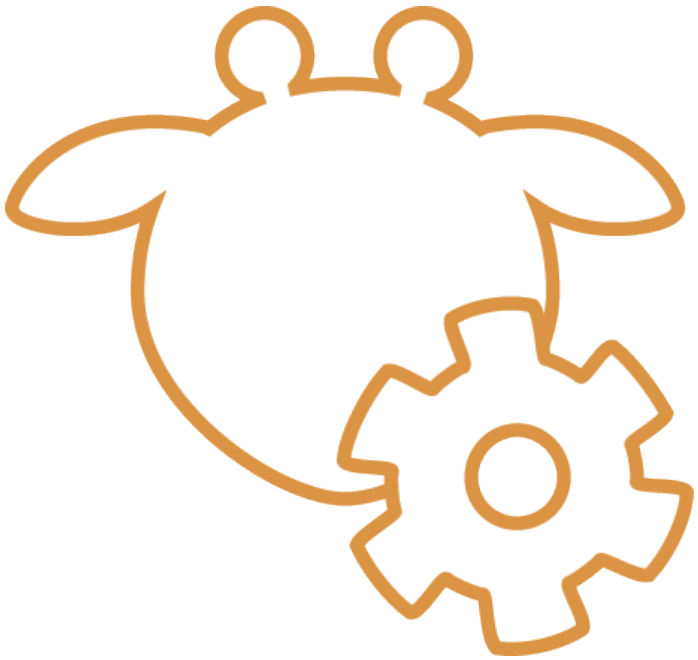
GIRAF - Admin



SW6 Projekt
Group SW601F13
Department of Computer Science
Aalborg University
The 4th of June 2013

AALBORG UNIVERSITY
STUDENT REPORT

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

Title:

GIRAF - Admin

Theme:

Developing Complex Software Systems

Project period:

P6, Spring Term 2013

Project group:

SW601F13

Participants:

Jens M. Lauridsen
Johan Sørensen
Lars Chr. Pedersen
Tommy Knudsen

Supervisor:

Shuo Shang

Circulation: 7

Page count: 131

Appendix count and type: 3, Installation Guide / Configuration, MockUps, Burndowncharts

Finished on Juni 4^{th} 2012

Synopsis:

TO BE MADE

Signatures:

Tommy Knudsen

Lars Chr. Pedersen

Johan Sørensen

Jens Mikkel Lauridsen

Contents

Introduction

This project is part of the multi-project called GIRAF, which stands for: "Graphical Interface Resources for Autistic Folk" which have been an active project at Aalborg University since 2011.
As the name suggests this system is a tool made for helping kids with autism to communicate. The system is to be actively used in the specialized kinder gardens and schools here in Aalborg.
This report contains first an introduction to the GIRAF project as a whole, which describes the GIRAF project as it is of the day this report was written. This introduction is contained in all project reports for the GIRAF multi-project, and can therefore be skipped if the reader already is familiar with the GIRAF project as of year May 2013. The place to be skipped to is then part II on page 35.
Second this report also contains the thoughts that went into constructing the new administration system, as well as some in depth going explanation of the implementations, for the GIRAF system.
The administration system makes it possible to edit and create users for the GIRAF system and in its finished form is possible to be controlled both from a tablet and from a PC over the internet. It also comes with the possibility to make, edit and manage pictograms on a PC over the internet.

In this report we also use some terms that often is misunderstood, as well as a few special words, we therefore want to clarify them here before we begin. The wording is taken from wikipedia [1].

Word	Meaning
Website	A set of related webpages served from a single web domain
Webpage	A single page displayed on a website
Homepage	The initial or main web page of a website
Site	Refers to the word website
Pictogram	A special element, developed for the GIRAF project. Containing sound, text and image
Guardian	A person with responsibility for a child with autism, either a pedagog or a parent

Table 1: Word Explanation

Part I

GIRAF Introduction

Chapter 1

The GIRAF Project

Graphical Interface Resources for Autistic Folk (GIRAF) started out in 2011 as a semester project targeting children with autism and their guardians. In the following chapter, the overall vision for the GIRAF project will be presented, the projects from previous years will be explained briefly along with the platform for the project. Lastly a section describing autism is included.

1.1 Vision for Giraf

The vision for GIRAF is to create a multi-purpose application based on *Android* that can simplify and ease the lives of autistic children and their guardians.

The purpose of GIRAF is to replace physical items that are being used daily by the children and their guardians with digitized versions. The idea being to gather several functionalities in one object and allowing customization for each individual child.

This will also optimize work procedures on the individual institution in such a way, that guardians will save time doing repetitive tasks such as making pictograms. This time could be spent with the children instead.

As of the spring of 2013 three schools and institutions for children with autism in Northern Jutland are involved in the development, but the hope is that GIRAF will be distributed across all similar institutions in Denmark.

1.2 Previous years

During the first year of development, four parts of the GIRAF project were developed. The four projects were developed during the spring semester of 2011 and included the projects:

Admin An administration interface used for administrating different aspects of the GIRAF system.

DigiPECS A digitized version of "Picture Exchange Communication System"[10] a system used as an aid for communication with people with special needs such as autism.

Launcher A home screen application and distribution platform for Android.

aSchedule A visual schedule for the Android platform.

During the spring semester of 2012, five new software groups continued development of the GIRAF project. The projects developed during 2012 were:

Launcher An enhancement of the launcher project developed during the spring semester of 2011.

Oasis An enhancement of the admin project from 2011. Furthermore the Oasis project developed a local database for the GIRAF system.

Parrot An enhancement of the DigiPECS project from 2011. The project was renamed because of trademark issues.

Savannah A server side database with web interface for the GIRAF system.

Wombat An Android application for measuring and visualizing time.

During the spring semester of 2012 two databases were developed, however synchronization between them was never achieved.

Problems with Initial Implementation

As the spring semester of 2013 started, an "install party" for the students was held. The party was intended to help the students compile and deploy the projects from 2012.

Even though a representatives for each of the 2012 groups were present, some compilation problems still occurred.

The repository used for distributing in 2012, was disorganized and difficult to navigate, i.e. due to:

- Multiple copies of the same project.
- Unclear dependencies among the different project.
- Projects only meant to be compiled from Eclipse for Windows.

During the following week a working workspace was created and shared with the rest of the students, along with install instructions. The install instructions were later updated to a more clear edition.

1.3 Target Platform

Android is an open-source operating system originally developed by Android Inc, and later bought by Google Inc. The first release came in 2007, where it was launched by Google Inc. together with Open Handset Alliance (OHA), which includes companies such as Samsung, HTC, LG and Google.

Before the first students where involved in the project in the spring of 2011 Ulrik Nyman considered two platforms for the development of the project. The Android and iOS platforms. The Android platform was chosen for three main reasons:

- That the platform is open source.
- That in Android the developers can take control of the functionality of the home button.
- That distribution of the software is possible outside the official marketplace.

For the the two following years it has been chosen to stay on the Android platform. This is done both to be able to reuse the source code and because Android compatible hardware is available for the students. In the very long tern the system could support multiple platforms.

1.4 Autism

Autism is a spectrum disorder, meaning that it appears in different variants and not all people who are diagnosed have the same symptoms. The disorder can often be observed within the first three years of a child's life. Autism is a physical condition and is linked to abnormal chemistry in the brain, however the exact causes of these abnormalities are still unknown.[3]

Symptoms

Children with autism usually have difficulties understanding the concept of "play pretend", meaning that they have a hard time imitating the actions of others when playing and therefore prefer to play alone. Furthermore they have difficulties with social interaction and communication – verbally and non-verbally.

People diagnosed with autism may;

- Be very sensitive to light, noise, touch, and taste.
- Have a hard time adjusting to new and changing routines.
- Show unusual attachments to objects.

Autism diagnosed individuals may have a hard time starting and maintaining a conversation. They may communicate with gestures instead of words, develop language slower or faster than normal and some do not develop any language at all. Furthermore the lack of social interaction means they might have a hard time making friends, may be withdrawn and may avoid eye contact.[3]

Signs and tests

If a child fails to meet any of the following language milestones, it may be an indication that it needs to be tested for autism;

- Babbling by 12 months.
- Gesturing (such as pointing or waving goodbye) by 12 months.
- Saying single words by 16 months.

Children failing to meet any of the previous mentioned language milestones might receive a hearing evaluation, a blood test and a screening test for autism. Since autism covers a broad spectrum of symptoms, a single brief evaluation cannot predict what abilities the child has. Therefore a range of different skills are evaluated, such as:

- Communication
- Language
- Motor skills
- Speech
- Success at school
- Thinking abilities

Some parents might be scared of having their child diagnosed, however without a diagnosis, the child might not get the necessary help.[3]

Treatment

Autism cannot be cured, however an early diagnosis and treatment can greatly improve the child's quality of life. Different treatment programs usually build on the child's interests and are highly structured to their needs and routines.[3]

Chapter 2

The GIRAF project 2013

When working in a multi-project consisting of eight groups, it is important to have a common goal for the project. This chapter describes this goal as a story. Furthermore the chapter includes description of the development process and the rules of conduct.

2.1 The Goals for 2013

Within the first couple of weeks, when all the groups had been assigned a project, a major story for the overall project was written.

The Major Story for 2013

> "The guardian arrives at the institution, and turns on the tablet. The guardian is aware of the arrival of a new child at the institution after lunch. The guardian sets up and customizes a profile for the child, this includes creation of new pictograms. Furthermore the guardian prepares games and a life story for the child.
>
> After lunch the new child and the guardian meet. The child is introduced to the communication tool Parrot. After some introduction they sit down to do some communication practice using the tool.
>
> Afterwards the child wants to go outside to see the rest of the institution, and needs to put on some outdoor clothes. The guardian introduces the child to the Zebra tool, and together they put on the child's outdoor clothes.
>
> When the child comes back in, the guardian and the child play the games prepared earlier by the guardian.
>
> When they are done playing the child and the guardian read the child's life story using Tortoise."

