AALBORG UNIVERSITY

STUDENT REPORT

GROUP SW604F12

BACHELOR PROJECT

Oasis: Part of the GIRAF System

Authors:
Henrik KLARUP
Jens Mohr MORTENSEN
Dan Stenholt MØLLER

Supervisor:
Ulrik NYMAN

Spring Semester 2012

AALBORG UNIVERSITY

STUDENT REPORT

Title: Oasis: Part of the GIRAF System
Subject: Application Development
Semester: Spring Semester 2012
Project group: sw604f12

Participants:

Henrik Klarup

Jens Mohr Mortensen

Dan Stenholt Møller

Supervisor:
Ulrik Nyman

Number of copies: X

Number of pages: X

Number of appendices: X Pages

Completed: June 3, 2012

Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Øst
Telephone +45 9940 9940
Telefax +45 9940 9798
http://cs.aau.dk

Synopsis:

Oasis is an administration module for the GIRAF system. The GIRAF system is designed to ease the daily routine for guardians of children with ASD.
Oasis will provide tools for the development of applications for the GIRAF platform. Oasis consists of three parts; a database used to store information, a library that supplies an API, and an administration application. We tested the functionality of Oasis, using unit testing and a usability test.
The Oasis tools are used in the development of various applications for the GIRAF platform. Even though we released a working version of Oasis, the tools are still open for further development.

Resume

Oasis is an administration module for the GIRAF system. The GIRAF system is developed as part of a multi project. The multi project is a continuation of a previous multi project by a group of 6th semester Software students at Aalborg University.

Besides Oasis, other groups have developed modules for the GIRAF system. These includes; Launcher, WOMBAT, PARROT, and Savannah. Oasis have received requirements from different multi project groups. These requirements specifies how the Oasis module is structured.

Oasis consists of three parts; a database, a library, and a Oasis administration app. The database part of Oasis is called Oasis Local Db. The Oasis Local Db is used for storing the data, which is used by GIRAF applications. The library of Oasis is called Oasis Lib. The Oasis Lib is the library, which works as a connection between the Oasis Local Db and GIRAF applications. The administration application is called Oasis App. The Oasis App demonstrates some of the utilities the Oasis Lib offers.

To ensure the correctness of the Oasis Lib we enforced dynamic white box testing through unit tests. The unit tests are created in order to test the Oasis Lib and the Oasis Local Db. It is only the normal operation of the Oasis Lib and Oasis Local Db that has been tested. The usability of the Oasis App has been tested and the result of the test have been documented.

The Oasis tools are used in the development of various applications for the GIRAF platform. Although we released a fully functional version of Oasis, there still exists some requirements Oasis does not fulfill. These requirements gives the opportunity for further development.

Preface

This report was written as a 6th semester bachelor project by Software group sw604f12 from the Department of Computer Science at Aalborg University in the spring of 2012. The report will document the process of developing an administration module for the GIRAF system. The GIRAF system is designed to support guardians of children with autism spectrum disorder.

When references are used in the report they will be referred to in the format [abc] with a corresponding entry in the bibliography (located in the back of the report). For larger works a range of relevant pages will also be specified in the format [Abc, pp10]. Figures, tables and equations will be referred to in the format [2.1], where the first number refers to the chapter in which it is placed and the second number is the actual number of the figure, table or equation.

The reader is expected to have an understanding of basic programming concepts.

Throughout this report the following abbreviations will be used:

- GIRAF - Name of the entire system, a title was never agreed upon, but "**G**raphical **I**nterface **R**esources for **A**ssistive **F**unctionality" was suggested.
- WOMBAT - Name of the timer application, **W**ay **O**f **M**easuring **Ba**sic **T**ime.
- PARROT - Name of the pictogram application, **P**ictogram **A**ssisting with **R**hetoric **R**easoning **O**r **T**alking.
- MVC - **M**odel **V**iew **C**ontroller [Mic12].
- ASD - **A**utistic **S**pectrum **D**isorder.

- Guardians - Parents, teachers, caretakers, and educators of children with ASD.
- Children - Children refers to “Children with ASD”.
- We - In the introduction, it refers to the entire multi project group. After the common report, it refers to the individual project group.
- Eclipse - An intelligent development environment for Java.
- Metadata - Data about data.
- API - **A**pplication **P**rogramming **I**nterface.
- GUI - **G**raphical **U**ser **I**nterface

The CD-ROM included with this report contains the source code for the system developed during this project, as well as a copy of this report.

The Oasis administration project group would like to thank the educators whom helped developing and designing the GIRAF system, as well as our supervisor for the interest and cooperation in the project.

Contents

*

Part I

GIRAF Introduction

CHAPTER 1

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.

1.1 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 [Uni].

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.

1.2 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.

1.2.1 Working with Children with ASD

This section is based upon the statements of a woman with ASD [Gra02], explaining what it is like to live with ASD, and an interview with an educator at Birken, a special kindergarten for children (see section 11.5 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 institution."*
>
> - Drazenko Banjak, educator at Egebakken.

1.3 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[Sam]. 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. [Tea12a]

1.4 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) [Wel09], and Scrum [All11b].

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 [All11a].

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.

1.5 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.

1.6 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.

1.7 Architecture

Our System architecture – shown in Figure 1.1 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 [JB11] to make it easier to work with. We have simplified the architecture because we feel it is unnecessarily complex.

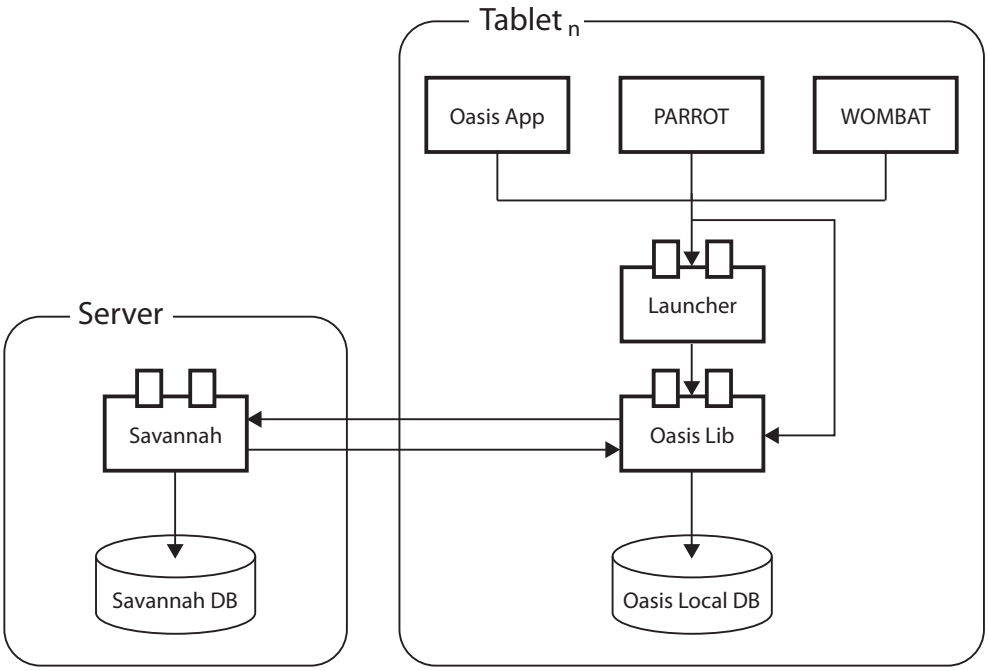


Figure 1.1: The GIRAF architecture

1.8 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 cur-

rent usability of the GIRAF platform as a whole, as well as of the individual parts of the platform. Furthermore, the next wave of developers will immediately be able to start correcting the found usability issues.

1.8.1 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 11.11.

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. [JK]

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 1.2.

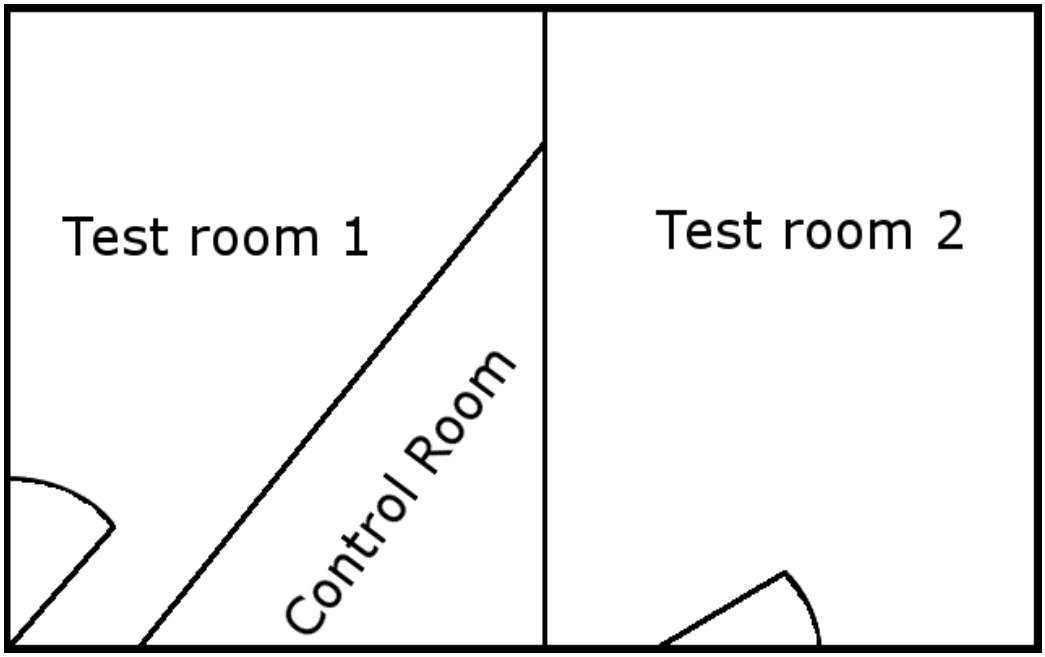


Figure 1.2: An overview of the usability lab at Cassiopeia, Department of Computer Science, Aalborg University.

Execution

The tests are conducted according to the schedule in Figure 1.3.

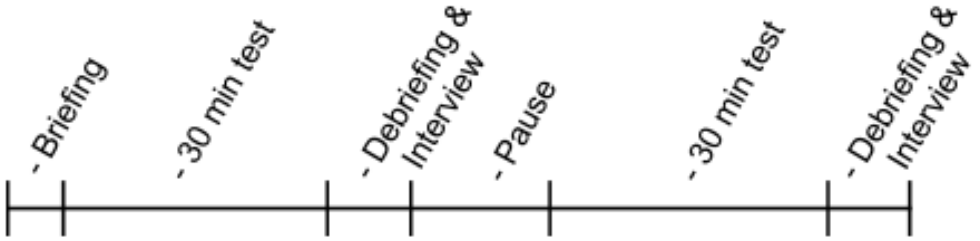


Figure 1.3: The schedule of the usability test.

Briefing, debriefing, and questionnaire documents can be found in section 11.6, and the results of the test can be found in subsection 7.2.1.

Part II

Oasis Introduction

CHAPTER 2

Initial Process

In this chapter the problem definition for Oasis is described. An analysis of the requirements, received from the other groups in the multi project, are performed and from the analysis the Oasis system architecture has been derived.

2.1 Problem Definition

In the multi project a common problem definition have been stated, see Section 1.5 on page 5. The common problem definition is – as stated – to vogue for the individual groups. Therefore we have devised a problem definition for Oasis. It is as follows:

> *How can we provide a set of tools, which can help develop applications for the GIRAF system?*

2.2 Requirements

Before the development can begin the requirements for the software must be analyzed. The requirements stem from the multi project groups and the contact persons. Examples of the requirements received can be seen in Appendix 11 on page 71 and 11.4 on page 86.

The derived informal requirements are:

- Child profile handling
- Guardian profile handling
- Child and guardian relation handling
- Application specific profile setting handling
- Certificate handling
- Media handling
- Media access handling
- Media to media relation handling
- Department handling
- Department to subdepartment handling
- Profile to department relation handling

We have decided to develop a database and a library, which are to be used by GIRAF applications. As an example of a GIRAF application, that uses the library and database, we will develop an administration application.

2.3 System architecture

Oasis consists of three parts; The Oasis Local Db, the Oasis Lib, and the Oasis App. This is shown in Figure 2.1 on the next page.

The Oasis Lib handles application interaction with the Oasis Local Db, and every GIRAF application should be utilizing this library. Another feature the Oasis Lib offers is the ability to handle synchronization between the Oasis Local Db and Savannah.

The last part is the Oasis App. This application allows interaction with the Oasis Local Db through the Oasis Lib. Some of the features of the Oasis App are creation and removal of profiles and departments. It also enables the user to handle relations between specific elements.

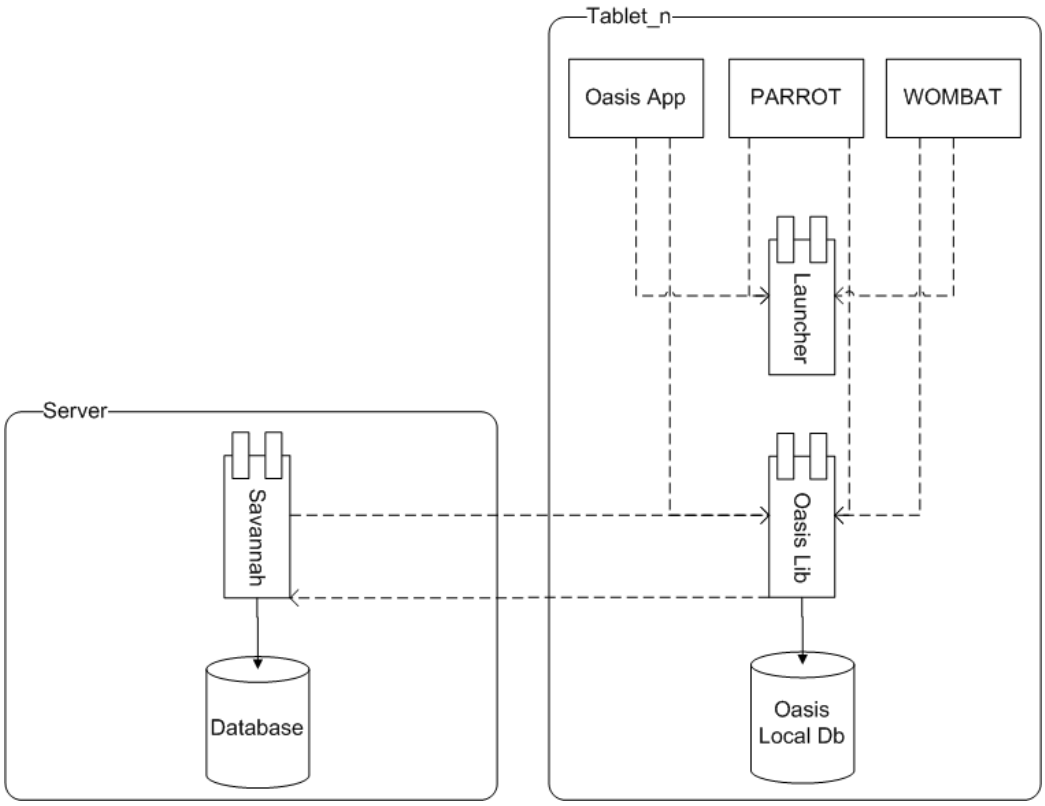


Figure 2.1: The GIRAF sytem architecture.

CHAPTER 3

The Process

In this chapter we describe the sprint process of our project, this can be seen in Section 3.1.

