workbench generates it



Invitation til deltagelse i test af GIRAF

- En Android applikation bygget til børn med autisme

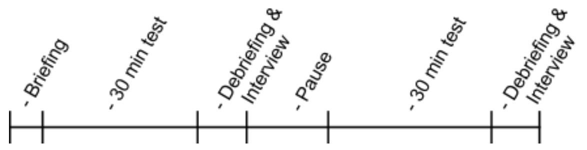
Kære ekspert

Vi vil gerne inviterer dig til at deltage i den første brugervenligheds test af GIRAF, en Android applikation bygget til børn med autisme. Formålet med denne test vil være at undersøge hvor brugervenlig applikationen er og hvor nemt eller svært det er at bruge den. Derfor er det helt fint hvis du aldrig har set eller hørt om denne applikation før nu, da vi gerne vil observerer, hvordan første gangs brugere så vel som brugere med kendskab til applikationen, har det med denne applikationen.

Bemærk venligst at vi er ikke tester din kendskab til applikationen eller evner med en tablet, men derimod om GIRAF applikationen er nem at bruge, vi har kun interesse i at kende til de svagheder der ville være i applikationen. Dette betyder også at du ikke kan give nogle forkerte svar, da du er eksperten.

Derfor vil vi gerne inviterer dig ud i vores brugervenligheds laboratorie, hvor vi kan studere din brug af applikationen. Under brugervenligheds testen vil du blive givet en række opgaver, som skal udføres. Yderligere vil du blive bedt om at tænke højt og fortælle alle tanker, indtryk og valg du tager ved brug af applikationen under testen. Under testen af applikationen vil der blive optaget både video og lyd, til at studere testen senere.

Dagen kommer til at bestå af:



Vi vil meget gerne høre fra dig hvis du har lyst og tid til at deltage i denne brugervenligheds test, den 22/5 - 2012, på Aalborg Universitet.
For at vide hvornår på dagen du kan komme vil vi gerne, at du går ind på denne side (http://www.doodle.com/d2h6swgbtsdf6z2b) skriver dit navn og vælger det tidspunkt på dagen du helst vil komme, dette er svar nok for at vi ved du gerne vil komme.

Kommentarer og spørgsmål kan sendes retur til den mail invitationen kom fra.

På forhånd tak,
Android projektet
Software 6. semester
Aalborg Universitet
Selma Lagerlöfs vej 300, 9220 Aalborg

AALBORG UNIVERSITY

Figure A.5: Invitation sent to the test persons of the usability test.

BIBLIOGRAPHY

[1] Scrum Alliance. Scrum alliance, 2011. URL http://www.scrumalliance.org/.

[2] Scrum Alliance. Advice on conducting the scrum of scrums meeting, 2011. URL http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting.

[3] D. Robinson, K. Coar. The Common Gateway Interface (CGI) Version 1.1. http://tools.ietf.org/html/rfc3875#section-1, October 2004. [Online; accessed 29-May-2012].

[4] Temple Grandin. Teaching tips for children and adults with autism, December 2002. URL http://www.autism.com/ind_teaching_tips.asp.

[5] Hans Bergsten. An Introduction to Java Servlets. http://www.developer.com/java/an-introduction-to-java-servlets-.html, 18. May 2000. [Online; accessed 17-May-2012].

[6] Jason Hunter. *Java Servlet Programming.* O'Reilly Media, 1998. ISBN 156592391X.

[7] Steffan Bo Pallesen Jacob Bang, Lasse Linnerup Christiansen. Administration module for giraf, May 2011. URL http://people.cs.aau.dk/~ulrik/Giraf/Admin.pdf.

[8] Java^{TM}Platform, Standard Edition 6. Overview (Java Platform SE 6. `http://docs.oracle.com/javase/6/docs/api/overview-summary.html`, 09. January 2012. [Online; accessed 20-May-2012].

[9] Jan Stage Jesper Kjeldskov, Mikael B. Skov. *Instant Data Analysis: Conducting Usability Evaluations in a Day.* ISBN 1-58113-857-1. Last viewed: 2012-05-24.

[10] live.gnome.org. Dia homepage. `https://live.gnome.org/Dia`. [Online; accessed 23-May-2012].

[11] Mehdi Achour, Friedhelm Betz, Antony Dovgal,Nuno Lopes, et. al. What can PHP do? `http://www.php.net/manual/en/intro-whatcando.php`, 11. May 2012. [Online; accessed 17-May-2012].

[12] Mehdi Achour, Friedhelm Betz, Antony Dovgal,Nuno Lopes, et. al. History of PHP - PHP Tools, FI, Construction Kit, and PHP/FI. `http://www.php.net/manual/en/history.php.php`, 11. May 2012. [Online; accessed 17-May-2012].

[13] Mehdi Achour, Friedhelm Betz, Antony Dovgal,Nuno Lopes, et. al. What can PHP do? `http://www.php.net/manual/en/tutorial.requirements.php`, 11. May 2012. [Online; accessed 17-May-2012].

[14] Microsoft. Terms of use. `http://www.asp.net/terms-of-use`, 2. November 2010. [Online; accessed 17-May-2012].

[15] Network Sorcery, Inc. ICMP, Internet Control Message Protocol. `http://www.networksorcery.com/enp/protocol/icmp.htm`, 20. April 2010. [Online; accessed 20-May-2012].

[16] Samsung. Samsung tablet. URL `http://www.samsung.com/global/microsite/galaxytab/10.1/index.html`.

[17] Scott Trent, Michiaki Tatsubori, Toyotaro Suzumura, Akihiko Tozawa and Tamiya Onodera. Performance Comparison of PHP and JSP as Server-Side Scripting Languages. `http://www.research.ibm.com/trl/people/mich/pub/200812_middleware2008specweb.pdf`, 2008. [Online; accessed 17-May-2012].

[18] Allen G. Taylor. *SQL For Dummies (For Dummies (Computers))*. For Dummies, 2007. ISBN 978-0-470-04652-4. URL `http://www.amazon.com/SQL-For-Dummies-Computers-ebook/dp/B003S9VTFU%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB003S9VTFU`.

[19] Android Development Team. Android 4.0.3 platform, 2012. URL `http://developer.android.com/sdk/android-4.0.3.html#relnotes`.

[20] Aalborg University. Studieordning for bacheloruddannelsen i software. *URL: `http://www.sict.aau.dk/digitalAssets/3/3331_softwbach_sept2009.pdf`*. Last viewed: 2012-03-15.

[21] Don Wells. extreme programming, 2009. URL `http://www.extremeprogramming.org/rules.html`.

[22] Wikipedia - the free encyclopedia. Java Servlet. `http://en.wikipedia.org/wiki/Java_Servlet`, 11. May 2012. [Online; accessed 17-May-2012].

[23] www.jdom.org. Jdom hompage. `http://www.jdom.org`. [Online; accessed 10-February-2012].