2.2 Definition of a Multi-project

A multi-project is a project that includes multiple groups that each work on their own sub-project, which is part of a larger project. In this case, the larger project is the GIRAF system and each group works on a separate part of the system.

Compared to working on a single project in isolation, working together creates new challenges. The software produced by each group has to be integrated to ensure the entire system works properly. Some projects are more independent of the rest, while others depends heavily on some projects like the database project Wasteland described in Section 3.2.7. Groups have to be flexible and pass any requirements to other groups' projects early to prevent halts.

To ensure the project is successful and no misunderstandings occur, there must be good communication and cooperation between the groups. This requirement is amplified by the fact that there are no definitive authoritative figures, other than those chosen by project members.

2.3 Group and Work Structure

This section describes the development methods used during the spring semester of 2013, including stories and project management tools.

The section is rounded off by a description of the development tools used, including Redmine, Git, and Jenkins.

2.3.1 Development Method

Having a development method is one of the main ways to structure the work process of a project. A development method is a collection of methods and structures, from the way to have meetings, gathering requirements and structuring the development. There are many development methods, each is structured and handles issues differently, however, it is rare that one fits a development problem perfectly. Different methods are often combined and customized to fit the problem at hand.

Implemented Development Methods

This project's nature calls for agile development, due to team collaboration, user feedback, product focus, and continuous integration. Agile development focuses on a flexible but structured work progress suited for projects with many unknown variables. The agile development method has the ability to adapt to changing requirements throughout the project and focuses on having a shippable product at the end of each iteration.

Stories

User stories is one of the tools that helps streamline the work process, keeps focus on a shippable product and is the main component for management of the project. First of all the product story works as a common problem statement for all work groups. A product story is the agreement on what is necessary for the product to be finished. From product story each group can extract what is required of them to complete the story.

Management

The semester coordinator, Ulrik Nyman, has supervised the project since it's beginning. Ulrik Nyman himself has a child with autism and will continue being a part of the project for the time to come, conveying his knowledge of the development process and the product. To help fit the product to the needs of guardians, for which the product is intended, a number of representatives are included for more detailed feedback on the process and the product.

To keep as many work hours in development and to keep a good overall management, common meetings were held weekly. The common meetings had focus on sprints and team cooperation. Problems that needed further discussion and/or development were discussed by a committee consisting of a few representatives from each group.

The common meeting and committee meeting are further specified in sections Section 2.4.1 and Section 2.4.3.

2.3.2 Development Tools

A number of tools were used in order to optimize team collaboration and to make the projects more accessible. These tools will be further explained in the following sections.

A dedicated Linux server was commissioned for the entire GIRAF project and several services installed to facilitate collaboration and agile development. Common to all current services are their free, open-source nature and support of LDAP authentication, allowing all students and supervisors to log in using their AAU credentials.

Redmine

Several tools were audited for use in the project management aspect of development, including Trac, PivotalTracker and Github. Redmine, a Ruby-On-Rails web application, was selected owing primarily to its support of multiple projects and support features such as wikis, forums, milestones and various charts. The features most broadly used will briefly be described here.

Projects All projects live in a shared project space, and can be placed in a hierarchy under a super project. In this regard, the primary multi project served as the base of each of the eight groups' underlying projects.

Issue handling Redmine's primary feature is its issue handling. Project members can create and react to issues within custom-defined domains. For GIRAF, this was primarily development tasks, but could just as well be used for report-related tasks or general maintenance in an attempt to manage time usage.

Burndown Charts Redmine does not have native support for burndowns, but does support it through a Free and Open-Source Software (FOSS) third-party plugin. Burndowns are a visual aid of each subproject's progress throughout a sprint, giving quick summary of development speed and whether proactive action may need to be taken.

Milestones A generic milestone feature in Redmine is Versions. Versions are simply markers with a set date, and can be open or closed for attachment of issues. The burndown plugin couples a version's end date with attached issues and their progress to generate the related charts.

Wiki A per-project wiki module exists in Redmine. The basic wiki markup has been expanded to allow referencing of almost any other element in the project hierarchy, such as projects, issues, files and VCS revision.

Redmine has many more features not directly applied during this project period. However, many could be applied to create a more centralised and structured development experience in future projects. Examples include file and document hosting, advanced issue workflows, permission management and VCS integration. Future multiprojects may consider expanding into these fields if they feel proficient in Redmine's basic usage.

Version Control System

The university's IT services offers only a single version control system, Subversion. Although centrally supported and backed up regularly, Subversion's shortcomings were challenged before main development had begun. Most notably, the system's centralised workflow and high operation cost. Many of SVN's actions require access to the central server. Two alternatives without these issues were suggested: Git and Mercurial (Hg). The former was chosen as a general question of broad platform support and popularity. A primary strength of these systems is their support of separate branches of development without the constant need to connect to a central server. This allows developers of each project to synchronize with a main branch while maintaining several development branches on their own workstation.

Most groups used Github as hosting solution for development of their projects, as a git hosting solution was not immediately forthcoming (contrary to Subversion and Mercurial, Git does not have a default server implementation). At the conclusion of the project period, a solution was configured using Apache-based LDAP authentication, deferring authorisation and repository management to Gitolite, a low-footprint open-source offering.

In the interest of easier cross-project code contribution and inspection, an improved web solution may prove a better choice. Due to time constraints, a few solutions were briefly audited but ultimately discarded in preference of Gitolite. Gitlab should be mentioned as it featured an interface and features very close to those of Github itself, but proved difficult to install and maintain.

Jenkins

A principal element of agile development is continuous integration, the automated concurrent building of new code as it is pushed to central repositories which ensure constant availability of newest binary packages while catching coding errors before pushing them to the public. Jenkins, a fork of Oracle's Hudson, was suggested early and, given no proponents, was implemented. Build jobs were set up for each project, polling their origin repositories for new Git builds to main branches. If a repository has new code, it is downloaded and built. In case of build errors, the project developers are notified by email. To facilitate the deployment phase of each sprint, all projects are rebuilt every Thursday night and pushed to a public FTP server as well as making them publicly available by HTTP.

Git support is not part of Jenkins' core feature set, but is available as a plugin. During development, unhandled exceptions in the plugin code resulted in thousands of of superfluous builds as a failed build due to unexpected circumstances was not marked as failed.

2.4 Decision Making - The Process

The following section will describe the decision making process, set in place to ensure that everyone would be heard on an equal and democratic footing. The decision making process during this semester's multi-project consists of two different steps.

2.4.1 The Weekly Meeting

It was strongly recommended by the semester coordinator, Ulrik Nyman, to hold a weekly meeting for all software students on the bachelor semester of 2013. The meeting's agenda consists of a few points of formalism at the very beginning, in which a secretary and a moderator are chosen by means of voting. Candidates for these roles are entirely self-appointing and a vote is issued to pick one of the candidates.

Though the weekly meeting is established to ensure a higher level of communication between students, as well as ensure that decisions will be taken on a multi-project level scale, not all points are actually discussed at this meeting. Instead, a committee approach is agreed upon, see Section 2.4.3. The purpose of establishing committees is to ensure that relevant discussions to a given topic can be had, but within a smaller audience.

Committees are discussed at the weekly meeting where voting determines which committees are established. A chairman for a committee is self-appointed and a vote determines if there is consent to let the given person be chairman.

The meeting will then proceed and discuss the ideas and suggestions agreed upon within each committee from the previous week and at the multi-project level determine, by voting, which ideas are okay, or if any of the points concluded by one of the committees are subpar and should be reworked.

2.4.2 Rules of Conduct

During the first weekly meeting some general rules of conduct were established, including decisions on how voting should be done. A number of ways to do this were suggested. Ultimately it was decided that every person present at the meeting has an individual vote, and the idea of a group based voting system was therefore discarded. Furthermore in the event that there is a 50/50 split, the vote will have to be reissued. There must be majority 'for' or 'against' a decision. Guidelines for when a decision should be taken at the weekly meeting were established as well. If a decision involved only two or three groups, then it would not be necessary to discuss at the weekly meeting. If, however, the decision impacted everyone, a committee would be established to make these decisions.

During a committee meeting every group has a single vote. It is possible to send as many group members as is deemed necessary to the committee meetings, however, it does not increase the number of total votes a group has.

2.4.3 Committees

A committee ideally consists of a representative from each multi-project group and a chairman agreed upon at the weekly meeting. The chairman is responsible for setting up the meeting, time, place, agenda as well as writing down the details of what is agreed upon during the committee meeting.

The resulting work product of the committee is a document, that potentially answers every question on the agenda, ready to be presented at the next multi-project meeting.

Important Committees

The following section describes an extract of some of the most important committees, that were established during one of the first weekly meetings.

- Wiki: *Ensures that the multi-project wiki page on Redmine is created in a uniform way by establishing guidelines for new articles.*
- Design Guidelines: *Ensures that the User Interface design of the GIRAF application is uniform (e.g. in regards to font, color scheme and various buttons - green for 'yes' and red for 'no').*

- Common Report: *This committee is responsible for the creation of the common-report chapters, which you are reading now, that are at the beginning of every semester report.*
- Pictogram Class: *Because every group requires a common pictogram class, it was decided to create a Pictogram Class committee to determine the functionality that this class needed.*
- GIT: *The GIT committee is responsible for working out a common structure across all repositories to create uniformity and make it easier to continuously integrate.*
- Public Pictogram: *Determines guidelines for how pictograms are handled in the database (e.g. who has access rights to what and why?).*
- Story: *The story committee is responsible for creating a story to follow every sprint. It puts the sprint's tasks into an overall context.*
- CI/Git: *This committee is responsible for coming up with solutions to potential issues that might occur as part of the Continuous Integration step when using GIT.*

Chapter 3

What was developed

This chapter describes the work done for GIRAF in the year 2013 and is rounded off by acknowledging the involved contacts and the semester coordinator.