3.1 Sprints

This project was divided into eight sprints. The length of the sprints was decided at every sprint meeting. At the end of each sprint period, a meeting was held to reflect on what was learned through the sprint. The burn down charts, sprint backlogs and the final change log can be seen in Appendix 11.2 on page 74.

3.1.1 Sprint One

Sprint one started on 19/03-2012 with a work period of seven half days. In this sprint period we started making the design for the Oasis Local Db schema, a prototype database, the model classes, that is supposed to wrap the data from the Oasis Local Db, and some methods to the library, which we got as requirements from the other groups.

Sprint Reflection

In this sprint we learned that we should be a bit better at estimating the size of each task and the solution would be combining the small tasks into bigger

tasks, thereby allowing us to work on one big task a day, instead of several small tasks.

3.1.2 Sprint Two

Sprint two started on 26/03-2012 with a work period of eight half days. In this sprint period we improved the design for the Oasis Local Db schema, started implementing the Oasis Local Db, and updated the library with the new requirements we got from the other groups. Besides that we used a little time on setting up a counter-strike server, this gave us a social activity that improved the relation between the multi project groups.

Sprint Reflection

In this sprint we learned that we still should be a bit better at estimating the size of each task, because there was some tasks we did not start on. Another thing we learned was that a good database design takes some iteration to achieve.

3.1.3 Sprint Three

Sprint three started on 10/04-2012 with a work period of nine half days. In this sprint period we froze the Oasis Local Db schema, continued on the implementation of the Oasis Local Db, and updated the Oasis Lib with the new requirements gathered from the other groups. Besides that we started thinking on the "Oasis App".

Sprint Reflection

In this sprint we learned that we should not be adding new tasks to the sprint backlog when the sprint has started, as this will make us less likely to complete all tasks on the sprint backlog. Besides that, we underestimated the size of the tasks, which lead to tasks not being completed before the end of the sprint. Another thing we learned was that some tasks depended on each other and this could result in some tasks not being completed because of such a dependency.

3.1.4 Sprint Four

Sprint four started on 23/04-2012 with a work period of six half days. In this sprint period we completed the implementation of the Oasis Local Db and updated the Oasis Lib with the new requirements gathered from the other groups. Besides that we started writing Java Doc to the Oasis Lib, refactoring of the code

in the Oasis Local Db, started making a method, which created dummy data, and wrote a part of the common report.

Sprint Reflection

In this sprint we learned that we still need to be better at saying no to other groups, whom came with new tasks in the middle of the sprint.

3.1.5 Sprint Five

Sprint five started on 07/05-2012 with a work period of six half days. In this sprint we worked on implementing the Oasis App. We also planned to work on synchronizing with Savannah, however this did not happen due to Savannah not being ready. The Oasis Lib was improved with minor bug fixes.

Sprint Reflection

In this sprint we learned to communicate with the other groups before planning a mutual task.

3.1.6 Sprint Six

Sprint six started on 14/05-2012 with a work period of eight half days. In this sprint we continued the implementation of the Oasis App, updated the Oasis Lib, and started creating a structure for the rapport.

Sprint Reflection

In this sprint we learned nothing new.

3.1.7 Sprint Seven

Sprint seven started on 21/05-2012 with a work period of nine half days. In this sprint we worked on unit tests for the Oasis Lib. We also conducted the usability test, where we tested the Oasis App. Apart from that we also added more content to the rapport.

Sprint Reflection

In this sprint we learned that not every choice you make as a developer is easy to understand for someone without much technical knowledge.

Examples on this can be seen in the result of the usability test conducted on the Oasis App seen in Section 7.2.1 on page 55.

3.1.8 Sprint Eight

Sprint eight started on 28/05-2012 with a work period of eight half days. In this sprint we finished the report.

Sprint Reflection

In this sprint we learned that writing some parts of the report earlier might be a good idea.

Part III

Design and Implementation

CHAPTER 4

Oasis Local Db

In this chapter we describe the Oasis Local Db. The Oasis Local Db is used for storing the data, which is used by the GIRAF applications. The database schema, which the Oasis Local Db is based upon, is described in Section 4.1. The structure of SQLite used on the Android Platform is described in Section 4.2 on page 26. Finally the implementation of the Oasis Local Db on the Android system is described in Section 4.3 on page 26.

4.1 Database Schema

The Oasis Local Db Schema is developed in cooperation with Savannah. The reason for this is to alleviate the complexities that could occur during a synchronization of the Oasis Local Db and Savannah.

The central point of the Oasis Local Db schema is the AuthUsers table. This table contains all the user id's and their certificates. A user in the Oasis Local Db can be either a profile or a department and therefore a role is stored as well to differentiate the two.

The Oasis Local Db schema for the profiles and departments are a simple model representing a kindergarten like Birken or a school like Egebakken. This means that a profile can either be a child or guardian. The Oasis Local Db supports the possibility to associate profiles to each other in order to allow a child to guardian relation. The profiles can be attached to one or more departments, and a department can be related to one or more sub departments.

An important part of the system for the guardians is the ability to access different kinds of media on the tablets. Therefore media can be stored in the Oasis Local Db along with information about who has access to them. A media can be owned by either a profile or a department, and the owner has the ability to decide who should have access to the media. The Oasis Local Db schema also allows the user to adds tags to media, these tags can be used to identify the media.

Another important part of the system is the ability to control applications. From the users viewpoint it consists of deciding which applications should be accessible and from the developers viewpoint it is a way of storing application settings for a profile.

The Oasis Local Db schema can be seen in Figure 4.1 on the facing page.

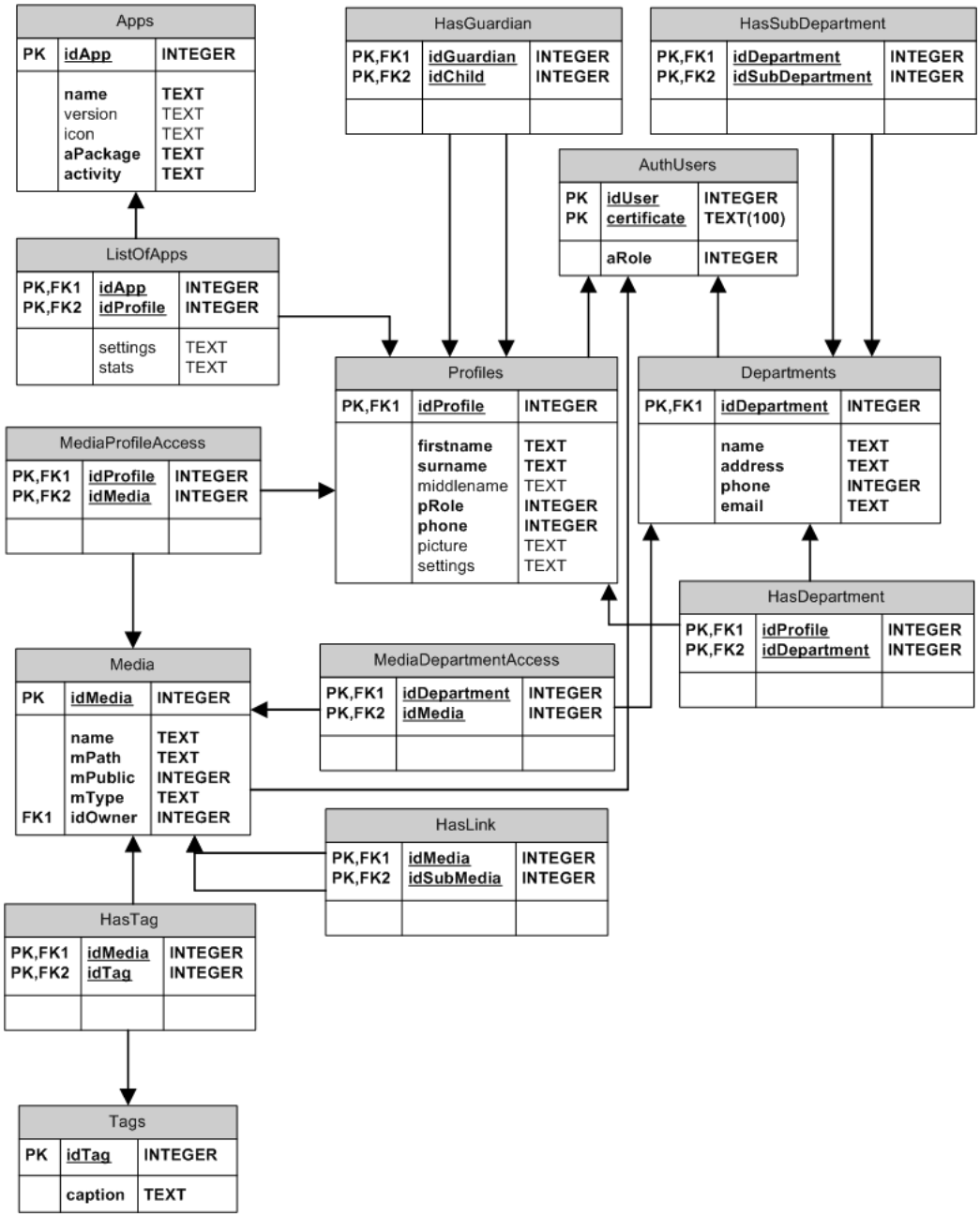


Figure 4.1: The schema for the Oasis Local Db.

4.2 Structure

When using the Android system to develop databases the developers are bound to use the open source database SQLite. [SQL12b] The reason for this is that only SQLite is supported on the Android platform. [Tea12b] It supports three kinds of data types; TEXT, which is similar to String type in Java, INTEGER, which is similar to long type in Java, and REAL, which is similar to double type in Java. [SQL12c] SQLite does not validate if the types written to the columns actually are of the defined type. This means that it for instance is possible to write an integer into a TEXT column. On the positive site SQLite only requires a little portion of memory at runtime. [SQL12a]

4.3 Implementation

As stated in Section 4.2, we implemented the Oasis Local Db using SQLite. The Oasis Local Db consists of three elements; metadata, tables, and a content provider. These are explained in the following sections.

4.3.1 Metadata

First we created metadata files for each table in the Oasis Local Db.

As an example the metadata file for the `AuthUsers` table can be seen in listing 4.1.

```
1  package dk.aau.cs.giraf.oasis.localdb;
2  import android.net.Uri;
3  import android.provider.BaseColumns;
4
5  public class AuthUsersMetaData {
6
7     public static final Uri CONTENT_URI =
            Uri.parse("content://dk.aau.cs.giraf.oasis.localdb.AutismProvider/authusers");
8
9     .
10    .
11    .
12
13    public class Table implements BaseColumns {
14       public static final String TABLE_NAME = "tbl_authusers";
15
16       public static final String COLUMN_ID = "_id";
17       public static final String COLUMN_CERTIFICATE = "authusers_certificate";
18       public static final String COLUMN_ROLE = "authusers_role";
19    }
20 }
```

Listing 4.1: The AuthUsers MetaData

In the `AuthUsers` metadata file the uri, `CONTENT_URI`, is used to defined which table to alter. The inner class Table defines the strings, which are used as names of the table and the columns.

4.3.2 Table

When the metadata file is created we make a table class file, which defines the SQL statements for creating, updating and deleting the table. The table class file for `AuthUsers` can be seen in listing 4.2.

```
1  package dk.aau.cs.giraf.oasis.localdb;
2  import android.database.sqlite.SQLiteDatabase;
3
4  public class AuthUsersTable {
5
6      private static final String TABLE_CREATE = "CREATE TABLE "
7              + AuthUsersMetaData.Table.TABLE_NAME
8              + "("
9              + AuthUsersMetaData.Table.COLUMN_ID + " INTEGER NOT NULL, "
10             + AuthUsersMetaData.Table.COLUMN_CERTIFICATE + " TEXT NOT NULL, "
11             + AuthUsersMetaData.Table.COLUMN_ROLE + " INTEGER NOT NULL, "
12             + "PRIMARY KEY (" + AuthUsersMetaData.Table.COLUMN_ID + ", " +
                   AuthUsersMetaData.Table.COLUMN_ID + ")"
13             + ");";
14
15     private static final String TABLE_DROP= "DROP TABLE IF EXISTS " +
               AuthUsersMetaData.Table.TABLE_NAME + ";";
16
17     public static void onCreate(SQLiteDatabase db) {
18        db.execSQL(TABLE_CREATE);
19     }
20
21     public static void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
22        db.execSQL(TABLE_DROP);
23        onCreate(db);
24     }
25 }
```

Listing 4.2: The AuthUsers Table

4.3.3 Content Provider

After the metadata and the tables for the Oasis Local Db has been created, a content provider is created. The content provider implemented in the Oasis Local Db allow applications to store and retrieve data from the aforementioned tables.

The content provider must implement several methods. These methods are presented along with a code snippet with AuthUsers as an example in the following bullets:

- `onCreate()` - called at startup to initialize the content provider see Listing 4.3.

```
1          public boolean onCreate() {
2             dbHelper = new DbHelper(getContext());
3             return false;
4          }
```

Listing 4.3: The `onCreate()` method.

- `getType()` - called when an application needs to know the type of the data. The `getType()` method can be seen in Listing 4.4. This method is not used in Oasis Local Db.

```
1         public String getType(Uri uri) {
2            switch(sUriMatcher.match(uri)) {
3            .
4            .
5            .
6            case AUTHUSERS_TYPE_LIST:
7               return AuthUsersMetaData.CONTENT_TYPE_AUTHUSERS_LIST;
8            case AUTHUSERS_TYPE_ONE:
9               return AuthUsersMetaData.CONTENT_TYPE_AUTHUSER_ONE;
10           .
11           .
12           .
13           default:
14              throw new IllegalArgumentException("Unknown URI: " + uri);
15           }
16        }
```

Listing 4.4: The `getType()` method.

- `query()` - called when an application wants to query in the Oasis Local Db. The query method can be seen in Listing 4.5.

```
1         public Cursor query(Uri uri, String[] projection, String selection, String[]
              selectionArgs, String sortOrder) {
2            SQLiteQueryBuilder builder = new SQLiteQueryBuilder();
3            switch(sUriMatcher.match(uri)) {
4            .
5            .
6            .
7            case AUTHUSERS_TYPE_LIST:
8               builder.setTables(AuthUsersMetaData.Table.TABLE_NAME);
9               builder.setProjectionMap(authusersProjectionMap);
10              break;
11           case AUTHUSERS_TYPE_ONE:
12              builder.setTables(AuthUsersMetaData.Table.TABLE_NAME);
13              builder.setProjectionMap(authusersProjectionMap);
14              builder.appendWhere(AuthUsersMetaData.Table.COLUMN_ID + " = " +
                    uri.getPathSegments().get(1));
15              break;
16           .
17           .
18           .
19           default:
20              throw new IllegalArgumentException("Unknown URI: " + uri);
21           }
22           SQLiteDatabase db = dbHelper.getReadableDatabase();
23           Cursor queryCursor = builder.query(db, projection, selection, selectionArgs, null,
                 null, null);
24           queryCursor.setNotificationUri(getContext().getContentResolver(), uri);
25           return queryCursor;
26        }
```

Listing 4.5: The `query()` method.