3.1 Pictograms, Morgana, and Design Guidelines

In this section the notion of a pictogram will be presented followed by how pictograms are currently being used and why they should be digitized. Furthermore the section includes a description of the Morgana library.

The section will be rounded off with the overall design guidelines for the entire GIRAF system.

3.1.1 Pictogram

In the context of this report a pictogram is defined thus: *A pictogram is an image representing a living being, a physical object or some form of action.* Pictograms can contain a text-label, describing the respective images, for clarification. There is currently no standard for the layout or contents of pictograms, due to the specific needs and opinions of the users. User **A** might like to have black and white images with text labels whereas user **B** might want colorful images without text. The images can themselves vary from cartoons to photographic representations. Pictograms are commonly used as means of communication, especially by those requiring assistance with communicating, including but not limited to individuals with autism.

Current Use

During the spring semester of 2013, when this report was written, the use of pictograms is mostly in the form of physical images. The images need to be drawn and/or edited, printed, cut out and then laminated to extend their lifespan. After this process the pictograms are ready for use, generally for one individual, making this repetitive and tedious for the guardians.When the required amount

Figure 3.1: Pictograms in use 2013

of pictograms have been created for an individual, they need to be organized and made accessible with the help of some sort of container. This container can be a folder with a pocket for the pictograms and a velcro-like strip for arranging the pictograms. For communication an individual can choose to form sentences by arranging the pictograms accordingly or use a single image to simply express needs and wants. Another purpose of the pictograms can be to graphically represent instructions for various tasks, in the form of "do **A**, followed by **B** and lastly do **C**" for individuals requiring special assistance.

Digitizing the Pictogram

The GIRAF project focuses on simplifying and digitizing a medium used by individuals with autism and their guardians. This includes digitizing the pictograms, making them available on devices running Android with added functionality. Added functionality includes the option to make the pictograms play a sound, dynamically change the layout of text-labels and editing images. Digitizing the pictogram also makes it possible to share them easily, carry them between devices and make backups of them. Previously, with the same idea in mind, it was attempted to digitize the pictogram. It was considered unsatisfactory (see section below) and therefore the re-implementation in this semester's project.

GIRAF Pictogram Design

The digitized pictogram consists of an image, text-label and a sound. With all elements included, it can be presented as each of the three, two parts combined or all three in union. This viewable container is designed as an extension of

the *Android* view class, making it easy for developers to include and present in their applications. The idea is to have users sharing the same pictograms, with the option to customize their contents without affecting the pictogram itself. The previous GIRAF pictogram design lacked documentation, portability and functionality such as text-labels. Therefore a new design was implemented, which hopefully fits the needs of both future GIRAF developers and GIRAF users.

3.1.2 Morgana

Development on the Morgana library started very late in the project, about a week before final code freeze.

Initially the goal of Morgana was to make it possible for all the GIRAF applications to use both the Wasteland database (see Section 3.2.7) and the local Oasis database seamlessly, however this turned out to be a bigger project than was possible to finish in a week, so the focus was shifted to making it parse and write JavaScript Object Notation (JSON) objects for use in calls to the Wasteland database.

The library implements a Java class for each value object documented in the Wasteland Application Programming Interface (API), each class parses a JSON object and turns it into an object which can be used by GIRAF applications, it is also able to create JSON objects from the stored Java object.

Since the Wasteland database was not done for the final deadline of the code the Morgana library never made it into any of the GIRAF applications of the 2013 semester, but extending the applications developed in the past with Morgana support should be as easy as writing the JSON objects needed for the correct calls to the database.

3.1.3 Design Guidelines

The purpose with the guidelines is to get a consistent look and feel across all of the different applications included in the GIRAF system. The design guidelines have been discussed among all of the project groups, and they are as follows:

- Keep the existing color palette
- Font: Helvetica
- Font size: use common sense. *Android* offers extra small/small/medium/large/huge
- Minimize the use of text, use images instead of text
- Graphical User Interface (GUI) in vector graphics
- Green and red are universal colors for 'accept'/'cancel'
- Applications have animal icons
- Icons are non-customizable
- Every application should be locked in landscape mode

The color palette will be the same as in the 2012 version of GIRAF. With regards to font type and size, Helvetica has been chosen and developers need to keep in mind, that the text has to be readable on the tablet.

The aim is to use more images and less text as the target audience are mostly children, many of which have communication and/or reading difficulties and some have problems imagining objects purely from text.

The GUI will be in vector graphics, because it scales well, which makes it possible to reuse some of the images. Green and red are universal colors for 'accept'/'cancel'. It may sound obvious but other applications have been developed with different colors. Tool-applications should have animal icons.

Lastly everything will be in landscape mode as this eliminates additional implementation for responsive layout, when the tablet is rotated.

3.2 The project of 2013

3.2.1 Admin

This project focuses on the creation of an administration interface for the GIRAF system. The Admin system consists of two parts, one for a desktop computer and one for *Android*. The desktop part will run on a Linux, Apache2, MySql and PHP (LAMP) stack and communicate with the database using the database API provided by the Wasteland (see Section 3.2.7) project. The *Android* part will run on the tablet using the same code base as the desktop part, using a web server application. The main focus of the project is for department managers and guardians to be able to administrate the GIRAF system.

3.2.2 Cars

The aim of the Cars project is to develop an application, which will help children with infantile autism to be more comfortable in using their voice. To ensure that the children learn to use their voice in creating different types of sounds, and not just speak in a monotone way, the application will require the children to create sounds covering different sides of the frequency spectrum.

Cars is a game in which the player has to lead a car through a street into a garage, controlling it with high or low frequency sounds. The car has a matching colored garage at the end, which when entered completes the game successfully. Randomly placed obstacles are used to force the player to avoid them to reach the end.

3.2.3 Croc

The Croc project aims to create an application for creation of pictograms for use in the GIRAF system.

Pictograms can be created in a number of ways:

Camera take a picture with the camera and turn that picture into a pictogram.

Drawing draw a pictogram.

Audio record sounds to attach to pictograms.

3.2.4 Parrot

Parrot is an enhancement of the Parrot project of 2012 and is an application for communication between guardian and child. Its development is based around the currently used physical system Section 3.1.1. The original Parrot application from 2012 also included the administration of categories. It was therefore technically possible for a child using Parrot to access these administration tools, and it is for this reason, that the currently developed version has relocated the administration to a separate application named Category Administration Tool. The version developed during this project will focus on making improvements to the GUI design, adding subcategories (such as breakfast item under the food category) and handle the interaction with pictograms. The primary focus for Parrot remains the same; providing an easier way for children to communicate with guardian in a way that they are familiar with.

Category Administration Tool

Category Administration Tool (CAT) focuses on administrating categories and subcategories. Currently CAT is also responsible for communicating with other applications that need specific pictograms, such as the Tortoise (Section 3.2.5) and Zebra (Section 3.2.8) applications, by providing search/deliver functionality.

3.2.5 Tortoise

The Tortoise application focuses on helping children learn about their own lives and strengthen their social skills. The hope is, that by letting the child interact with pictures and sentences, that are associated with their life, the child can develop an identity. By developing their own identity, the child will learn how to interact with other people by learning what kind of topics to talk about in a conversation with others.

3.2.6 Train

The inspiration for Train comes from an exercise, that one of the guardians practices with the children. The purpose of the game is to create a dialogue between the child and the guardian. The child has to drag pictograms from a train station onto the train wagons and make the train drive. When the train arrives at the next station, the child has to drag the correct pictograms from the train and onto the station. The correct pictograms are decided by the station category.

The category for each station is chosen by the guardians by clicking the category picture frame and browsing CAT (Section 3.2.4) for the picture they want to use. After selecting a category, they select which pictures they want associated with the station.

3.2.7 Wasteland

The purpose of the Wasteland project is to handle all of the data for the GIRAF system. In order to achieve this goal, a database will be implemented on a central server and a local database will be kept on the tablet. The two databases will synchronize data on a regular basis.

3.2.8 Zebra

The aim of the Zebra project is to create a software application aiding guardians in their work. The application should aid the guardian in situations where a child is to perform an ordered sequence of actions. These actions are typically represented by pictograms. Zebra should replace the current paper based version of this system. The guardian should be able to create and manage digital versions of such sequences specific to each child. Upon selecting a sequence for the child to follow, the child should be able to mark actions as done when they are completed to illustrate their progress.

3.3 Acknowledgement

The group of students working with GIRAF during the spring semester of 2013, would like to thank the contacts, who were;

Tove Søby - speech therapist, and contact for three groups.

Mette Als Andreasen - kindergarten teacher at Birken Langholt, and contact for two groups.

Kristine Niss Henriksen - kindergarten teacher at Birken Vodskov, and contact for one group.

Drazenko Banjak - teacher at Egebakken Vodskov, and contact for one group.

Mette Frost - teacher at Egebakken Vodskov, and contact for one group.

In addition the group would like to thank Ulrik Nyman, semester coordinator, for his help, guidance and engagement during the project.

Part II

Analysis

Chapter 4

System Design

As explained in section 3.2.1 on page 30, the Admin system is based on a single code base running on a LAMP stack. Which means we use Apache, MySQL and PHP for the system to operate. However the MySQL part is handled by another group, the WASTELAND group. The Admin system simply interface with it through an API.
Given that the system is written in PHP, the product of this project will be a website. Which has the ability to manage users, pictograms and applications.

This chapter will give an insight into the initial choices made by the Admin group at the very beginning of the project. Choices such as, starting the project from scratch for the 3rd time, how we intend to make PHP code run on a Android device, and how we organize the source code.

4.1 Starting from scratch

As just mentioned, we chose to start the admin project from scratch. Even though there have been 3 admin projects before this one, in the GIRAF project as a whole.
This choice was made based on several problems with the 2 admin projects from the GIRAF project year 2012. The projects was named Oasis and Savannah.

Oasis was a system mainly concerned with constructing a database for the tablet, and every application run on the tablet. The administration part of this project did therefor not get far. It could more or less only display the information about users in the local database that it was constructed with.

The Savannah admin interface got a little further then Oasis. However they too were also concerned with constructing a database, this time for the website version. It was supposed to be able to match that of the Oasis project. However this was not the case when the two projects was done. The database schemes

did not match, which lead to the start of the WASTELAND project.
The WASTELAND project was supposed to have been mainly about the synchronization of the two databases, but ended up having to start the database project from scratch because the available code from the Savannah project was outdated and that of Oasis was unusable as a central database setup.
By the same two reasons could the Admin project not be continued on from the Savannah or Oasis project. If we had chosen to continue on the Savannah and Oasis project, we would first of all have to get the Savannah code back up to date and then make both the Savannah and Oasis project complete, while making both ready for a new database implementation. This would also mean that it would be more difficult then necessary to maintain the Admin system when the project was finished, then it would have to be. Since this would result in two entirely different code bases, one in Java and another in Java Servlet.

For these reasons we decided to start this project from scratch. Developing the system in PHP, since this would make it possible to keep a single code base. And the time it would have taken to fix Savannah and Oasis up to a state where they could both be worked on properly, both with a new DB and with their original features, would approximately take the same time it would to rewrite the system in a single code base.

The next problem that this presents is that, PHP code is not normally something Android systems can compile on their own. How we intend to solve this problem is what the next section is about.

4.2 PHP on Android

PAW! Is the answer.
PAW stands for Pro Active Webfilter and is an application that makes it possible to run a website, written in PHP code on an Android device. This makes it possible, in combination with a web-browser for the user to use the Admin system without being online, which in most cases is the way the target group will be using the GIRAF system.
However we did not have the chance to implement this ourselves, since halfway through the project our tablet went haywire and was not fixed before the end of the project. This also meant that nearly nothing was finished in regard of this feature. Instead we focused on finishing the code base.

4.3 Folder structure and other practical issues

In this section we breifly explain the folder structure as well as why we chose to implement language support and use PHP and JavaScript as the coding languages.

4.3.1 Folder Structure

When designing a system that is supposed to come with a high maintainability it is important to design a useable folder structure. The folder structure of the admin project can be seen in figure 4.1.

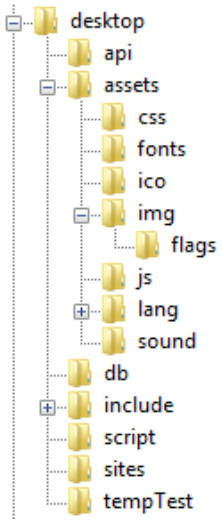


Figure 4.1: The Code Folder Structure

desktop: Is simply the main directory for the dekstop version, since there might come some small variations between the desktop and the Android version of the system.

api: Is intended to contain API's that is not JavaScript or PHP code. And is empty as the end of this project.

assets: Contains every asset in the system. Images, JavaScript, sound, fonts, css and language files.

assets - css: Contains the CSS files for the system.

assets - fonts: Contains special fonts for the system.

assets - ico: Contains the icon files for the system.

assets - img: Contains the static image files for the system. Pictograms and Profile Images are loaded from the database.

assets - img - flags: Contains images of flags, used for changing language.

assets - js: Contains all JavaScript in the system, files are named as they fit to PHP files, as far as this is possible.

assets - lang: Contains all language files for the system. Each webpage and some JavaScripts, have their own folder where these are in. Right now they contain English and Danish language files.

assets - sound: Contains all static sound files for the system. Sounds for Pictograms are loaded from the database.

db: Contains the database files. As of now, all active database functions is stored in the file "new.db.php"

include: Contains any JavaScript or PHP code that is not developed by the Admin project, that is included in the system.

script: Contains PHP scripts that does not display anything, but rather works to execute some function. Such as changing a profile image.

sites: Contains .html and .php files that corresponds to a webpage on the website.

tempTest: A folder for any need of generating files during development.

4.3.2 PHP and JavaScript

As already mentioned in this chapter we chose to use PHP as the backend language. And in order to make some more advanced user interface we decided to use JavaScript, HTML and CSS as frontend language.
We chose to use PHP because most of the Admin group already had experience with the language, and also because we found PAW that would make it possible to execute the PHP code on the tablet.
Also another bonus, that was not considered much when we made the choice, is that PHP is natively able to work with JSON objects which is how we communicate with the database API.

We chose JavaScript, HTML and CSS as frontend because we are writing a website. And there is no real alternative to this.

4.3.3 Language Support

It was in the beginning of the project suggested that the system could be operated in multiple languages, as a start in Danish and English. This was suggested by the semester coordinator. The admin group though, if this was to

implemented at any point, it should be in the start of the system, which meant we would have to implement it into our system.
The way it is implemented is much like it is in Android applications, and can be read in further depth in 9.4 on page 66.

The reader should now have an understanding of the most basic decisions of the Admin project, and be able to navigate in our system.
The next chapter will go through the project progress, followed by the project's problem statement.

Chapter 5

Project Progress

This chapter concerns the work progress of the project and the documentation here of. The work progress was split into weekly progress that was documented with a weekly abstract that is to be conjunction with sprint story. At the end of this chapter should the reader have the knowledge of the development structure that is used and have the overview of the work that is done in this project.

5.1 Weekly Abstracts

Weekly abstracts is a small resume on that has been developed over the week and problems that may have occurred. Usually is the weekly abstract between 5-10 lines that is written the end of each week. Writing weekly abstracts is a easy way to ensure that the development move forward because there will nothing to write if none is done. New comers or a involved third person will quickly manage to get up to date with reading the abstract. At the end of the project it is possible to extract the problems to improve the development method, use as documentation for project and to overall analysis.

5.2 Weekly Progress

In the rest of the section is a description of the weekly progress in conjunction with sprint story. The conclusion of the work progress will be at the end of this chapter.

The first two weeks of project was used to create the groups and determent projects. Internal in the group was the expectation aligned and work conditions was established. Grafical figures and design guidelines was also established.

Sprint stories was first used in week 3 (04/03/2013-08/03/2013).

Week 1 (18/02/2013-22/02/2013) Designs and features for the Admin interface was discuessed. Low-fi board designs was made to give abstract view of our ideas regarding the meeting with our contact person. The first considerations to incorporation of multi language support. The stucture of php, project

and report has been made so it is ready for the first sprint. Setup GIT and Redmine. It was agreed upon to make a web based administration system that is convertible to both desktop and tablet. For further information this choice is readable in chapter 4 on page 35.

Week 2 (25/02/2013-01/03/2013) The focus has been on finishing the preliminary interface design and web application structure. All of the interface design that was drawn on whiteboards is converted to Balsamiq mock-ups for a coherent feel. At the meeting with the contact person there was feedback on the design and features. The work in the committees is started. The feedback, questions and answers regarding how the day care works can be found in appendix C on page 119. The Balsamiq mock-ups and web application structure is found in appendix B on page 103

Week 3 (04/03/2013-08/03/2013) This week work has been on implementing the design and functionality of the login screen, main navigation. We have also edited our mockup designs with the feedback we got from our contact person. Bootstap is included to streamline the design.

The first scrum sprint started with the main story being our, which will last from 04/03/2013-18/03/2013.

"The guardian is in the launcher and starts all the applications one after one, the gaurdian can freely move from application to launcher at any given time. The guardian also enters the administration."

From that story we concluded the demand for our sprint was for a user to log in to the system and view his/her profile page.

Week 4 (11/03/2013-15/03/2013) The first scrum sprint is completed. Following sites is somehow functional implemented in the Admin system:

- Log In/Out
- Own Profile
- View own information
- Edit own information

The site can still not be shown outside of the campus network, and therefore can it not be shown to the contact person. Regarding functionality is the site not connected to the final database there is all the data dummy data. The first parts of the common report is created.

Week 5 (18/03/2013-22/03/2013) Most progress was in implementing the design and edit the mock ups. The committee for the DB-api is started. This committee is important as it will define connection to the database.

The new sprint is: "The guardian is in a application and is working with a picture"

The focus is therefor on implementing profile pictures.

Week 6 (25/03/2013-29/03/2013) Time was short because of Easter. Following features was integrated the Profile Picture Edit and ability to Change QR. Mock ups of Profiles, Create Profile and Department Management was also worked on.

Week 7 (02/04/2013-05/04/2013) This week there only were 4-8 hours of free work time, because of the many lectures and the holiday of this week.

The new sprint is as follows: "The guardian creates a pictogram, and imports the pictogram to an application. The guardian personalizes the application."

Week 8 (08/04/2013-12/04/2013) This week an auto update script that fetches the git-repository for our web-server were made. Uploading the profile picture was reworked to a faster and more data efficient way. But still missing the database to store the data and to give a report when the image fails to upload or succeeds. The QR generators is now fully functional. There is now a printer function to send our QR-images to the browser or OS printing service.

Week 9 (15/04/2013-19/04/2013) The QR system is changed, so that it is secure. Language support now work for all the developed sites.

Week 10 (22/04/2013-26/04/2013) Uploading of profile picture is finished. The work is now on getting the Admin interface down to the tablet. A date with Mette Als (the contact person), to test our system.

New sprint: "The guardian can navigate between applications and use them"

Week 11 (29/04/2013-03/05/2013) This week were used on solving the whole problem with the missing database. A front end was created for the Picto Admin. Features in our system can not be supported in IE9 and below. Therefor is a warning added at the log-in page that informs the user that they are using an old browser if they do not support the "FileReader" system in JavaScript.

Week 12 (06/05/2013-10/05/2013) The week was used to prepare for the usability tests(monday the 13th). Profile Create and Make Relations got finished, with exception of the DB create implementation. Also Picto Admin Create is fully finished. But it does not support categories. In the weekend between the 10th and the 13th was used to ensure that the syestem could work with the newly developed database API. The system was not ready in time for the usability tests.