- `insert()` - called when an application wants to insert data into the Oasis Local Db. The `insert()` method can be seen in Listing 4.6.

```
1         public Uri insert(Uri uri, ContentValues values) {
2            SQLiteDatabase db = dbHelper.getWritableDatabase();
3            long rowId;
4            Uri _uri;
5
```

```
6             switch(sUriMatcher.match(uri)) {
7             .
8             .
9             .
10            case AUTHUSERS_TYPE_LIST:
11               try {
12                  rowId = db.insertOrThrow(AuthUsersMetaData.Table.TABLE_NAME, null, values);
13                  _uri = ContentUris.withAppendedId(AuthUsersMetaData.CONTENT_URI, rowId);
14                  getContext().getContentResolver().notifyChange(_uri, null);
15               } catch (SQLiteConstraintException e) {
16                  _uri = ContentUris.withAppendedId(AuthUsersMetaData.CONTENT_URI, -1);
17               }
18               return _uri;
19            .
20            .
21            .
22            default:
23               throw new IllegalArgumentException("Unknown URI: " + uri);
24            }
25         }
```

Listing 4.6: The `insert()` method.

- `update()` - called when an application wants to update existing data in the Oasis Local Db. The `update()` method can be seen in Listing 4.7.

```
1          public int update(Uri uri, ContentValues values, String where, String[] whereArgs) {
2             SQLiteDatabase db = dbHelper.getWritableDatabase();
3             int rowsUpdated = 0;
4             String rowId;
5
6             switch(sUriMatcher.match(uri)) {
7             .
8             .
9             .
10            case AUTHUSERS_TYPE_LIST:
11               rowsUpdated = db.update(AuthUsersMetaData.Table.TABLE_NAME, values, where,
                       whereArgs);
12               break;
13            case AUTHUSERS_TYPE_ONE:
14               rowId = uri.getPathSegments().get(1);
15               rowsUpdated = db.update(AuthUsersMetaData.Table.TABLE_NAME,
16                     values,
17                     AuthUsersMetaData.Table.COLUMN_ID + " = " + rowId +
                             (!TextUtils.isEmpty(where) ? " AND (" + where + ")" : ""),
18                     whereArgs);
19               break;
20            .
21            .
22            .
23            default:
24               throw new IllegalArgumentException("Unknown URI: " + uri);
25            }
26
27            getContext().getContentResolver().notifyChange(uri, null);
28            return rowsUpdated;
29         }
```

Listing 4.7: The `update()` method.

- `delete()` - called when an application wants to delete existing data in the Oasis Local Db. The `delete()` method can be seen in Listing 4.8.

```
1          public int delete(Uri uri, String where, String[] whereArgs) {
2             SQLiteDatabase db = dbHelper.getWritableDatabase();
3             int rowsDeleted = 0;
4             String rowId;
5             .
6             .
```

```
7               .
8               case AUTHUSERS_TYPE_LIST:
9                  rowsDeleted = db.delete(AuthUsersMetaData.Table.TABLE_NAME, where, whereArgs);
10                 break;
11              case AUTHUSERS_TYPE_ONE:
12                 rowId = uri.getPathSegments().get(1);
13                 rowsDeleted = db.delete(AuthUsersMetaData.Table.TABLE_NAME,
14                       AuthUsersMetaData.Table.COLUMN_ID + " = " + rowId +
                              (!TextUtils.isEmpty(where) ? " AND (" + where + ")" : ""),
15                       whereArgs);
16                 break;
17              .
18              .
19              .
20              default:
21                 throw new IllegalArgumentException("Unknown URI: " + uri);
22              }
23
24              getContext().getContentResolver().notifyChange(uri, null);
25              return rowsDeleted;
26          }
```

Listing 4.8: The `delete()` method.

The implemented methods in the content provider supply the functionality for the Oasis Local Db. This functionality can be utilized in the implementation of the Oasis Lib.

CHAPTER 5

Oasis Lib

In this chapter we describe the Oasis Lib. The Oasis Lib is the library, which works as a connection between the Oasis Local Db, see Section 4 on page 23, and the GIRAF applications. The Oasis Lib provides an API, which the GIRAF applications can use. The structure of the Oasis Lib is described in Section 5.1. Finally the implementation of the Oasis Lib is described in Section 5.2 on page 33.

5.1 Structure

The structure of the Oasis Lib, have been inspired of the MVC pattern, where the system is divided into three parts; Model, View, and Controller.

In Oasis Lib the Model part is a package containing model classes. Each model class represents a table in the Oasis Local Db. The model classes are used to encapsulate the data, which is to be stored and retrieved from the Oasis Local Db. The reason that we use model classes is to ease it for the users of the library and to make a uniform way of storing and retrieving data. The model classes can be seen in Figure 5.1 on the next page.

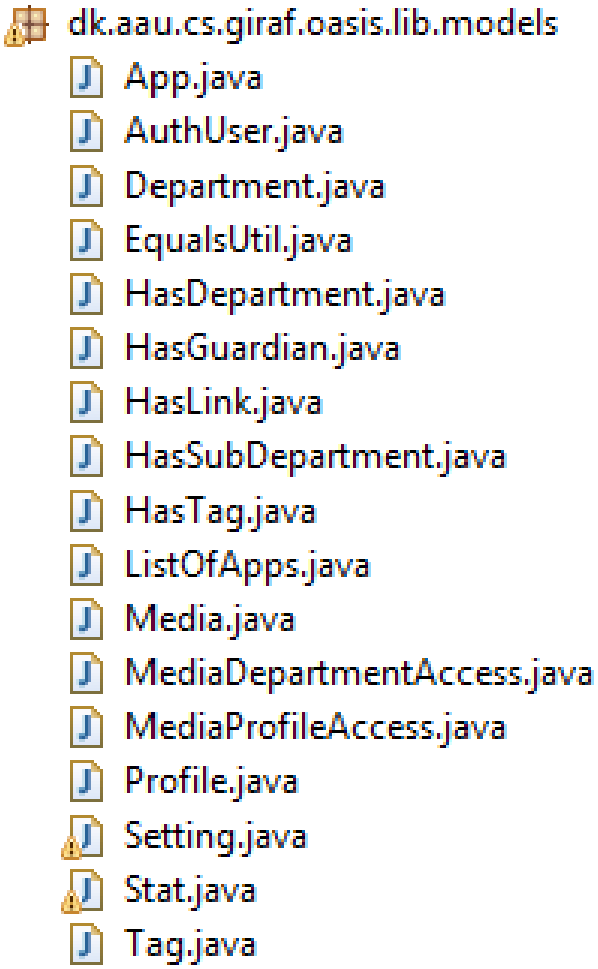


Figure 5.1: The models in the Oasis Lib

The Controller part is the package containing all the methods, which the developers can use to interact with Oasis Local Db. The controller methods have been divided into several classes. The division can be seen in Figure 5.2 on the facing page.

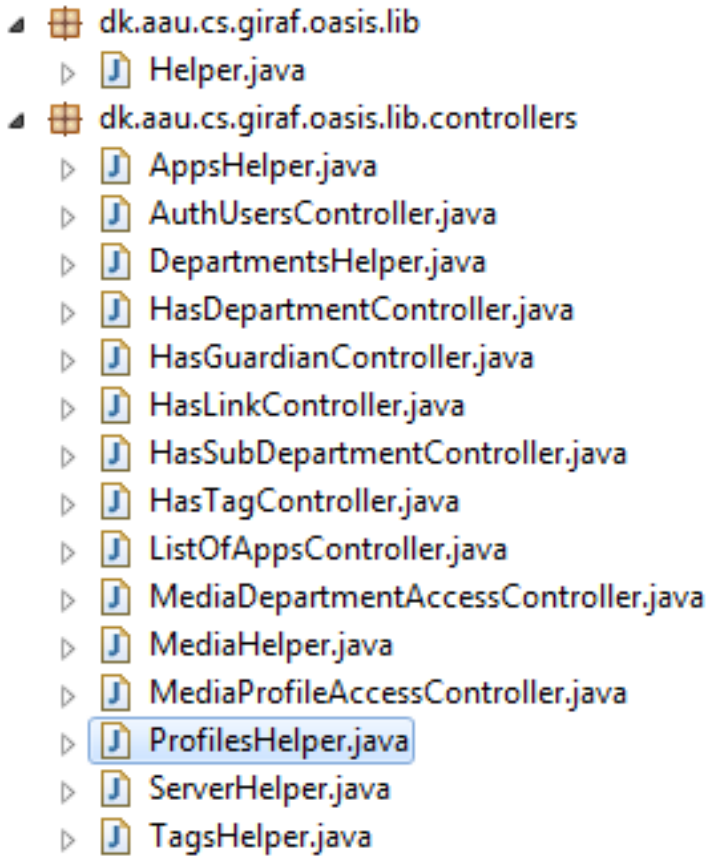


Figure 5.2: The controllers in the Oasis Lib

The controllers are divided by the models they manipulate, this means that for each table in the Oasis Local Db there is a controller.

In the Oasis Lib there is no direct reference to the Views from the MVC pattern. This is because the individual applications in the GIRAF system is seen as a view.

5.2 Implementation

The Oasis Lib has been designed with the intention of having as few method calls as possible to gain access to the Oasis Local Db. The code for all the implemented methods will not be show in this section. Two examples from the Oasis Lib will be presented, these methods are `autenticateProfile` and `getProfileById`.

The method `autenticateProfile` is used to authenticate a profile in order to decide what informations the user should be allowed access to.

First the method verifies that the input is not `null` and that the certificate conformes to the rules for the certificates, if the certificate is rejected, `null` is returned to indicate this.

After the certificate has been verified the profile id will be retrived from the Oasis Local Db, iff a profile exists with that certificate. In case a profile exist with the certificate the id will be returned and the profile model will be retrieved from the Oasis Local Db. If no profile with the certificate exists, `-1` will be returned and no profile model will be retrieved.

The code for `autenticateProfile` can be seen in Listing 5.1.

```
1  public Profile authenticateProfile(String certificate) {
2     if (certificate == null || !certificate.matches("[a-z]{200}")) {
3        return null;
4     }
5
6     Profile profile = null;
7     long id = au.getIdByCertificate(certificate);
8
9     if (id != -1) {
10       profile = getProfileById(id);
11    }
12
13    return profile;
14 }
```

Listing 5.1: The `authenticateProfile` method.

The method `getProfileById` is used the retrieve a profile model from the database. The model is retrived using the profile id as a parameter to get the model by.

First it must be ensured that the id is above zero, as the database can not handle zero or negative values. After this check the database is queried to retrive the profile model. A conversion is needed as the database returns a Cursor object and this object must be converted into a profile model. [Tea12c] This conversion is done in the auxiliary method `cursorToProfile`, which maps values in the Cursor to values in the profile model. If the Cursor from the database is null or empty, no profile was found in the database.

The code for `getProfileById` method can be seen in Listing 5.2.

```
1  public Profile getProfileById(long id) {
2     Profile profile = null;
3
4     if (id <= 0) {
5        return null;
6     }
7
8     Uri uri = ContentUris.withAppendedId(ProfilesMetaData.CONTENT_URI, id);
9     Cursor c = _context.getContentResolver().query(uri, columns, null, null, null);
10
11    if (c != null) {
12       if (c.moveToFirst()) {
13          profile = cursorToProfile(c);
14       }
15       c.close();
16    }
17
18    return profile;
19 }
```

Listing 5.2: The `getProfileById` method.

For further information on how the Oasis Lib methods work, see the JavaDoc or the source code, which are placed on the attached CD-ROM.

CHAPTER 6

Oasis App

In this chapter we describe the Oasis App. The Oasis App is an application, which demonstrates some of the utilities the Oasis Lib offers. First the design of the Oasis App is described in Section 6.1. After that the implementation of the Oasis Lib in the Oasis App is described in Section 6.2 on page 39.

6.1 Design

The idea behind the Oasis App is that we want to make a tool, for the guardians, to manage the profile data, by giving them CRUD (Create, Read, Update, and Delete) options.

The application design should be revolving around the database structure with the key functionality being profile handling. The application design has been divided into two parts; the home screen of the application and an overview screen showing relevant data.

Paper prototypes of the GUI has been created and evalutated. Several prototypes have been made and the chosen prototype for the application can be seen in Figure 6.1 on the next page and Figure 6.2 on the following page.

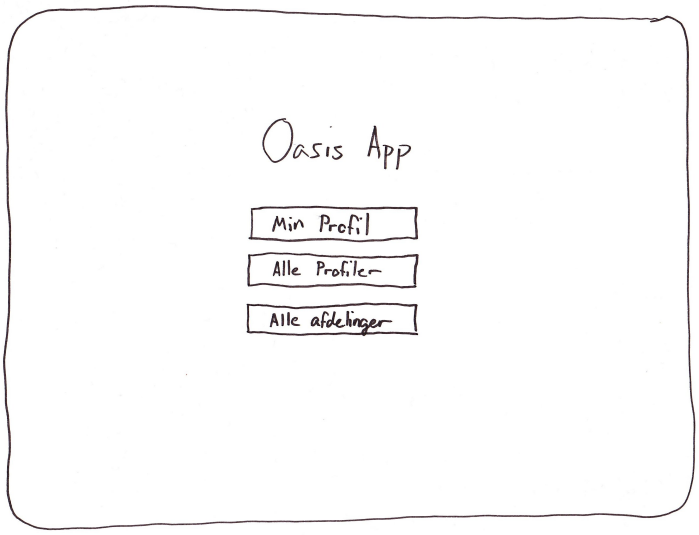


Figure 6.1: The prototype for the home screen.

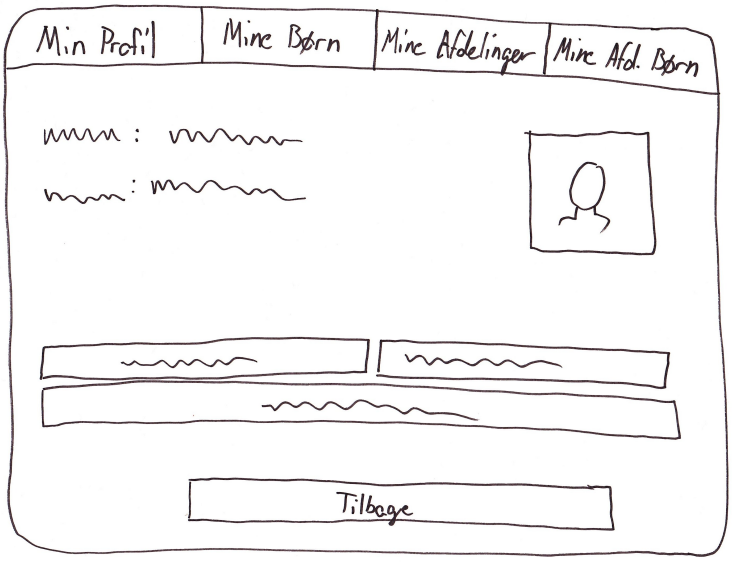


Figure 6.2: The prototype for the overview screen.

The final prototypes have been implemented as the GUI in the Oasis App.

6.2 Implementation

The Oasis App is split into two activities; `MainActivity` and `FragParentTab`.

6.2.1 MainActivity

The `MainActivity` is the activity that starts at application startup. It uses the main.xml as its layout file, which is a layout file containing three buttons, which can be seen on Figure 6.3.