Week 13 (13/05/2013-17/05/2013) The tests was completed and analyzed. All the bugs and wanted changes got documented and set to be fixed. Most of week was used on implementing the DB API in the system and make fixes. The rapport structure was made and writing is started. The finished functionality can be found in chapter 7 on page 49.

At the end of the project the program was tuned to be fully functional and include notes. Until the report delivery was writing documentation.

5.3 Conclusion of Project Progress

The weekly abstracts was good point to acknowledge the progress and evaluate the work effort. Other the that it not really used. The group work process was more a natural development process the forced by the sprint story. The one thing that did complicated the work the lack of cooperation that should have been much closer.

Chapter 6

Problem Statement

The general purpose for this project has been explained both in this reports introduction and in the GIRAF introduction, see and 3.2.1.

The first GIRAF Admin project was started in 2011, then in 2012 it was split into two projects, called Savanah and Oasis. These two last named projects was focused on constructing Database systems as well as user interfaces to use these. Savanah was focused on an internet version and Oasis on a tablet version. When these two projects was finished, the databases could not communicate and had no chance to do so, because of different database schemes.
Now in 2013 we have split the assignments into constructing an Admin interface and constructing a proper synchronizable database. The database project this year is called WASTELAND [6].

Since the database has been redesigned in order to accommodate the use on several platforms it was a logical choice to also redesign the admin interface, so that it also could be used on several platforms, and keep it in one code base for higher maintainability.
In order to ensure that the Admin project does not have to be discarded again, we want to make sure that this system is properly designed, for maximal user friendliness.
Also in order to fulfill the need of user creation that the GIRAF system as a whole has, we will have to focus on creating all the profile management tools first. This means that if we are unable to construct the whole system that we will be explained in chapter 7 on page 49. We will at least have to construct the user management interface. This will leave the management of pictograms to the tablet projects.

This have lead us to this problem statement:

> *"Currently there are two different administration interfaces for the GIRAF system. This results in a problem with maintainability and user friendliness. How can we make a single user friendly administration interface for the GIRAF system?"*

In order to verify if we have successfully completed the user friendliness part of the problem statement, we have completed a usability test that can be found at chapter 10 on page 75.

Part III

Solution

Chapter 7

System Overview

In this chapter we give an overview of the features that the system contains, their status and why they are necessary.

Below is shown a table containing each feature that we have designed. As can be seen by table 7.1 we have designed a few features that we think is necessary for the online PC system. It can also be seen that we only managed to finish 3 out of 12 features and 4 out of 12 is only in need of DB implementation. That leaves 5 out of 12 features unstarted, where 4 of these are related to the Pics Manager.

Feature Name	Status
My Profile	Done
Profiles	Done
Create Profile	Needing DB
Add Relation	Needing DB
Pics Manager - Make	Done
Pics Manager - Add	Unstarted
Pics Manager - Remove	Unstarted
Pics Manager - Edit	Unstarted
Pics Manager - Delete	Unstarted
Dep. Information	Needing DB
QR Manager	Needing DB
App Manager	Unstarted

Table 7.1: Feature Table - The status explain if they have been finished or not

As it can also be seen from the table we have focused on implementing features that makes it possible to handle Profile related issues. This have been done because it is already possible to handle pictograms from the tablet systems. We did implement one of the Pics Manager features, Make, this is due to it not being system wise directly connect to the remaining 4. The 4 remaining features of the Pics Manager are all based on the same search and select system

and should therefore not prove difficult to implement at a later point. But more on this later, see chapter 12 on page 89.

7.1 My Profile

Status: Done
Resume: This feature displays the logged in users information as well as the users relations and makes it possible to change these information. My Profile also has a sub feature to change the Profile Image of a user, which will be explained in further detail in section 9.3 on page 64.

7.2 Profiles

Status: Done
Resume: This feature is an admin feature, it displays all profiles in a department that have been related to a pedagog in some way. It also makes it possible to navigate to these profiles.

7.3 Create Profile

Status: Needing DB
Resume: This feature is an admin feature, it makes it possible to create new profiles for a department. But is not able to give admin rights to a user. Admin rights can only be given directly from the Database.

7.4 Add Relation

Status: Needing DB
Resume: This feature is an admin feature, it makes it possible to form relations between children and pedagogs, as well as children and parents.

7.5 Pics Manager - Make

Status: Done
Resume: This feature makes it possible to create pictograms.

7.6 Pics Manager - Add

Status: Unstarted
Resume: This feature makes it possible to link a pictogram to one or more children.

7.7 Pics Manager - Remove

Status: Unstarted
Resume: This feature makes it possible to remove a link to a pictogram from one or more children.

7.8 Pics Manager - Edit

Status: Unstarted
Resume: This feature makes it possible to edit a pictogram that have been already created in the database.

7.9 Pics Manager - Delete

Status: Unstarted
Resume: This feature makes it possible to permanently remove a pictogram from the system. 

7.10 Dep. Information

Status: Needing DB
Resume: This feature is an admin feature and makes it possible to edit the information about a department.

7.11 QR Manager

Status: Needing DB
Resume: This feature is an admin feature and makes it possible to change the QR key for a user.

7.12 App Manager

Status: Unstarted
Resume: This feature is for adding or removing the rights to use an application for a child, and is the only feature still in the design step. 

This chapter should now have given a proper overview of the systems features, and it should be clear what the system is intended for and how. The next chapter will explain how we designed the system, followed by a deeper explanation of some of the more advanced features in the system.

Chapter 8

Design

This chapter will describe the design procces of the administraion system.
Seeing as the work from previous years will not continue in this project the opportunity to make a new design filosophy rose. In 2012 the Luancher group made a design guide. This guide was extended in the Design committee 3.1.3 and this project will follow that guide. Basecally it says that one should follow the color theme, which can be found in appendix E, and use vector graphics as much as possible.
Looking at other web administration interfaces like Wordpress insert wordpress ref here the general idea of having a navigation bar at the left side of the screen and the conteent for a given menu on the right side as 8.1 shows. This idea also came from the Android systems settings app, which looks like wordpress admin interface but include a more nuetral way of displaying catagories. The login page, which is the only page that does not follow the genral idea, is inspired directly from Twiiter's twitter ref Bootstrap sign in layout.
Every menu-item went through the same procces of using a whiteboard as a screen and then draw the design in hand. Afterwards the hand-drawm mockups was created as digital mockups in Balsamiq Mockups ref. Then at a meeting showed to our contact person and changed to reflect the issues risen by her. The complete set of final mockups are available in appendix B.

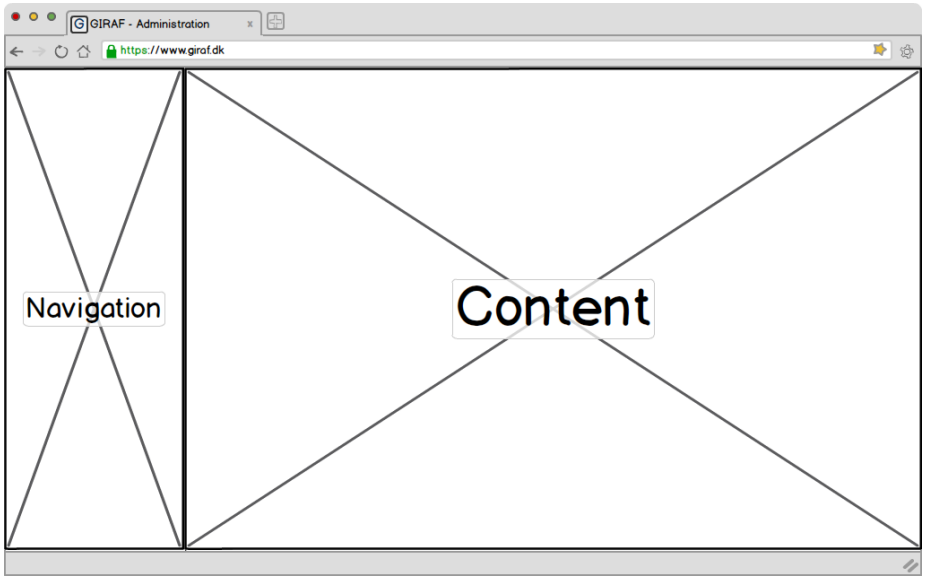


Figure 8.1: General idea of building web-pages.

8.1 Designing individual pages

8.1.1 Login

The Login page is designed to be as minimal as possible. There should be no cluttered information and with a single exeption no other option than to login, the exeption being changing language. The mockup is viewable on figure 8.2.

8.1.2 Navigation

Navigation was one the design items that went through a lot of small changes. The general idea being that it should look an feel natural to an android user. This being that the navigation bar conforms to a principle about accessability so that there is no foldout points or other elements that would be considered diffucult to get to on a touch screen. On the mockups each menu category has a little image next to it, this is changed so that now the first letter is capatalised and uses a bigger font than the rest of the test. If a user with administrator rights login he will be able to see all menu items but if a user with degrated rights logs in he will only see Own Profile, Pics Manager and App Manager further information about department for an example is then accesible through own profile without editting rihgts. A mockup showing the navigation and the profile page is figure 8.3.

Figure 8.2: Login

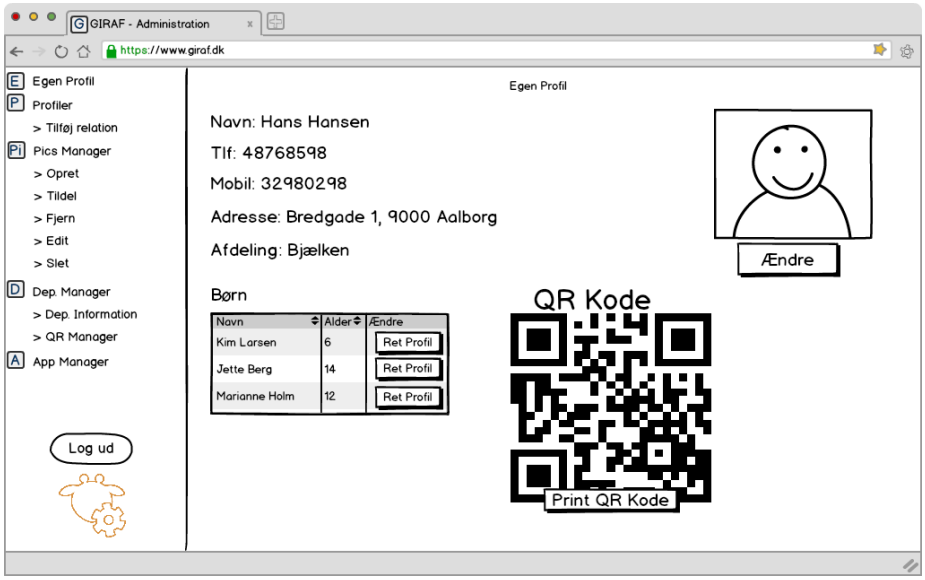


Figure 8.3: Navigation and Profile page

8.1.3 Profile Page

The profile page is the page that is showed when a user logs in. It should have information about the user as well as information about linked profiles such as attached children, parents or pedagouges. A user should also be able to edit his own information as well as attached childrens information. As a result of our usability test 10 the current design used does not include the abillity to change ones QR-code. Instead only the Ddepartment manager has that abillity. A mockup showing the navigation and the profile page before usability testing is figure 8.3.

8.1.4 Profiles

It is based on the idea that having a complete overview of how things are linked sometimes gives a better overview and therefore it is designed such that all links between pedagouges, children and their parents are displayed in an easy to comprehend way. To do this we agreed upon a single table approach which gracefully full-fills the comprehension wanted. This is even more underpinned when color coding is applied to guide the user.

Create Profile

The title basecally says everything. Here the point is that one, if privileged, would be able to create other profiles. A number of input fields as weel as the option to select which type of user one want to create is what this page consits of.

Add relation

Here a privileged user should be able to create relations or links between other profiles. A scenario would be that a child profile has just been created and it should now be linked to it's alreade created parents. This procedure should take place here.

8.1.5 Pics Manager

Pics Manager is based opun having the capabillity to create, add, remove, edit and delete pictograms which all are designed to the same principles of easy accessibility as the rest of the system. Pics Manager should not be accessible on tablets. This is mostly due to the fact that sepperate applications has been developed for it's primary purpose. It should instead open theese apllications and the user should not be bothered by this. The different components design should have the same look and feel as the android application but take advantage of the fact that they are run on a dektop computer and not a tablet.

Create

Originalle named Make this tool should supply the user with the capability to create pictograms in the database.

Add

Add pictograms to users.

Remove

Remove pictograms from users.

Edit

Enables the user to edit pictograms that the user has available.

Delete

If a user own a pictogram he can delete it permanently from the database. 

8.1.6 Department Manager

Enables a privileged user to edit department infomartion and view a shortlist of attached pedagouges.

Department Information

Essentially the same as Department Manager but displays the information as an unprivileged user would see it. An unprivileged user accesees the page through a link on his profile page. 

QR mamanger

Enables a privileged user to change users qr-codes. This is also one of the pages that have gone through a number of deisgn itterations. as viewable in B it originally had three sub items but was refinded to a single page with a much more comprehensive layout. This enables the user to complete a given task much more easaly. A screenshot of the current lauoyt is 8.4

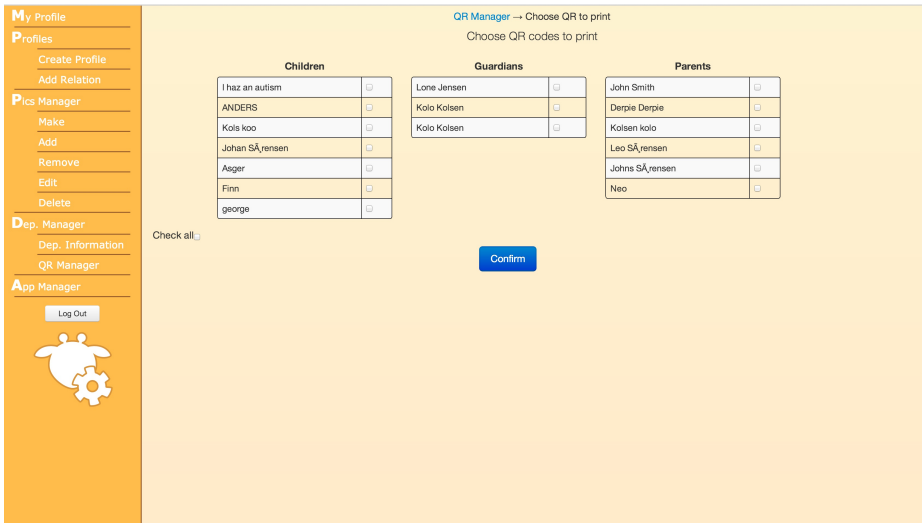


Figure 8.4: Current design of QR manager

Chapter 9

Implementation

This chapter is about implementation...

9.1 Navigation

The navigation system that we have implemented in our system is based on JavaScript and the event `window.onhashchange`, with the addition of Ajax. We use Ajax to fetch the sites without ever navigating away from the index site. This is done because we wanted to minimize the amount of data transfer from the server that the site is made on. However this do present itself with a few challenges. For example we have had to come up with a special solution in order to send the PHP `$_POST` and `$_GET` data around, in this section we will explain how this is implemented and how it should be used in the future.

But first we will elaborate on the alternatives that was available to us.
There is at least two other solutions that could have been implemented with different advantages. First we could have implemented the old HTML method, which means that the user would have to navigate directly to the files name. This would be an easy way to build the website, but would leave it very hard to maintain, since the design of the website would have to be written more than one place, or at least use an include in the top and bottom of each page.
Another alternative would have been the use of a PHP based switch. The method is commonly used in larger website systems, because it makes for easy design and maintainability.

As seen in listing 9.1 it is easy to create the switch method, and maintain it, since all that is needed is to add another case when a new site is made.
What listing 9.1 contains has to be included in the index.php file, and the links, or hyperlinks, will have to set the `$_GET` variable `site` to a fitting name for the site, and it will then include the content of that site.

However this PHP switch solution has one downside to it. It still sends the

```
1   $site = $_GET['site'];
2   switch ($site) {
3     case "ownProfile":
4         include "sites/ownProfile.php";
5         break;
6     case "picsMake":
7         include "sites/picsMake.php";
8         break;
9     case "":
10    default:
11        include "sites/home.php";
12        break;
13  }
```

Figure 9.1: A PHP switch Example

data of the index file from the server to the user each time the user presses a link. This might not be much data, but it becomes so in the long run. We therefore went with a third alternative, which is much like the PHP switch method. We simply took the same idea and wrote it in JavaScript, this however means that we must use Ajax to perform the fetching of new data.

Listing 9.2 contains the main contents of the JavaScript switch that we created. `destination` is the variable that we use for storing the actual hash value[1] and the `postData` is created in a unique way, so that we can send the PHP `$_POST` data on to the site that the JavaScript is Ajax'ing to.

As seen in listing 9.3 we convert the `$_POST` data from PHP into a JavaScript variable that then again is send on to the next PHP site as seen in listing 9.2.

However we also wanted to be able to use the `$_GET` variable from PHP on other sites that we call with Ajax. In order to do this we created a rule as can be seen in listing 9.4, the rule says that instead of using the usual syntax of "?" after the hyperlink, there needs to be a "/" instead. We do this because we think it gives a better look on the link itself. 

Then when the code in listing 9.4 and the switch in 9.2 has been executed we append `info` to `destinationPath` with the normal syntax. Then the Ajax call does the rest.

But this method also has a bad side. It requires a more complex way of calling scripts that is dependent on large amounts of data from the user. For example when the user wish to upload an image for a pictogram.

[1] Hash value is the value followed by the hash symbol # in hyperlinks.

```
switch(destination)
  {
    case "":
    case "#ownProfile":
    case "#otherProfiles":
      destinationPath = "sites/own_profile.php";
    break;

    case "#profiles":
      destinationPath = "sites/profiles.php";
    break;

    case "#profilePicUpload":
      destinationPath = "script/profilePicUpload.php";
    break;

    ...

  }

  ...

  $.ajax({
    type: "POST",
    url: destinationPath,
    data: postData,
    success: function(result) { // result is the content ↩
        that the php file 'ECHO's.
      $("#content").html(result);
    }
  });
```

Figure 9.2: The JavaScript switch

```
  echo "<script>
    var postData = ";
    echo json_encode($_POST);
  echo "</script>";
```

Figure 9.3: The POST transform code

```
1     var hashInfo = location.hash;
2   var hashArray = hashInfo.split("/"); // We use / instead ↩
        of ? in our URL's (for $_GET), they do the exact same,↩
         but gives a different look
3   var destination = hashArray[0];
4   var info = hashArray[1];
5   var destinationPath = "";
```

Figure 9.4: The GET transform code

As seen in listing 9.5 which is a cutout of the `headInclude.php` file, which is always included in our `index.php` file, we include the script directly into the index file instead of using our special Ajax JavaScript function. If we did not do this, the user would have to send the data to the server twice. First in order to send it to the index file, then the user will receive it again, and then call the Ajax function with the same data.

```
1   if(isset($_POST['picsManagerMakeSubmit'])){
2     //Call upload script
3     require "script/picsManagerMakeUpload.php";
4   }
```

Figure 9.5: The handling of big PHP POSTs

And then the upload script must always make sure to use a PHP `header` call to navigate the user to the right site, depending on whether he got an error or not.