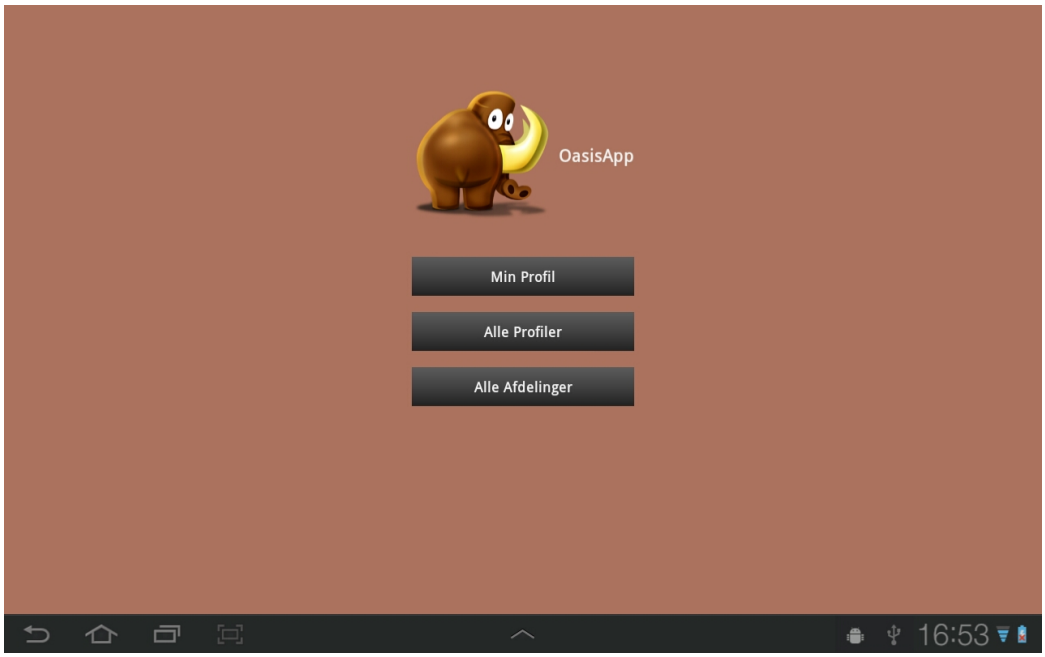


Figure 6.3: An image of the `MainActivity` layout.

A code snippet of the MainActivity's code can be seen in Listing 6.1.

```
1  .
2  .
3  .
4  public class MainActivity extends Activity implements OnClickListener {
5
6      private Button bMyProfile, bAllProfiles, bAllDepartments, bAddDummyData;
7      private Intent direct;
8      private long guardianId;
9      public Helper helper;
10     public static Profile guardian;
11     public static Profile child;
12     public static int color;
13
```

```
14      @Override
15      public void onCreate(Bundle savedInstanceState){
16         super.onCreate(savedInstanceState);
17
18         .
19         .
20         .
21
22         helper = new Helper(this);
23
24         Bundle extras = getIntent().getExtras();
25         if (extras != null) {
26            guardianId = extras.getLong("currentGuardianID");
27            color = extras.getInt("appBackgroundColor");
28            guardian = helper.profilesHelper.getProfileById(guardianId);
29         }
30
31         setContentView(R.layout.main);
32
33         initializeViews();
34      }
35
36      private void initializeViews() {
37         findViewById(R.id.UpperLayout).setBackgroundColor(color);
38
39         bMyProfile = (Button) findViewById(R.id.bMyProfile);
40         bMyProfile.setOnClickListener(this);
41         bAllProfiles = (Button) findViewById(R.id.bAllProfiles);
42         bAllProfiles.setOnClickListener(this);
43         bAllDepartments = (Button) findViewById(R.id.bAllDepartments);
44         bAllDepartments.setOnClickListener(this);
45         bAddDummyData = (Button) findViewById(R.id.bAddDummyData);
46         if (guardian == null) {
47            bAddDummyData.setOnClickListener(this);
48         } else {
49            bAddDummyData.setVisibility(View.GONE);
50         }
51      }
52
53      @Override
54      public void onClick(View v) {
55         direct = new Intent(this, FragParentTab.class);
56
57         switch (v.getId()) {
58         case R.id.bMyProfile:
59            if (guardian != null) {
60               direct.putExtra("tabView", FragParentTab.TABPROFILE);
61               startActivity(direct);
62            } else {
63               Toast.makeText(this, R.string.noprofile, Toast.LENGTH_SHORT).show();
64            }
65            break;
66         case R.id.bAllProfiles:
67            direct.putExtra("tabView", FragParentTab.TABALLPROFILES);
68            startActivity(direct);
69            break;
70         case R.id.bAllDepartments:
71            direct.putExtra("tabView", FragParentTab.TABALLDEPARTMENTS);
72            startActivity(direct);
73            break;
74         .
75         .
76         .
77         }
78      }
79   }
```

Listing 6.1: The MainActivity class

As the activity starts it gets the information of which guardian who is currently logged in to the GIRAF system and what background color the Oasis App is currently set to by the Launcher. When one of the buttons is clicked, the `MainActivity` will start the `FragParentTab` activity and put the corresponding integer in the intent's extra data.

6.2.2 FragParentTab

The FragParentTab is the activity, which is started by the `MainActivity` activity. The activity has the responsibillity of managing what view to show, by using fragments. We chose fragments because we wanted a tab layout. An example of the tab layout view can be seen in Figure 6.4.

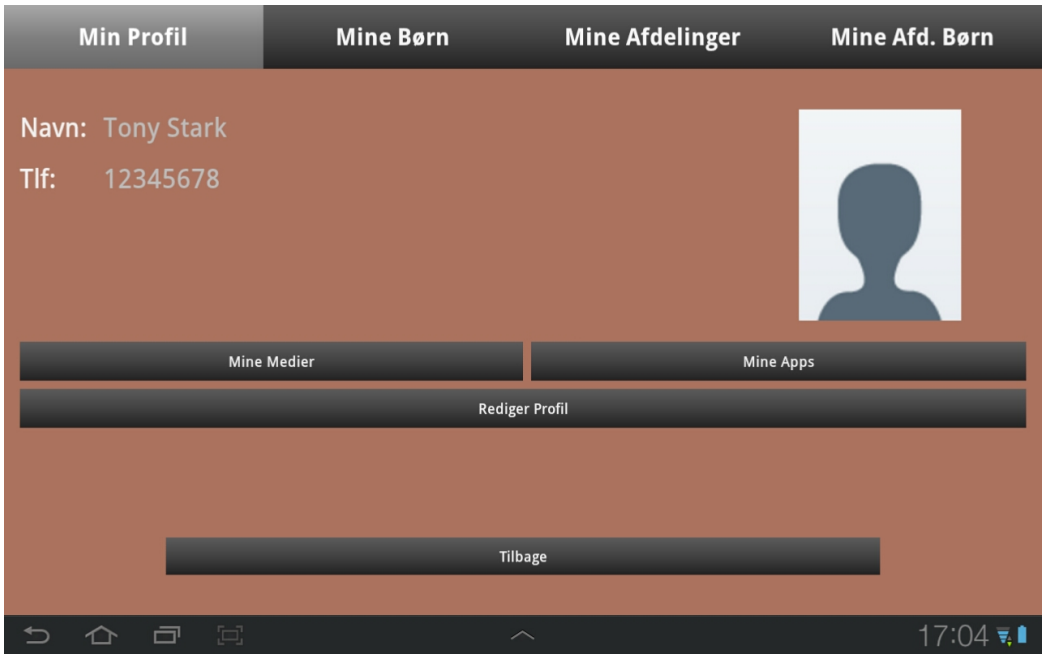


Figure 6.4: An image of the `FragParentTab` layout.

The FragParentTab activity can be seen in Listing 6.2.

```
1  .
2  .
3  .
4  public class FragParentTab extends Activity {
5
6      private int tabView;
7      public final static int TABPROFILE = 0;
8      public final static int TABAPP = 1;
9      public final static int TABMEDIA = 2;
10     public final static int TABALLPROFILES = 3;
11     public final static int TABALLDEPARTMENTS = 4;
12     public final static int TABCHILD = 5;
13     public final static int TABCHILDAPP = 6;
14     public final static int TABCHILDMEDIA = 7;
15     static FragmentManager t;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20
21         .
22         .
23         .
24
25         Bundle extras = getIntent().getExtras();
26         if (extras != null) {
```

```
27          tabView = extras.getInt("tabView");
28       } else {
29          tabView = -1;
30       }
31
32       setContentView(R.layout.fragments_view);
33
34       findViewById(R.id.fragUpperLayout).setBackgroundColor(MainActivity.color);
35
36       t = getFragmentManager();
37
38       switch(tabView) {
39       case TABPROFILE:
40          t.beginTransaction().add(R.id.fDetails, new TabManagerProfile()).commit();
41          break;
42       case TABALLPROFILES:
43          t.beginTransaction().add(R.id.fDetails, new TabManagerAllProfiles()).commit();
44          break;
45       case TABALLDEPARTMENTS:
46          t.beginTransaction().add(R.id.fDetails, new TabManagerAllDepartments()).commit();
47          break;
48       case TABCHILD:
49          t.beginTransaction().add(R.id.fDetails, new TabManagerChild()).commit();
50       }
51    }
52
53    public static void switchTab(int tabViewId) {
54
55       switch(tabViewId) {
56       case TABPROFILE:
57          t.beginTransaction().replace(R.id.fDetails, new TabManagerProfile()).commit();
58          break;
59       case TABMEDIA:
60          t.beginTransaction().replace(R.id.fDetails, new TabManagerMedia()).commit();
61          break;
62       case TABAPP:
63          t.beginTransaction().replace(R.id.fDetails, new TabManagerApp()).commit();
64          break;
65       case TABCHILD:
66          t.beginTransaction().replace(R.id.fDetails, new TabManagerChild()).commit();
67          break;
68       case TABCHILDMEDIA:
69          t.beginTransaction().replace(R.id.fDetails, new TabManagerChildMedia()).commit();
70          break;
71       case TABCHILDAPP:
72          t.beginTransaction().replace(R.id.fDetails, new TabManagerChildApp()).commit();
73          break;
74       }
75    }
76
77    @Override
78    protected void onResume() {
79       super.onResume();
80       t = getFragmentManager();
81    }
82 }
```

Listing 6.2: The FragParentTab class

The activity controls which fragment it must show and this is done in two ways. The first way is when the activity is created, it decides which fragment to show, by using the integer it gets from the `MainActivity`. This integer represents a fragment class of every tab layout view. This fragment is then added to the fragment stack.

The other way is when a fragment wants to replace itself with another fragment. Here the fragment calls the switchTab method, in the `FragParentTab` activity, with the replacing view integer as a parameter.

6.2.3 Utilizing the Oasis Lib

Before utilizing the methods within the Oasis Lib, the developer must include the library. It is also necassary to instantiate a helper object before it can be used. When instantiating the object it is necassary to input the current activity's context as a parameter. This is needed to give the Oasis Lib the information about where it is called from. An example of how to instantiate the helper object, can be seen in Listing 6.3.

```
1 Helper helper = new Helper(getActivity().getApplicationContext());
```

Listing 6.3: Example of Instantiating a helper object.

After the instantiation it is possible to call all the methods within the Oasis Lib. An example of calling a method can be seen in Listing 6.4.

```
1 guardian = helper.profilesHelper.getProfileById(guardianId);
```

Listing 6.4: Call method from the Oasis Lib.

CHAPTER 7

Testing

In the chapter aspects of dynamic white box testing will be explained in Section 7.1. Furthermore the results of the usability test will be presented in Section 7.2.1 on page 55.

7.1 Dynamic White Box Testing

To ensure the correctness of the Oasis Lib we enforced dynamic white box testing through unit tests [Pat, pp106] [IEE93]. The Oasis Lib is used by all GIRAF applications this means that if a bug exists in the Oasis Lib there is a potential bug in all GIRAF applications. Therefore the Oasis Lib must be thoroughly tested to make sure that few or no bugs exists.

The Oasis Local Db will be tested along with the Oasis Lib, this is more efficient as more code will be tested in every test, but it has the drawback that if a bug appears more code will have to be investigated in order to locate the bug.

As this project have been developed using the agile development method Scrum we have not devised a full test plan as this is not needed [Pat, pp263]. The focus of the tests will be on test-to-pass [Pat, pp66]. This ensures that the Oasis Lib will function as intended, though there is no guarantee that the Oasis Lib will work if invalid parameters are used.

7.1.1 The Test Designs

A test design have been elaborated for each method in the helper classes of the Oasis Lib [Pat, pp281]. Examples of the test designs are presented in the following tables.

The test design in Table 7.1 is for the `authenticateProfile()` method. This test design tests if a profile can be authenticated. This is an essential method for the GIRAF platform, and therefore it is tested both using test-to-pass and test-to-fail tests to ensure that this method is particular robust.

Identifier:	TD00001
Feature to be tested:	Authenticate profile.
Approach:	An automated test will be made to authenticate a profile by its certificate. 1. Enter a profile with a specific certificate in the database. 2. Authenticate the profile by its certificate.
Test case identification:	Check valid certificate Test Case ID# 00001 Check too long certificates Test Case ID# 00002 Check too short certificates Test Case ID# 00003 Check invalid certificates Test Case ID# 00004
Pass/fail criteria:	All valid profile certificates that matches the certificate in the database must be accepted as well as all invalid certificates must be rejected.

Table 7.1: Test Design for `authenticatingProfile()`.

The test design in Table 7.2 on the facing page is for the `getProfileById()` method. This is also an important method for the GIRAF system, therefore it is tested with the same mix of test-to-pass and test-to-fail tests. The tests ensures the robustness of the method.

Identifier:	TD00002
Feature to be tested:	Get profile by id.
Approach:	An automated test will be made in order to ensure that the Oasis Library supports getting a profile by its id from the database. 1. Add Profiles to the database. 2. Get profile by id and verify the output.
Test case identification:	Check valid id present in the database Test Case ID# 00005 Check id not in the database Test Case ID# 00006 Check negative id Test Case ID# 00007
Pass/fail criteria:	The profile matching the id should be returned else null should be returned.

Table 7.2: Test Design for `getProfileById()`.

The test design in Table 7.3 on the next page is for the `getChildrenByGuardian()` method. The test ensures the method performs as intended under normal operation. This is done with a single test-to-pass test, which tests if children associated to a guardian can be retrieved from the database.

Identifier:	TD00003
Feature to be tested:	Get children by guardian.
Approach:	An automated test will be made to ensure that the Oasis Library supports getting all children associated with one guardian. 1. Children and guardians should be added to the database. 2. Associations between some children and guardians should be made. 3. Get children by guardian should be called and the output verified.
Test case identification:	Check valid guardian with children associated Test Case ID# 00008
Pass/fail criteria:	The list of children should match the children associated with the guardian.

Table 7.3: Test Design for `getChildrenByGuardian`.

7.1.2 The Test Cases

For each test design in Section 7.1.1 on page 46, one or more test cases has been created [Pat, pp283]. Each test case ensures a part of the tested method performs as intended in the test situation. The test cases for the test designs presented in Section 7.1.1 on page 46 will be shown in the following tables: Table 7.4 on the facing page, Table 7.5 on the next page, Table 7.6 on the facing page, Table 7.7 on page 50, Table 7.8 on page 50, Table 7.9 on page 50, Table 7.10 on page 51, and Table 7.11 on page 51.

Identifier:	TC00001
Test item:	Valid Certificate handling of the `authenticateProfile()` method.
Input specification:	A valid certificate.
Output specification:	The model of the authenticated profile.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.4: Test Case for valid certificate handling of the `authenticateProfile()` method.

Identifier:	TC00002
Test item:	Certificate length too short handling of the `authenticateProfile()` method.
Input specification:	A certificate shorter than 200 chars.
Output specification:	Null.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements	None.
Intercase dependencies:	None.

Table 7.5: Test Case for certificate length too short handling of the `authenticateProfile()` method.

Identifier:	TC00003
Test item:	Certificate length too long handling of the `authenticateProfile()` method.
Input specification:	A certificate longer than 200 chars.
Output specification:	Null.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.6: Test Case for certificate length too long handling of the `authenticateProfile()` method.