If the reader wants to learn more about the navigation script the files used for this script is: `/include/headInclude.php` , `/assets/js/navigation.js` and `index.php` .

9.2 QR Code Generation and Printing

The multi-group last year decided that QR codes would be used as the only login method on tablets, which meant that the GIRAF administration system would need a system for generating and printing new QR codes, in the event a person lost their QR code. In collaboration with WASTELAND it was decided that it would be a security risk to allow people to print out existing QR codes, which meant that our system only has to facilitate the generation of new QR codes and the printing of those new QR codes. The last multi-group also decided

that the QR codes would be based on a 512 character long string. The system generates the new QR codes using the code displayed in Figure 9.6. The QR code generation uses the function microtime in order to get data to hash. The function microtime relies on the system call gettimeofday, which means that this implementation will only work on UNIX based systems and Windows (PHP supplies its own implementation on Windows). The time from microtime is then hashed with the sha512 hashing function which generates a 128 hexadecimal long string. This is then repeated 3 more times in order to get a 512 hexadecimal long string, which is then the newly generated QR code. The newly generated

```
1  function generateNewQr()
2  {
3    $qr = "";
4    for ($i=0; $i < 4; $i++) {
5      $time = microtime();
6      $qr .= hash("sha512",$time);
7      usleep(100); // sleep for 100 microseconds (0.1 ↩
           milliseconds) to get a different time from microtime
8    }
9    return $qr;
10 }
```

Figure 9.6: QR Code Generation

QR code is then inserted into the GIRAF database using the WASTELAND database API. Now the user is prompted to print out the new QR code, as this is the only chance the user has, without generating yet another new QR code. The PHP library phpqrcode[5] is used to generate the QR code itself into an image. In order to support the scalability of the QR code, the image is generated as an SVG, but because the original implementation of phpqrcode does not support SVG output, a modification of phpqrcode is used[2] which adds support for SVG and EPS. Due to a bug with Internet Explorer, phpqrcode was further modified so that the colour of the QR code is statically black. The bug with Internet Explorer was that phpqrcode would truncate the hex code of the colour black to #0 which in Internet Explorer would display as white, while in other browsers it would display as black.
After phpqrcode has generated the SVG, it is then added to a hidden iframe, which only contains the SVG and a separate CSS file which contains the style for printing the QR code. Javascript is then used to open the printing dialogue of the browser with the iframe as its focus, so that only the QR code is printed and not the whole page.

9.3 Profile Picture Change

The changing of a profile picture is normally a very easy task and can be done with a simple uploading of a file and transforming it into the right size with the build in PHP functions.
But in our system it is a bit more complex. Both because we store the images in the Database, so that they can be retrieved on the tablet as well as our PC version. And also because of the before mentioned way we move data around, see section 9.1.

As again, mentioned in one of the former sections, see section 9.1, we have made our system this complex in order to minimize the data transferal from the server. This forces us to make some complex solutions to anything that requires big data transferals.
As for the database, it is generally bad practice to store images in a database, because of the encoding that is needed to do so. But exactly because we want to be able to use the image both on the tablet and on the website, this is actually a good solution. There might have been some alternatives to this, but this is not part of our project. For more information on this the reader should read the WASTELAND project report, that also is a project under the GIRAF project.

As explained in the end of section 9.1, we have made it so that the upload script is called on submit, from the index files, see listing 9.5.
In order to submit the user must go to the profile page, press the ”Edit” button underneath the display of the profile picture, they will then be meet with a modal window[2] where they are asked to chose a file.

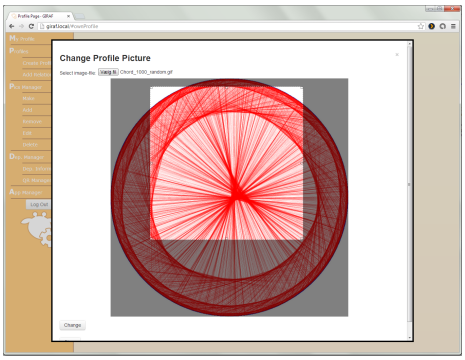


Figure 9.7: Changing a profile picture

When that file is chosen they have to crop the image, as shown on figure 9.7, and finish by pressing the ”Change” button. The cropping here is made with the help of a freeware project called imgAreaSelect [12].
Then when this is done the user will be send back to the profile site, with a

[2] A modal window is a sort of pop-up window on a website.

success or an error message and also be able to see the new profile image on the profile.

But what the user does not see is that on submit they are actually navigated to our profilePicUpload script, that then navigates them back with the error or success message in the URL.
It is this script that handles the encoding and the database query for updating the profile picture.

When the profile picture is uploaded the script first check for any possible upload errors, as well as making sure that the file is actually an image and it was send from our form. This is to prevent security issues. Then when that is handled, the script will load the image into PHP and will transform it by the help of a specialized cropping tool inspired by a PHP project called SimpleImage [8].

```
1    function resizeCordsColor($width,$height,$x1,$y1,$x2,$y2,↩
         $red,$green,$blue) {
2        $new_image = imagecreatetruecolor($width, $height);
3        $white_image = imagecolorallocate($new_image, $red, ↩
             $green, $blue); //Change image color to: White (↩
             RGB format)
4        imagefill($new_image,0,0,$white_image);
5        imagecopyresampled($new_image, $this->image, 0, 0, $x1↩
             , $y1, $width, $height, $x2-$x1, $y2-$y1);
6
7        $this->image = $new_image;
8    }
```

Figure 9.8: The function used for cropping a profile image.

The function seen in listing 9.8 is what we use for reshaping the image into a more manageable size. What it does is first create a new fully white image (L. 2-4), because it is called with the RGB value 255,255,255. Then it proceeds to copy the selected area, that the user selected before onto this white image (L. 5), in a 300 x 300 pixel area, which is the size we decided a profile picture always will be. Finally it stores itself in its own class (L. 7).

As for implementing the database query, we have actually designed it and according to the WASTELAND API [7] it should work. But when we try to update the image it only returns an SQL error in the console. We have asked the WASTELAND group why this is, but they are as clueless as we are. So when the WASTELAND group manages to find the error, this feature will be fully implemented.

This concludes the explanation of the Profile Picture Changer, if the reader wants to learn more about the Profile Picture Changer the files used for this feature is: `/include/SimpleImage.php` , `/assets/js/profileEdit.js` and `script/profilePicUpload.php` .

9.4 Multi-language Support

Android has simple multi-language support, which means that in order for GIRAF Admin to stay consistent with the GIRAF suite of apps, it should also support multiple languages easily. In order to make it as easy for developers to add support for more languages in GIRAF Admin a simple language system was developed. The system consists of separate PHP files for each sub-site and language. Each sub-site has a sub-folder in `assets/lang` in which all of the language files should be. Each language has a separate file with the format `sub-site.language.php`. These files then contain an associative array of strings which contains all of the static strings of the given sub-site. The language of the site is chosen when the user first logins. Each language currently has a flag below the login box on the login site, and so to change language the user just has to click on the flag of the given language. The chosen language is set in the users PHP session, and so all sub-sites has to do is check the session variable and then include the language file for that language.

9.4.1 Adding a new language

Described here is the procedure for adding new languages into the multi-language system

1. For each folder(sub-site) in `assets/lang` a new file should be created with the format `sub-site.language.fileext` where fileext is either `php` or `js` depending on if the language file is for PHP or Javascript.

2. Copy all of the variables from an existing language file into the new language file and translate all of the strings into the new language

3. In `login.php` add the new flag for the new language and give it a link to `login.php?lang=language` where language is the ISO 3166-1 alpha-2 code of the given languages country (although `en` is used to designate English). When the number of flags increase to a number that can no longer be shown visually pleasing, the selection should switch an alternative selecting method more suited for large amounts of languages

4. For each sub-site's PHP file (`sub-site.php`) add the new language to the switch statement and include the language files for the new language for that given sub-site

9.4.2 Adding a new sub-site

Described here is the procedure for adding a new sub-site to the multi-language system

1. On the new sub-site locate all of the static strings, and replace them with references to an associative array like this `$SUB-SITE_STRINGS`

$$'STATIC-STRING-NAME'$$

2. Add code to get the current language from session and a switch case statement to include the language files at the top of the PHP file in the format shown in Figure 9.9.
3. Create a folder for the new sub-site in `assets/lang`
4. Create a new file for each of the currently supported languages in the format `sub-site.language.php`.
5. In each of the new language files create the associative array referenced earlier and add all of the strings with their translation

```
1  session_start();
2  if (isset($_SESSION['lang'])) {$lang = $_SESSION['lang'];} ↩
       else {$lang = 'en';}
3
4  switch ($lang) {
5    case 'en':
6      include($_SERVER['DOCUMENT_ROOT'].'/assets/lang/sub-site↩
           /sub-site.en.php');
7      break;
8    case 'dk':
9      include($_SERVER['DOCUMENT_ROOT'].'/assets/lang/sub-site↩
           /sub-site.dk.php');
10     break;
11   default:
12     include($_SERVER['DOCUMENT_ROOT'].'/assets/lang/sub-site↩
           /sub-site.en.php');
13     break;
14 }
```

Figure 9.9: Including language support on a sub-site

9.5 Pics Manager - Make

The way the Pics Manager - Make feature is implemented is very similar to how the Profile Picture Changer is implemented, see section 9.3.

They both get called the same way and both relay on the HTML Form Element to receive their data. They also give back their error message in the same way.

But what is more interesting is their differences. Looking at figure 9.10 it can been seen how the user interface is designed. The top left part is for entering the general data, and the left side is for the image and the sound file.

We have designed it so that when a user chooses a sound file, or an image file, they can hear, or see, a preview of it before uploading. We also did this so that the design could be reused for the editing feature.
The preview is possible because of a JavaScript function called `FileReader`. Which simply loads the file into the browser, and makes it possible to work with it without uploading the file. But this is only available in some of the newer browsers. Which means we have had to disregard the use of IE9 [4]. In doing so we have created a warning on our login screen that checks if the `FileReader` is available, and if not the user will be meet with a warning, telling them some features will not function for them.