Identifier:	TC00004
Test item:	Invalid Certificate handling of the `authenticateProfile()` method.
Input specification:	An invalid certificate.
Output specification:	Null.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.7: Test Case for invalid certificate handling of the `authenticateProfile()` method.

Identifier:	TC00005
Test item:	Valid id present in the database handling of the `getProfileById()` method.
Input specification:	A valid id.
Output specification:	The profile matching the id.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.8: Test Case for valid id handling of the `getProfileById()` method.

Identifier:	TC00006
Test item:	Invalid id not present in the database handling of the `getProfileById()` method.
Input specification:	An invalid id.
Output specification:	Null.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.9: Test Case for invalid id handling of the `getProfileById()` method.

Identifier:	TC00007
Test item:	Negative id handling of the `getProfileById()` method.
Input specification:	A negative id.
Output specification:	Null.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.10: Test Case for negative id handling of the `getProfileById()` method.

Identifier:	TC00008
Test item:	Valid guardian with children associated handling of the `getChildrenByGuardian()` method.
Input specification:	A valid guardian.
Output specification:	A list of associated children.
Environmental needs:	A database is needed and the profile model.
Special procedural requirements:	None.
Intercase dependencies:	None.

Table 7.11: Test Case for valid guardian with children associated handling of the `getChildrenByGuardian()` method.

7.1.3 Unit Test Implementation

All the tests cases have been used to construct the 89 unit tests, which have helped in the development of the Oasis Lib. The tests have been split up into three parts; initialization, execution, and assertion.

The unit test for the `testAuthenticateProfileWithValidCertificate()` method can be seen in Listing 7.1 on the following page. The method starts by initializing the environment. This means that a random valid certificate is created and a profile – the expected profile – is entered in the database and the certificate for the profile is set to the newly created certificate.

After the initialization the method is executed and the retrieved profile is stored as the actual profile in order to have some data for the assertion. At the end an `assertEquals` is called to ensure that the expected profile is the same as the actual profile. If this is the case the test passes, otherwise the test fails and the two profiles are printed in the test log.

```
1  public void testAuthenticateProfileWithValidCertificate() {
2     Random rnd = new Random();
3     StringBuilder cert = new StringBuilder();
4     for (int i = 0; i < 200; i++)
5     {
6        cert.append((char)(rnd.nextInt(26) + 97));
7     }
8     String certificate = cert.toString();
9     Profile expectedProfile = new Profile("Test", "Profile", null,
            Profile.pRoles.GUARDIAN.ordinal(), 12345678, null, null);
10
11    long id = mActivity.helper.profilesHelper.insertProfile(expectedProfile);
12    expectedProfile.setId(id);
13
14    mActivity.helper.profilesHelper.setCertificate(certificate, expectedProfile);
15
16    Profile actualProfile = mActivity.helper.profilesHelper.authenticateProfile(certificate);
17
18    assertEquals("Should return profile; Test Profile", expectedProfile, actualProfile);
19 }
```

Listing 7.1: The `testAuthenticateProfileWithValidCertificate()` method.

The unit test for the `testAuthenticateProfileWithInvalidCertificate()` method can be seen in Listing 7.2. This method also starts by initializing its environment but in this method the created certificate is invalid. The rest of the initialization is the same as in the valid test and the execution is the same as well. In this test `assertNull` is used to confirm that there is not retrieved a profile from the database due to an invalid certificate.

```
1  public void testAuthenticateProfileWithInvalidCertificate() {
2     Random rnd = new Random();
3     StringBuilder cert = new StringBuilder();
4     for (int i = 0; i < 200; i++)
5     {
6        cert.append((char)(rnd.nextInt(26) + 65));
7     }
8     String certificate = cert.toString();
9     Profile expectedProfile = new Profile("Test", "Profile", null,
            Profile.pRoles.GUARDIAN.ordinal(), 12345678, null, null);
10
11    long id = mActivity.helper.profilesHelper.insertProfile(expectedProfile);
12    expectedProfile.setId(id);
13
14    mActivity.helper.profilesHelper.setCertificate(certificate, expectedProfile);
15
16    Profile actualProfile = mActivity.helper.profilesHelper.authenticateProfile(certificate);
17
18    assertNull("Should return null", actualProfile);
19 }
```

Listing 7.2: The `testAuthenticateProfileWithInvalidCertificate()` method.

The remaining tests from the test suite have been created using the same structure; an initialization phase, an execution phase, and one or more assertions. These tests can be seen in the source code of the Oasis App.

7.1.4 The Test Results

The result for the unit tests are 88 out of 89 tests passed and this can be seen in Figure 7.1.

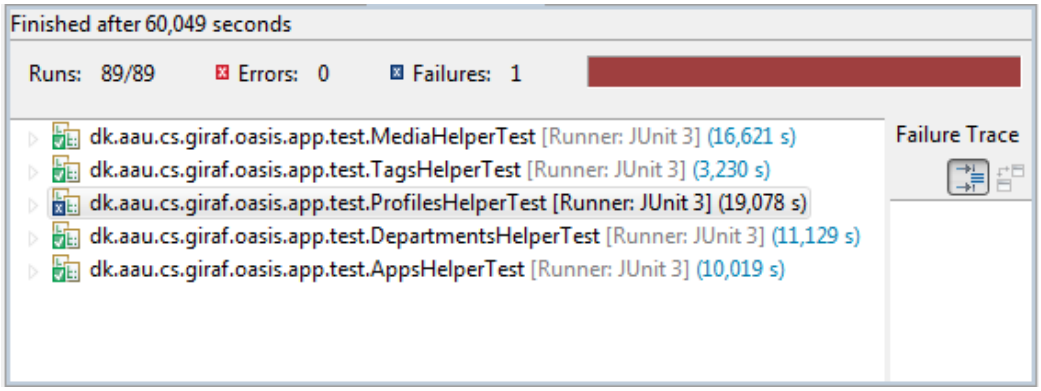


Figure 7.1: The result from all the performed unit tests.

The test that failed is `testGetChildrenByDepartmentAndSubDepartments()` and the result can be seen in Figure 7.2 on the next page.

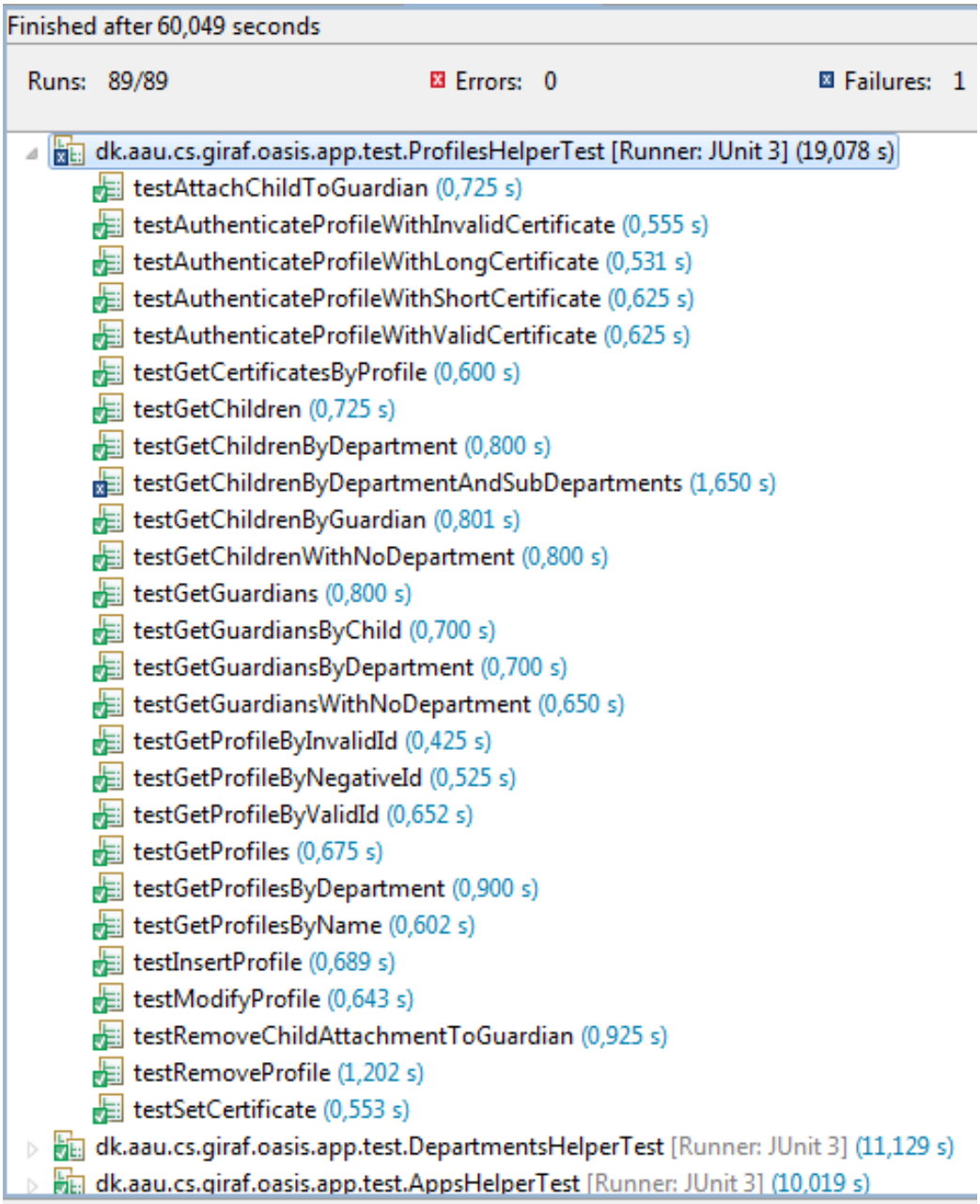


Figure 7.2: The result from the `profilesHelper` tests.

The individual results for the rest of the test suite can be seen in Appendix 11.8 on page 102.

7.2 Usability Test

We decided to have a usability test performed on the Oasis App even though it was not completed. This gives a unique opportunity to modify the application before it is done.

The approach for the usability test can be found in 1.8 on page 7. The questions for the usability test of the Oasis App can be seen in 11.7 on page 102.

7.2.1 Results and Observations

The result of the usability test can be seen in Table 7.12.

Issues	**Description**
Cosmetic:	Profile id was showing. "Add child" not clear how to.
Serious:	Too many options, missing the overview. Difficult finding a child profile.
Critical:	Data availability unclear.

Table 7.12: The issues found in the usability test.

The result shows the following number of issues; two cosmetic issues, two serious issues, and one critical issue.

Part IV

Epilogue

CHAPTER 8

Discussion

In the follow sections we will discuss the used development method and the system architecture.

8.1 Development Method

In the following section we will discuss the development method, as described in Section 1.4 on page 5.

8.1.1 Agile Development

Using the agile development method, has given us the ability to adjust according to the requirements coming from the other groups. The Oasis Lib has been subject for many requirements, coming during sprints, and the agile development method has made it easy to adjust to requirements as they came.

8.1.2 Meetings

In the initial planning phase of the project we held several meetings to discuss which product should be made. During the project we have been meeting in the multi project group for the start of each sprint, as well as an evaluation at the end of each sprint. These meetings have given us the ability to follow what the other groups are doing at all times, and therefore plan a bit ahead of the requirements that might come.

8.1.3 Sprint Length

On the meetings at the start of each sprint, we decided how long the sprint should run. Most sprints ended up being around seven half days of work. This sprint length was fitting for our group because we could finish some tasks, while keeping up with what the other groups needed.

8.1.4 Project Owner

In this project we had no project owner, this made it a bit harder to decide things in the multi project meetings, as every person had to agree. However it added the option to influence the project, as an individual, instead of one person deciding it all. In some situations a project owner would have been able to make a decision and we could have avoided a lot of pointless discussion.

8.2 System architecture

In the following section we will discuss the system architecture for Oasis, as described in section 2.3 on page 14.

When designing the architecture, there are multiple ways we could do it. We will list the options we have, their pros and cons, and argue why we choose the one we did.

The pros and cons can be seen in the following tables; Table 8.1 on the facing page, Table 8.2 on page 62, Table 8.3 on page 62, Table 8.4 on page 63.

8.2.1 Choosing the architecture

One of the requirements for the system was accessibility of data in offline mode. This requirement rules out the two options without a local database. The main difference between the two solutions left, is that one requires more development time, and the other forces the tablets to be updated every time the external database changes. Seeing that we could manage the increased development time, from developing two software layers, this solution seems like the more optimal one. Therefore we choose the one described in Table 8.3 on page 62.

Pros:	**Cons:**
• Reduced development time, as you only have to develop one software layer • Access to data in offline mode	• It is not recommended to have direct access to the external database, due to the loss of security and the maintainability will be more complex • If more applications wants to access the external database, then a lot of connections will be made • If a change is made in the external database structure, one has to update the tablet to use it

Table 8.1: Oasis Lib with direct connection to an external database and with a local database

Pros:	Cons:
• Data are always synchronized • Reduced development time, as you only have to develop one software layer	• No access to data, if the external database is offline • If a change is made in the external database structure, one has to update the tablet to use it • It is not recommended to have direct access to the external database, due to the loss of security and the maintainability will be more complex • If more applications wants to access the external database, then a lot of connections will be made

Table 8.2: Oasis Lib with direct connection to an external database and without a local database

Pros:	Cons:
• The external database can be changed without updating the tablet • Access to data, if the external database is offline	• Increased development time, as you have to develop two software layers • Increased development time, as you have to setup two databases • Data is not always synchronized

Table 8.3: Oasis Lib with connection to software layer on the server and with a local database

Pros:	**Cons:**
• The external database can be changed without updating the tablet • Data is always synchronized	• No access to data if the external database is offline • Increased development time, as you have to develop two software layers • Data is not always synchronized

Table 8.4: Oasis Lib with connection to software layer on the server and without a local database

CHAPTER 9

Conclusion

In this chapter we described the things we concluded along the project.

9.1 GIRAF Problem Definition

In the multi project we made a problem definition. The problem definition is as follows:

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

To comply with the study regulations, we (the multi project) have designed and implemented a system called GIRAF. To ensure that every project group has the opportunity to be up to date of the progress of the multi project, we (the multi project) agreed on using the same development method 1.4 on page 5.

9.2 Oasis Problem Definition

In Oasis we have specified the multi project problem definition to fit our project. The problem definition is a follows:

> *How can we provide a set of tools, which can help develop applications for the GIRAF system?*

We have designed and implemented an administration module for the GIRAF system. The administration module consists of three parts; a local database, called Oasis Local Db, a library, called Oasis Lib, and an administration application, called Oasis App. The Oasis Local Db ensures that the data is saved correctly. The Oasis Lib ensures that the different applications of the GIRAF system can interact with Oasis Local Db. The Oasis App ensures that the guardians can manage profiles of the GIRAF system, directly on the tablets.

9.3 Oasis

We began by examining the previous student reports, to check if there was any aspects we could reuse in the project. After that we examined the possibilities of how to save data on an Android device, which lead us to start working on the architecture of the local database. Along with that we gathered requirements from the other groups to start working on the architecture of the library. When we finished the Oasis Local Db and the Oasis Lib we started working on the Oasis App. The Oasis App shows some of the capabilities of the Oasis Lib, and by the same time give the guardians a possibility on managing the different profiles of the GIRAF system.

9.4 Testing

To verify the quality of the multi project, we conducted a usability test. The test subjects consisted of the customers of the multi project. The test highlighted some issues in the Oasis App, which could be corrected by the next group of developers, the issues can be seen in Section 7.2.1 on page 55. We created unit tests for the Oasis Lib. The tests were made at the end of the project period. This should be an ongoing process instead of doing them at the end of the semester. The tests ensures indirectly, that the Oasis Local Db works.

CHAPTER 10

Future Work

A number of tasks did not get completed in this semester. As this project is going to be continued by others students, we will provide an overview of some of the changes we did not complete.

10.1 Server Synchronization

One of the main things which did not get completed was the synchronization with Savannah. This was due to components not being ready at the time the task was scheduled. In the continuation of the project, this can be implemented by using the components which Savannah provides. Completing this task will make the sync status component in the launcher work properly. Another improvement which can be implemented in a future continuation of the project, is the ability to synchronize images on the device, and update the paths dynamically.

10.2 Unit Tests

Unit testing is an essential part of the project. We created unit test for all the helper classes in the Oasis Lib, but for a major part it was only test-to-pass tests. Therefore the Oasis Lib can be made more robust by implemting test-to-fail tests. In the future it can be beneficial to make unit tests for the Oasis Local Db

and the Oasis App. This would make the administration module more robust, because every “part” of the module is tested.

10.3 Certificates

Certificates is one of the core elements in the Launcher, and this is reflected in the Oasis Lib.

A feature that can be implemented for the certificates, is the possibility to set a time limit on the certificate, thereby enforcing a renewal of the certificate after the time limit has been exceeded. This would make the system more secure, but would rely on the users printing out new QR-codes, and the Oasis Lib to generate new QR-codes.

10.4 Media Table

As seen in the database schema in 4.1 on page 23, a media should be capable of having either a department or a profile as its owner id. The Oasis Lib only supports a profile as the owner of a media. The option for departments to be owners should be added, to make the Oasis Lib fully represent the database schema.

10.5 Oasis App

The Oasis App shows how the Oasis Lib can be utilized. A couple of changes and improvements can be done.

One thing which can be done is refactoring of the code. This refactoring would lower the amount of classes, increase the readability, and help with the understanding of the Oasis App source code.

Besides that the Oasis App is still missing some functionality. The functionality that is missing is; view other guardians profiles, create new media, create new applications, and create and manage settings of the applications and profiles.

The usability test showed that the Oasis App can use a better visual design to give a better overview of the application.

Part V

Appendix

CHAPTER 11

Requirements from Wombat

I behøver ikke smide det ind i objekter, da vi har vi allerede lavet objekter til vores data. Hvis vi blot kan få dataen i en eller anden form for array, er det helt fint.

Vi ved ikke helt hvad for noget data der skal gemmes i settings, men vi har forstået på Henrik at man selv kan definere det når man gemmer. Template

Function template Her skriver man funktionen skal kunne Data Her skriver man hvilken data man gerne vil modtage Damer Create

Funktion createAutistSettings Lave multiple Settings der er forbundet til en Autist

Funktion createLastUsedGuardian Lave LastUsed liste der er forbundet til en Guardian Retrieve

Funktion retrieveGuardianAutists Man skal kunne hente Guardian samt alle autister der er linket til denne guardian. Data Guardian Navn på guardian Autister

Funktion retrieveAutistSettings Man skal kunne hente en specifik autist. Data Autist Navn på autist Settings på autist

Funktion retrieveLastUsed Hente LastUsed liste fra en guardian Data Guardian LastUsed Update

Funktion updateAutistSetting Update setting på en bestemt autist

Funktion updateLastUsedGuardian Update en bestemt guardians LastLused Delete

Funktion deleteSettingAutist Slette en setting for en bestemt autist Funktion deleteLastUsedGuardian Slette LastUsed liste på en guardian

11.1 Project Backlog

Here is the full project backlog for the project.

Oasis Project Backlog

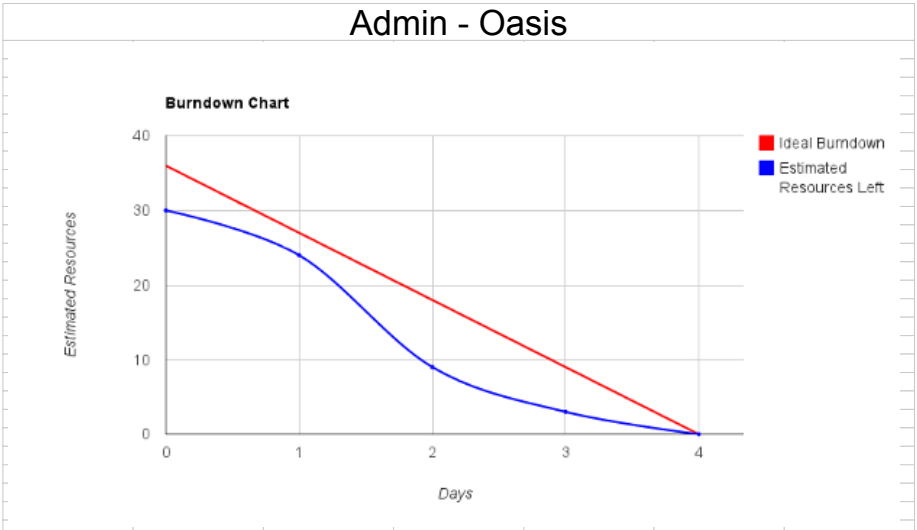
ID	Name	*Area*	Prioritet	Estimat	*Dependency*	How to demo	*Note*	*Status*
19	Sync with the online database	Server	1	10	Server	Enter data to the local db and sync with the online db. Open the online db and validate that the data is entered		Not started
27	Oasis app	App	5	30	OasisLib	Show the app		Done
1	Create table scheme for profiles	Database	5	2	None	CRUD via demo app		Done
2	Create table scheme for media	Database	5	2	None	CRUD via demo app		Done
3	Create table scheme for departments	Database	5	2	*None*	CRUD via demo app		Done
4	Create table scheme for certificates	Database	5	2	None	CRUD via demo app		Done
5	Create table scheme for a list of apps	Database	5	2	None	CRUD via demo app		Done
6	Create table scheme for apps	Database	5	2	None	CRUD via demo app		Done
7	Create view model for a profile	Model	3	1	Profile table	CRUD via demo app		Done
8	Create view model for a media	Model	3	1	Media table	CRUD via demo app		Done
9	Create view model for a department	Model	3	1	Department table	CRUD via demo app		Done
10	Create view model for a certificate	Model	3	1	Certificate table	CRUD via demo app		Done
11	Create view model for a list of apps	Model	3	1	List of apps table	CRUD via demo app		Done
12	Create view model for an app	Model	3	1	Apps table	CRUD via demo app		Done
13	Create a profile controller	Controller	4	3	Profile table Profile model	CRUD via demo app		Done
14	Create a media controller	Controller	4	3	Media table Media model	CRUD via demo app		Done
15	Create a department controller	Controller	4	3	Department table Department model	CRUD via demo app		Done
16	Create a certificate controller	Controller	4	3	Certificate table Certificate model	CRUD via demo app		Done
17	Create a list of apps controller	Controller	4	3	List of apps table List of apps model	CRUD via demo app		Done
18	Create an app controller	Controller	4	3	Apps table Apps model	CRUD via demo app		Done
20	Elaborate on the database design	Database	7	10	None	New database scheme		Done
21	Update the database tables	Database	10	5	None	CRUD via demo app		Done
22	Update the data models	Model	10	5	Database tables	CRUD via demo app		Done
23	Update the controller	Controller	10	5	Database tables, Data models	CRUD via demo app		Done
24	Database scheme	Database	10	5	None	Show the db scheme		Done
25	Settings model	Model	8	20	Database tables	CRUD via demo app		Done
26	Stats model	Model	8	20	Database tables	CRUD via demo app		Done
30	GunGame Server	Social	100	100	Savannah	Open CS and play		Done
31	Common report	Report	15	5	Earlier sections	Report pdf		Done
32	Update the database tables v0.3	Database	10	5	None	CRUD via demo app		Done
33	Update the data models v0.3	Model	10	5	Database tables	CRUD via demo app		Done
34	Update the controller v0.3	Controller	10	5	Database tables, Data models	CRUD via demo app		Done
35	Update and implement the hasControllers in the other helpers	HasControllers	10	10				Done
36	insertTag should return tagId	TagHelper	15	2				Done
37	attachTagsToMedia removeTagsAttachm insertMedia should return mediaId	MediaHelper	12	12				Done
38	profilesHelper should handle settings = null make pRole enum	ProfilesHelper	5	5				Done
39	loac should handle settings and stats = null	ListOfAppsControlle	5	5				Done
40	Add new fields to app model record Update getter/setter	AppModel	20	4				Done
41	Add new columns to appHelper columns insertApp should return appId	AppHelper	20	5				Done
42	Add columns icon, packageName, activityName to database	App DB	20	5				Done
43	auc iduser and cert should be primary key make aRole enum	AuthUsersDB	10	4				Done
44	Refactor local db code	Database	12	2				Done
45	Java doc	All	3	3				Done
46	Dummy data	All	20	5				Done
47	Correct commen report	Report	25	10				Done
48	Minor error correction in the library	Library	4	5				Done
49	Add extra functionality to the library	Library	4	5				Done
50	Create report structure	Report	2	10		Read the report		Done
51	Unit tests	Library	15	20		See the result of the test suite		Done
52	Usability	Oasis app	10	10		See the result of the usability test		Done
53	Report content	Report	5	200		Read the report		Done
54	Improve Oasis App	Oasis app	5	10	Oasis lib	Show app		Not started

Figure 11.1: An overview of the Oasis project backlog.

11.2 Burndown Charts and Sprint Backlogs

Here are an overview of all the sprints in this project.

Oasis Burndown Chart/Sprint1 Backlog



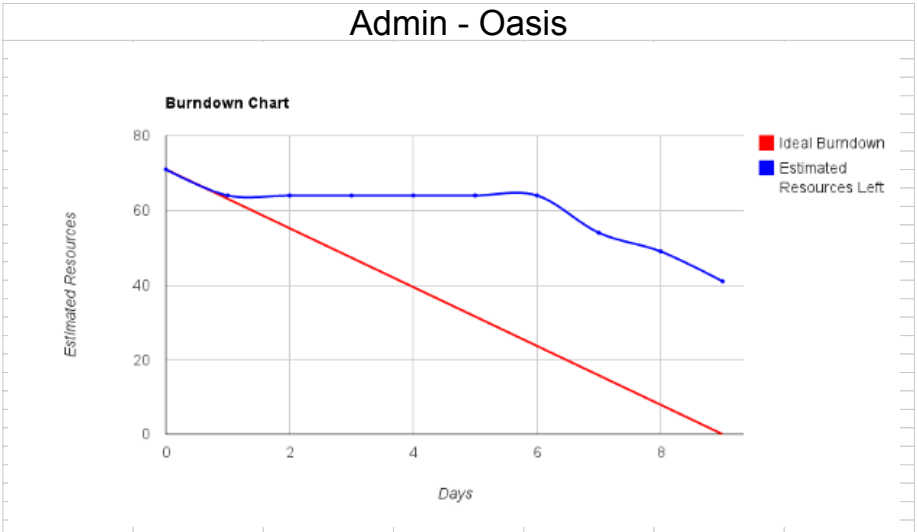
ID	Name	Prioritet	Estimat	How to demo	Note	Status
1	Create table scheme for profiles	5	2	CRUD via demo app		Done
2	Create table scheme for medias	5	2	CRUD via demo app		Done
3	Create table scheme for departments	5	2	CRUD via demo app		Done
4	Create table scheme for certificates	5	2	CRUD via demo app		Done
5	Create table scheme for a list of apps	5	2	CRUD via demo app		Done
6	Create table scheme for apps	5	2	CRUD via demo app		Done
7	Create view model for a profile	3	1	CRUD via demo app		Done
8	Create view model for a media	3	1	CRUD via demo app		Done
9	Create view model for a department	3	1	CRUD via demo app		Done
10	Create view model for a certificate	3	1	CRUD via demo app		Done
11	Create view model for a list of apps	3	1	CRUD via demo app		Done
12	Create view model for an app	3	1	CRUD via demo app		Done
13	Create a profile controller	4	3	CRUD via demo app		Done
14	Create a media controller	4	3	CRUD via demo app		Done
15	Create a department controller	4	3	CRUD via demo app		Done
16	Create a certificate controller	4	3	CRUD via demo app		Done
17	Create a list of apps controller	4	3	CRUD via demo app		Done
18	Create an app controller	4	3	CRUD via demo app		Done

Figure 11.2: The burndown chart and sprint backlog from sprint 1.

Oasis Burndown Chart/Sprint2 Backlog

ID	Name	Prioritet	Estimat	How to demo	Note	Status
9	GunGame Server	100	100	Open CS and play		*Done*
1	Update the database tables	10	5	CRUD via demo app		*Done*
2	Update the data models	10	5	CRUD via demo app		*Done*
3	Update the controller	10	5	CRUD via demo app		*Done*
6	Database scheme	10	5	Show the db scheme		*Done*
4	Settings model	8	20	CRUD via demo app		*Done*
5	Stats model	8	20	CRUD via demo app		*Done*
9	Elaborate on the database design	7	10	New database scheme		*Done*
9	BMI app	5	30	Show the app		*Continued in next sprint*
7	Server xml language	1	1	No demo		*Not started*
8	SSL test app	1	20	Show the app		*Not started*

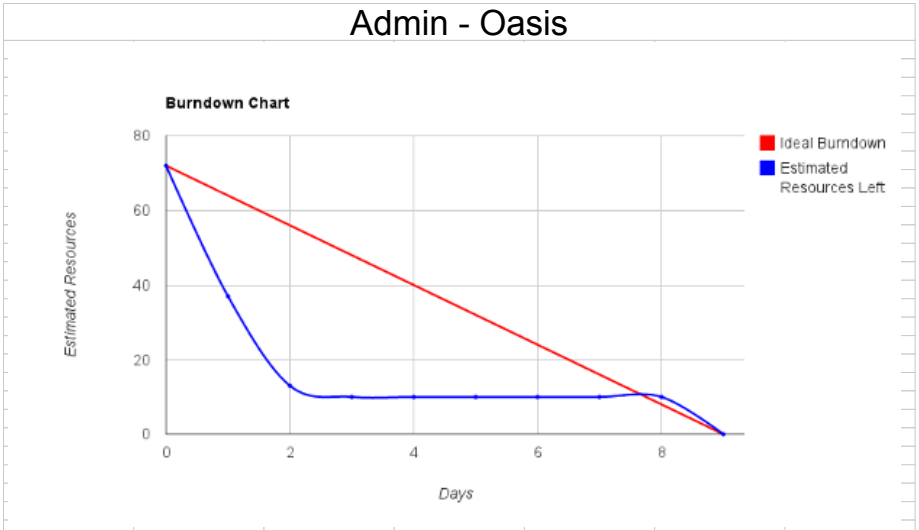
Figure 11.3: The burndown chart and sprint backlog from sprint 2.



ID	Name	Prioritet	Estimat	How to demo	*Note*	*Status*
31	Common report	15	5	Earlier sections	*Report pdf*	*Done*
32	Update the database tables v0.3	10	5	None	*CRUD via demo app*	*Done*
33	Update the data models v0.3	10	5	Database tables	*CRUD via demo app*	*Done*
34	Update the controller v0.3	10	5	Database tables, Data models	*CRUD via demo app*	*Done*
27	BMI app	5	30	OasisLib	*Show the app*	*In progress*
28	Server xml language	1	1	None	*No demo*	*Not started*
29	SSL test app	1	20	OasisLib	*Show the app*	*Not started*

Figure 11.4: The burndown chart and sprint backlog from sprint 3.

Oasis Burndown Chart/Sprint4 Backlog



ID	Name	Prioritet	Estimat	How to demo	Note	Status
35	Update and implement the hasControllers in the other helpers	10	10			Done
36	insertTag should return tagId	15	2			Done
37	attachTagsToMedia removeTagsAttachr insertMedia should return mediaId	12	12			Done
38	profilesHelper should handle settings = null make pRole enum	5	5			Done
39	loac should handle settings and stats = null	5	5			Done
40	Add new fields to app model record Update getter/setter	20	4			Done
41	Add new columns to appHelper columns insertApp should return appId	20	5			Done
42	Add columns icon, packageName, activityName to database	20	5			Done
43	auc iduser and cert should be primary key make aRole enum	10	4			Done
44	Refactor local db code	12	2			Done
45	Java doc	3	3			Done
46	Dummy data	20	5			Done
47	Correct commen report	25	10			Done

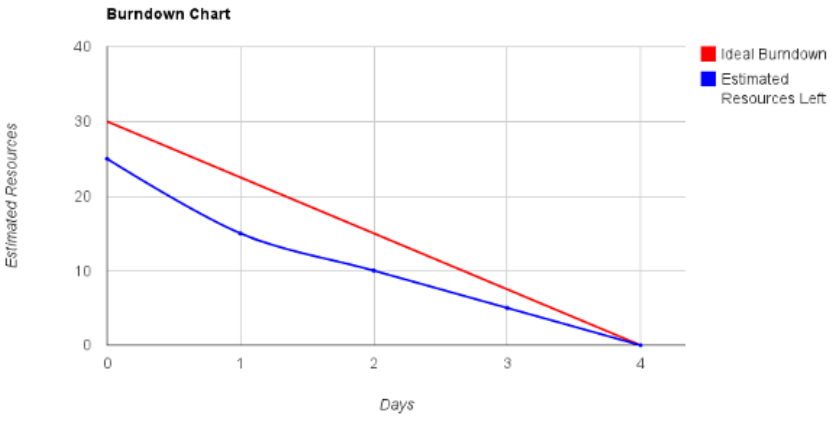
Figure 11.5: The burndown chart and sprint backlog from sprint 4.

Oasis Burndown Chart/Sprint5 Backlog

ID	Name	Prioritet	Estimat	How to demo	*Note*	*Status*
27	Oasis app	5	30		*Show the app*	*In progess*
19	Sync with the online database	1	10		*Enter data to the local db and sync with the online db. Open the online db and validate that the data is entered*	*Not started*
48	Minor error correction in the library	4	5			*Done*

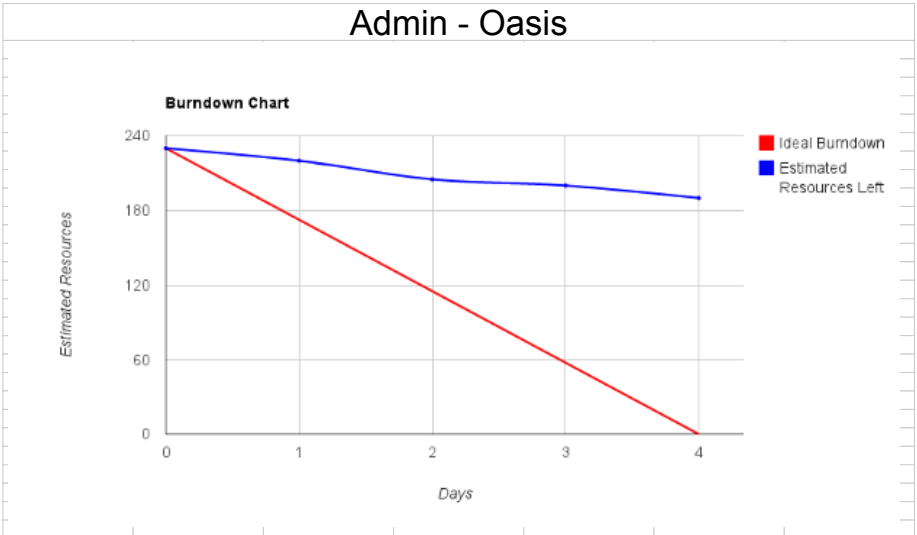
Figure 11.6: The burndown chart and sprint backlog from sprint 5.

Admin - Oasis



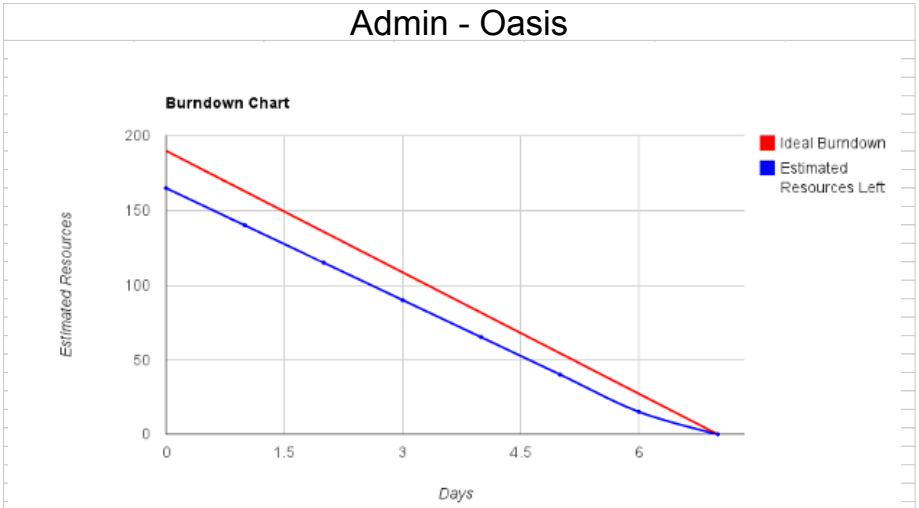
ID	Name	Prioritet	Estimat	How to demo	Note	Status
27	Oasis app	5	15		*Show the app*	*Done*
49	Add extra functionality to the library	4	5			*Done*
50	Create report structure	2	10	Read the report		*Done*

Figure 11.7: The burndown chart and sprint backlog from sprint 6.



ID	Name	Prioritet	Estimat	How to demo	Note	Status
51	Unit tests	15	20	See the result of the test suite		*Done*
52	Usability	10	10	See the result of the usability test		*Done*
53	Report content	5	200	Read the report		*In progress*

Figure 11.8: The burndown chart and sprint backlog from sprint 7.



ID	Name	Prioritet	Estimat	How to demo	Note	Status
53	Report content	5	190	Read the report		Done

Figure 11.9: The burndown chart and sprint backlog from sprint 8.

11.3 Change Log

Here is the full change log for the Oasis Library – along with the models used in it – and the Oasis Local Database.

Oasis Changelog

OasisLib version 0.8

- Minor bug fixed to the controllers

OasisLib version 0.7

- Minor bug fixed to the controllers

OasisLib version 0.6

- Controllers
    - Added AppsHelper.removeApp
    - Renamed AppsHelper.modifyAppSettingsByProfile to AppsHelper.modifyAppByProfile
    - Added AuthUsersController.removeAuthUser
    - Added DepartmentsHelper.removeDepartment
    - Added HasDepartmentController.removeHasDepartmentByDepartmentId
    - Added HasDepartmentController.removeHasDepartmentByProfileId
    - Added HasGuardianController.removeHasGuardianByProfile
    - Added HasGuardianController.removeHasGuardian
    - Added HasLinkController.removeHasLinkByMediaId
    - Added HasLinkController.removeHasLinkBySubMediaId
    - Added HasSubDepartmentController.removeHasSubDepartmentBySubDepartmentId
    - Added HasSubDepartmentController.removeHasSubDepartmentByDepartmentId
    - Added HasTagController.removeHasTagByTagId
    - Added HasTagController.removeHasTagByMediaId
    - Added ListOfAppsController.removeListOfAppsByProfileId
    - Added ListOfAppsController.removeListOfAppsByAppId
    - Added MediaDepartmentAccessController.removeMediaDepartmentAccessByMediaId
    - Added MediaDepartmentAccessController.removeMediaDepartmentAccessByDepartmentId
    - Added MediaHelper.removeMedia
    - Added MediaHelper.getMyPictures
    - Added MediaHelper.getMySounds
    - Added MediaHelper.getMyWords
    - Added MediaProfileAccessController.removeMediaProfileAccessByProfileId
    - Added MediaProfileAccessController.removeMediaProfileAccessByMediaId
    - Added ProfilesHelper.removeProfile
    - Added ProfilesHelper.getGuardians
    - Added ProfilesHelper.getChildren
    - Added ProfilesHelper.getCHildrenWithNoDepartment
    - Added ProfilesHelper.getGuardiansWithNoDepartment
    - Added ProfilesHelper.getGuardiansByChild
    - Added TagsHelper.removeTag

Figure 11.10: The change log for the library and database.

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 11.11: Invitation sent to the test persons of the usability test.

11.4 Mail correspondence with Customer

11.4.1 Mail To Customer

Hej Kristine

Vi er blevet tildelt dig som kontakt person i forbindelse med vores projekt. Som nævnt sidst så arbejder vi på at udvikle applikationer til android, som kan bruges enten af jer som pædagoger og måske af autisterne på sigt. Vi vil udvikle flere forskellige applikationer, og vi vil gerne løbende aftale møder med dig, hvor vi kan vise det samlede produkt som er lavet. På den måde kan vi få feedback på hvad der går godt og hvad der er knap så godt.

Vores udviklings gruppe består af tre personer og vi skal lave en applikation der kan hjælpe med at lave profiler der passer til børnene.

Da vi stadig kun er i gang med at planlægge mener vi ikke at det er nødvendigt at holde et møde endnu. Men vi har nogen spørgsmål som vi gerne vil have dig til at svare på:

Hvilke informationer gemmer i omkring det enkelte barn?

- Journal nummer?
- Person nummer?
- Navn?
- Alder?
- Særlige behov?

a - Ur

a1. Vil barnet kunne forstå at en hel cirkel kan have forskelligt tidsinterval? a1.1 Eller er det bedst hvis cirklen har et fast tidsrum fx 1 time?
a2. Hvis man skal måle et tidsinterval på uret, er det så bedst at lade uret efterligne et almindeligt ur med 12 timer eller et stop-ur med kun 1 time?

b - Timeglas

b1. Vil barnet kunne forstå at det samme timeglas med den samme mængde sand kan varierer i tid?
b2. Er det bedst at man varierer i mængden af sand i timeglasset eller at man varierer i timeglassets størrelse?

c - Aktivitetstid

c1. Vil barnet kunne forstå at en linje der går hele vejen hen over skærmen kan varierer i tidsinterval? c1.1 Eller er det bedre hvis linjen har et fast tidsinterval og fx en halv linje derfor svarer til en halv time og en hel linje til en hel time?

d - Dagsplan

d1. Hvis man laver en visuel dagsplan er det så bedst at man laver et interval som viser tiden imellem to aktiviteter, eller at man viser alle aktiviter i løbet af dagen kombineret med en tidslinje?

11.4.2 Mail From Customer

Hej. Tak for jeres mail. Jeg skal besvare jeres mail så godt som muligt, og så må i give lys hvis i har brug for at jeg uddyber.
Vedr. informationer vedr. barnet: Vi benytter et elektronisksystem som hedder, EKJ, hvor alle oplysninger på børnene er gemt. Det vil sige, pers. nr., adresse oplysninger, indbydelser, handleplaner og referater fra diverse møder.
UR: Hvis det er tydeligt vist at "tiden går" /skiven bliver mindre/forsvinder, som tiden går, vil barnet forstå meningen med uret. For at indikere forskellig tid, kan man benytte forskellige farvet baggrunde. Lilla:5 min. Grøn:10 min osv. Vi benytter kun kortere tidsintervaller,(1. min. 3. min. 5 min. -op til ca. 10-15. min) da 1 time er for abstrakt.
Timeglas: Hvis der er en tydelig markering af tidsintervallet, som beskrevet ovenfor, er det muligt at bruge samme timeglas. Tror det vil give bedst forståelse for barnet, hvis mængden af sand varieres efter tid.
Aktivitetstid og dagsplan: (Tror J) Aktivitetstid kan bruges ved, at tiden bliver indikeret af mængden af aktiviteter. - Altså 3-5 viste aktiviteter af gangen, og ikke så meget om det er en time eller 15 min. Tiden kunne være en mulighed at tilføre, om nødvendigt. Mange af vores børn har manglende fornemmelse for tid, og ofte har de brug for at se små konkrete sekvenser/beskeder frem for mange over længere tid. Derfor vil jeg tror de bedst kan overskue ½ dag af gangen, men stadig have mulighed for at have dagen på skemaet, hvor det kan vises i sekvenser.
Jeg har samlet de to ovenstående punkter, da de nemt kommer til at gribe ind i hinanden. Vores ugeskemaer i børnehaven er vist med internationale farver, dem vil i ligeledes kunne benytte til at tydeliggøre ugedagene. Mandag: Grøn, Tirs.: Lilla, Ons.: orange, tors.: blå, Fre.: gul, lør.: rød og søndag: hvid.
Håber dette er uddybende nok, ellers må i gerne skrive eller ringe til mig hvis

det er nemmere.
Ser frem til at hører fra jer igen.

11.5 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.

11.6 Usability Documents

Briefing

Goddag og velkommen til denne brugervenlighedsundersøgelse.

Vi vil gerne starte med at takke dig for, at du vil hjælpe os med at gennemføre denne brugervenlighedsundersøgelse. Vi læser op fra dette dokument for at sikre os, at alle personer som deltager i vores studie for samme introduktion. Hvis du har spørgsmål undervejs, er du naturligvis meget velkommen til at stille disse spørgsmål.

Vi har i dette semester bygget et system til Android til at hjælpe børn med autisme og deres pædagoger og forældre, og det er nu nået til et stadie hvor vi gerne vil teste systemet. Denne test handler udelukkende om at finde problemer og mangler i systemet, og ikke om at teste jeres viden af systemet, så alle tanker I må have om produktet vil vi meget gerne høre.

Før vi starter første del af testen, vil jeg bede dig om at underskrive denne samtykkeerklæring for at sikre, at du er indforstået med rammerne for studiet. Derudover skal du også svare på et demografisk spørgeskema inden testen går i gang.

Testen består af fire dele:

- Test af applikationer (20 min)
- De-briefing og spørgeskema (5 min)
- Test af Administrations applikation og web applikation (20 min)
- De-briefing og spørgeskema (5 min)

Undervejs vil der være en pause.

I de to tests vil du blive stillet en række opgaver som du skal løse. Læs opgaveformuleringen grundigt og fortæl så test hjælperen hvad du mener opgaven går ud på. Derefter skal du forsøge at løse opgaven så godt som muligt. Opgaverne skal løses i den rækkefølge de står således at du starter med opgave 1 og arbejder dig ned af.

Det er meningen at du skal tænke højt mens du løser opgaverne. Dvs. at du siger hvad du har tænkt dig at gøre for at løse opgaven, hvilke ting du synes virker uklare eller komplicerede og hvordan du tror systemet virker. For eksempel vil det være godt hvis du nævner hvad du forventer en knap gør inden du trykker på den.

Når testen er færdig vil der være nogle afsluttende spørgsmål som du skal besvare omkring hvordan du synes testen er forløbet og hvad din opfattelse af systemet er.

Figure 11.12: The document used to brief and de-brief the test persons.

11.6.1 Questionnaires from usability

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

__ Kvinde X Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Wombat:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
X Lettere erfaren
__ Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
_X_ Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
_X_ Middel
__ Svær
__ Meget svær

Wombat:

__ Meget let
__ Let
_X_ Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
_X_ Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
_X_ Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

__ Ja X Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
__ Middel
X Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
__ Middel
X Svær
__ Meget svær

Wombat:

__ Meget let
X Let
__ Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
X Let
__ Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Wombat:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Usabilitytest - Spørgeskema

1. Hvilket køn er du?

X Kvinde __ Mand

2. Hvor erfaren vurderer du at du selv er med computerer? (vælg en)

__ Meget begrænset erfaring
__ Lettere erfaren
X Forholdsvis erfaren
__ Meget erfaren

3. Har du brugt en tablet før (f.eks. en iPad)?

X Ja __ Nej

4. Tror du at en tablet med de rigtige programmer vil kunne forbedre din arbejdsgang?

X Ja __ Nej

5. Tror du at en tablet med de rigtige programmer vil kunne forbedre børnenes hverdag?

X Ja __ Nej

Hvor let mener du at applikationerne var at bruge?

GIRAF:

__ Meget let
__ Let
X Middel
__ Svær
__ Meget svær

Parrot:

__ Meget let
X Let
__ Middel
__ Svær
__ Meget svær

Wombat:

X Meget let
__ Let
__ Middel
__ Svær
__ Meget svær

Admin:

__ Meget let
X Let
X Middel
__ Svær
__ Meget svær

Hvor let mener du at Web interfacet var at bruge?

Meget let
__ Let
__ Middel
X Svær
__ Meget svær

11.7 Usability Assignments

Admin applikation er en applikation der skal bruges til at styre informationer fra hver enkelt institution. Dette kan gøre ved at manipulere data omkring institutioner og brugere.

Usability Opgaver:

- Opret en ny barne profil med følgende informationer:
  - navn: Thomas Thomasen
  - telefonnummer: 12345678
  - Afdeling: Myretuen
- Få vist den nu oprettede barne profils informationer.
- Find min profil og rediger navnet på profilen fra Tony Stark til dit eget navn.
- Tilføj den nu oprettede Thomas Thomasen til mine børn.
- Tilføj Applikationerne Wombat og Parrot til barnet Thomas Thomasen.
- Fjern Thomas Thomasen fra afdelingen Myretuen.
- Tilføj Thomas Thomasen til afdelingen Bikuben.

11.8 Unit Test Results

In this section the results for the remaining unit tests can be found.

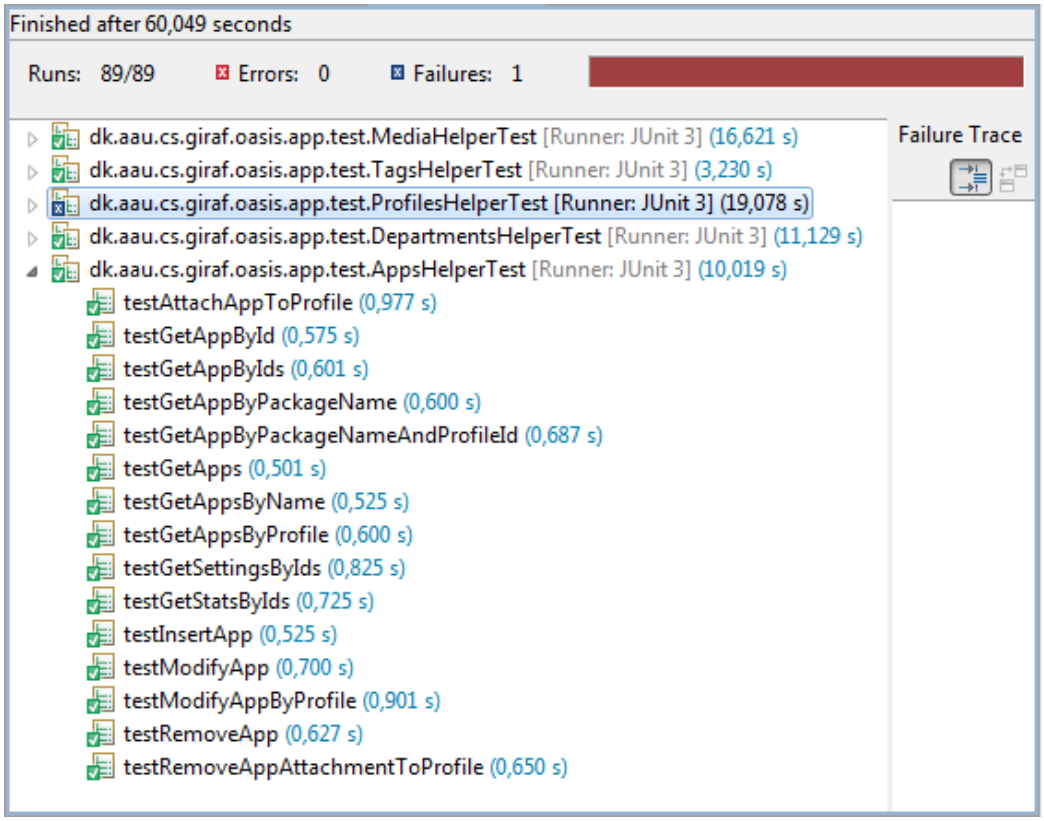


Figure 11.13: The result from the `appsHelper` tests.

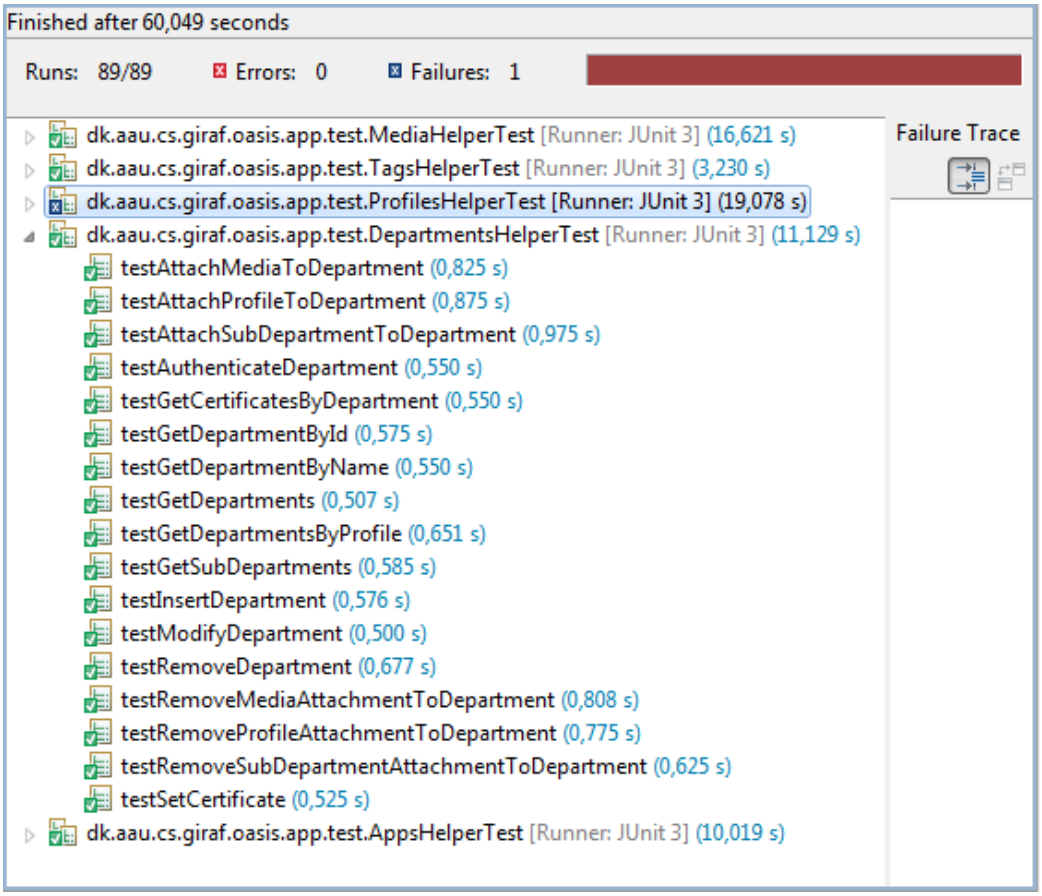


Figure 11.14: The result from the `departmentHelper` tests.

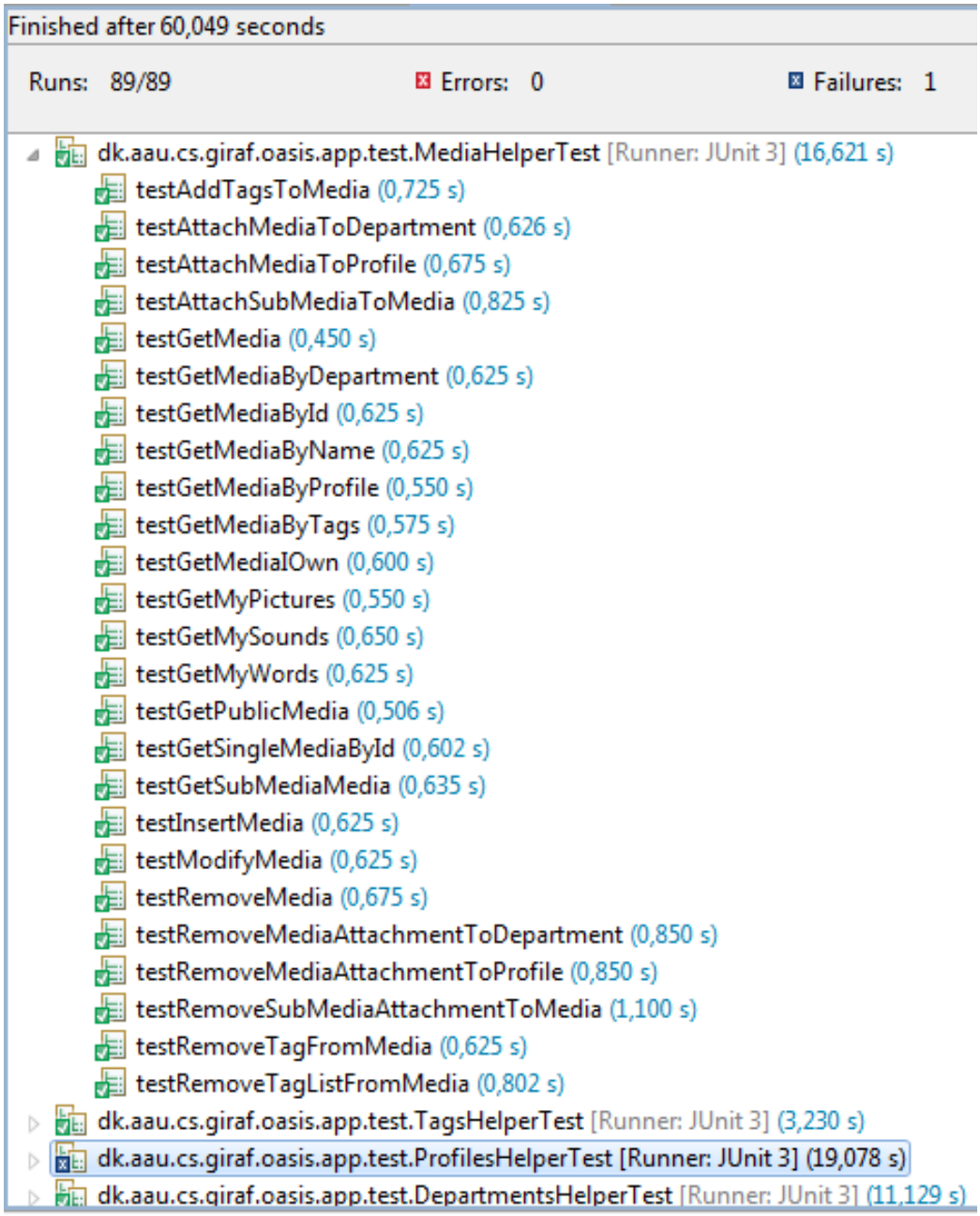


Figure 11.15: The result from the `mediaHelper` tests.

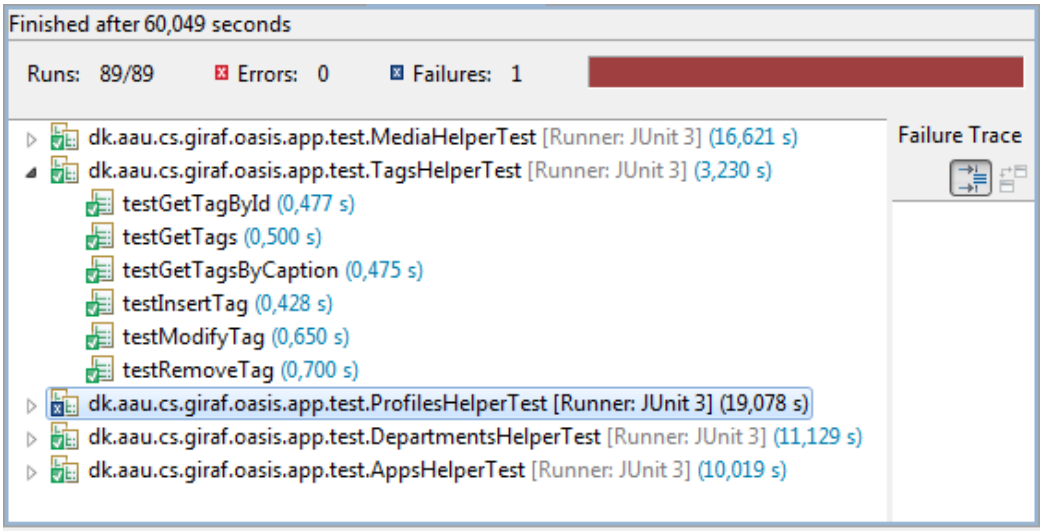


Figure 11.16: The result from the `tagsHelper` tests.

Bibliography

[All11a] Scrum Alliance. Advice on conducting the scrum of scrums meeting. http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting, 2011.

[All11b] Scrum Alliance. Scrum alliance. http://www.scrumalliance.org/, 2011.

[Gra02] Temple Grandin. Teaching tips for children and adults with autism. http://www.autism.com/ind_teaching_tips.asp, December 2002.

[IEE93] IEEE. Unit testing. http://aulas.carlosserrao.net/lib/exe/fetch.php?media=0910:1008-1987_ieee_standard_for_software_unit_testing.pdf, December 1993.

[JB11] Steffan Bo Pallesen Jacob Bang, Lasse Linnerup Christiansen. Administration module for giraf. http://people.cs.aau.dk/ ulrik/Giraf/Admin.pdf, May 2011.

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

[Mic12] Microsoft. Model view controller. http://msdn.microsoft.com/en-us/library/ff649643.aspx, May 2012.

[Pat] Ron Patton. *Software Testing*.

[Sam] Samsung. Samsung tablet. http://www.samsung.com/global/microsite/galaxytab/10.1/index

[SQL12a] SQLite. About sqlite. http://www.sqlite.org/about.html, May 2012.

[SQL12b] SQLite. Sqlite. http://www.sqlite.org/, May 2012.

[SQL12c] SQLite. Sqlite datatypes. http://www.sqlite.org/datatype3.html, May 2012.

[Tea12a] Android Development Team. Android 4.0.3 platform. http://developer.android.com/sdk/android-4.0.3.html#relnotes, March 2012.

[Tea12b] Android Development Team. Android architecture. http://developer.android.com/guide/basics/what-is-android.html, March 2012.

[Tea12c] Android Development Team. Cursor. http://developer.android.com/reference/android/database/Cursor.html, March 2012.

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

[Wel09] Don Wells. extreme programming. http://www.extremeprogramming.org/rules.html, 2009.

This page is left blank for the purpose of containing the attached CD-ROM.