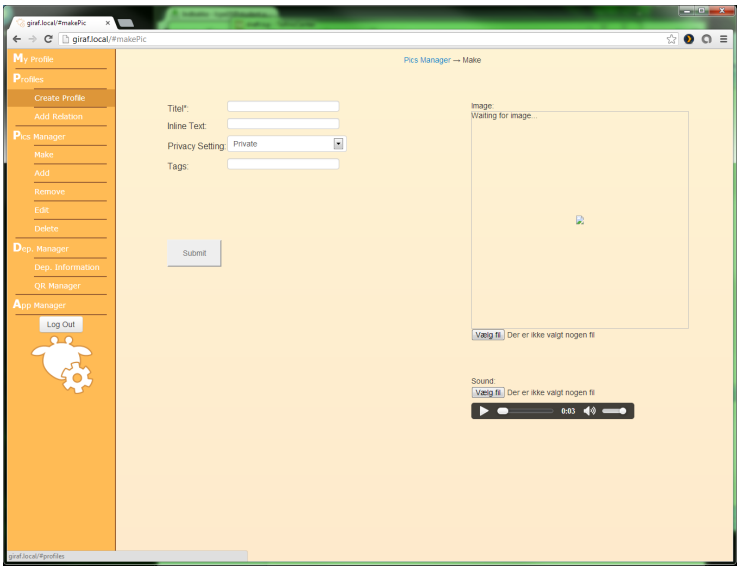


Figure 9.10: The user interface for the Pics Manager - Make

Again when speaking about the backend of this feature, it is very similar to the Profile Picture Changer Feature, where the main difference is that Pics Manager - Make has a specialized sound file recognizer, since this is not natively supported in PHP.
The function used for this can be seen in listing 9.11, the first 2 lines (L. 2-3)

creates two arrays, one for accepted mime-types and the other for accepted file extensions.
Next we do the actual check on file extension and mime-type. (L. 6-23), this is done by simply pulling out the information from the uploaded sound file, and see if it matches any of the variants in the arrays. This do mean that the user can upload a file with a `mp3` mime-type and a `.wav` file extension. - But since this is still an accepted sound file it is no security risk.
As it can be seen we use two variables which is used as booleans, `$fileExtOkay` and `$fileMimeTypeOkay`, these are used when we want to return the information of whether or not this was an acceptable sound file (L. 26-31).

When the sound and image file have been verified they are sent on to the database query, the main part can be seen at listing 9.12. But as also can be seen it is called with a JSON object, which we have another function for. It simply constructs a JSON string from what information the user sent us, we call this function `makeJsonPictogram`.
When we use the function `makeJsonPictogram` the image and the sound file is encrypted to 64 bit, which is needed in order to ensure we do not mess up the actual database query, not just the JSON object we call the WASTELAND system with. This can be done natively in PHP.

When the database query have been completed, the feature will as the Profile Picture Changer, send the user to the creation page with a message of success.

This is the end of all the advanced and complicated solutions we have managed to complete, the next chapter will focus on the results of our usability testing of the somewhat finished system.

```
1   function isAllowedSoundFile($fileName,$fileTmpName){
2     $supportedExtensions = array('3gp','3gpp','flac','mp3','↩
          mid','xmf','mxmf','rtttl','rtx','ota','imy','ogg','↩
          wav');
3     $supportedMimeTypes = array('audio/mpeg','audio/mp3','↩
          audio/mid','audio/wav','audio/x-wav','audio/rtx','↩
          audio/3gpp','audio/ogg','audio/mobile-xmf','audio/↩
          mxf'); //Could not find mime type of .rtttl, .ota ↩
          and .imy
4
5     //Check file extension
6     $ext = pathinfo($fileName, PATHINFO_EXTENSION);
7     if(in_array($ext,$supportedExtensions)){
8       $fileExtOkay = true;
9     }
10    else{
11      $fileExtOkay = false;
12    }
13
14    //Check Mime Type
15    $finfo = finfo_open(FILEINFO_MIME_TYPE); // return mime ↩
          type ala mimetype extension
16
17    if(in_array(finfo_file($finfo, $fileTmpName),↩
          $supportedMimeTypes)){
18      $fileMimeTypeOkay = true;
19    }
20    else{
21      $fileMimeTypeOkay = false;
22    }
23    finfo_close($finfo);
24
25    //return
26    if($fileMimeTypeOkay && $fileExtOkay){
27      return true;
28    }
29    else{
30      return false;
31    }
32  }
```

Figure 9.11: The function used for checking sound files.

```
1  function db_uploadePictogram($jsonPictogram){
2    global $session,$username,$password;
3    $data = '{
4      "action": "create",
5      "auth": {
6        "username": "'.$username.'",
7        "password": "'.$password.'"
8      },
9        "data": {
10         "type":"pictogram",
11         "values":['.$jsonPictogram.']
12       }
13   }';
14
15   $result = db_query($data);
16
17   if ($result['status'] == 'OK')
18   {
19     return $result['data'];
20   }
21   else
22   {
23     return false;
24   }
25 }
```

Figure 9.12: The main part of the DB query for creating Pictograms

9.6 PHP Session Variables Overview

The login system of the GIRAF Admin system uses PHP sessions to check if a user is logged in. The PHP session is also populated with various variables that help the rest of the GIRAF Admin system. An overview of the different variables can be seen in Table 9.1 Many of the variables stored in the session is used to

Session Variable	Description
session_id	Stores the PHP session id of the user
username	Stores the username of the logged in user, Temporary variable until WASTELAND implements API session, should be removed before deploying anywhere!
password	Stores the password of the logged in user, Temporary variable until WASTELAND implements API session, should be removed before deploying anywhere!
userId	Stores the user id of the logged in user, and is used for database calls
profileId	Stores the profile id of the logged in user, and is used to reduce database calls (you can get profileId from userId)
lang	Stores the chosen language (chosen at login) of the logged in user, and is used on all sub-sites to determine what language file should be used
dbsess	Stores the database API session, which is used to make database calls after the initial login, this is here for future use and does not contain anything at the moment, as the database API have not implemented session yet
department	Stores the department id that the logged in user is attached to, and is used to reduce database calls
role	Stores the role of the logged in user, and is used to determine what pages to show
update	Stores the update value of the logged in user, and is used to determine rights
delete	Stores the delete value of the logged in user, and is used to determine rights
isAdmin	Stores whether the logged in user is an admin or not, which determines what menu items should be available

Table 9.1: PHP Session Variables

reduce database calls, which could help the system scale better if that would be necessary. Because API session is not yet implemented by WASTELAND, it has been necessary to store the username and password of the logged in user, and is then used to make database calls instead of the session. When API session is implemented the username and password variables should be removed from PHP session and the authentication method in `new.db.php` should be replaced with API session in all calls except `db_getSession`. At this moment it is not

clear what exactly the update and delete variables signify, and the reader should refer to the WASTELAND report or source code for more details.

Chapter 10

Usability Testing

When we started this project the main reason was to simplify the administration systems to one single interface. So that there is only one code base and only one interface for the user to learn.
The first part should be given by default because we use a specialized system to run PHP code on the tablet. The system is called PAW [9] and is still in its beta phase.
The second part is the tricky part. There is only one interface to use, but it must be easy to learn and use so that the system does not have to be reworked. That is why we have performed a usability test.

10.1 Structure of the test

The test was performed in Cassiopeias usability lab, where we had logged the computer onto our website. The setup of the usability lab is as displayed on figure 10.1. We did not use the second room.
Each test person was lead into the test room, offered a cup of coffee and then we started the prepared briefing, everyone was given the same briefing and we asked them to sign a consent form that they were being tapped for research purposes.
After this we asked them to execute a set of assignments within the system, always the same assignments, designed to lead them around the whole active system. The assignments as well as the introduction given to all participants can be found in appendix D on page 123.
We had the same one person sit in the room to assist the user, both to give them their assignment and to tell them when they had completed it. All assignments was kept very short, as to not disturb the user more then needed.
When the user had completed all the assignments we had the two men that sat in the control room come and help ask questions to the test person about some

of the ways the test person used the system.

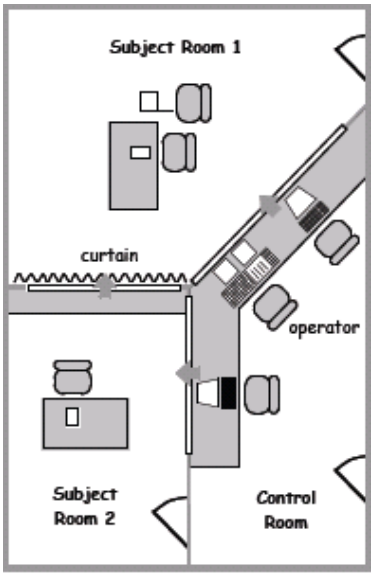


Figure 10.1: Usability Lab

This test was only performed on 4 persons, this might not sound as many. But being that there is only a few persons in Denmark that will use the administration features that we have designed. We had to make an expert test, and use the gathered data as such. We only had one person available that we thought would use the main system that we have designed, but this person who is responsibly for a kinder garden, is not even high enough up in their system to be using most of the admin features. But this is all we had to go on, as we had no way of contacting the persons who is in a high enough position to be the intended user.
So the test was carried out with the one person to be the actual test person, and the 3 others be confirmation test, so that we did not draw conclusions from one test, that could be a result of lacking computer skills or other variables of the same sort.
3 out of the 4 test persons is working with kids with autism everyday, and is educated pedagogs, and is therefore very close to the intended user for the administration system, but is in fact the intended user for some of the sub-functionality of the system. The last test person is an educated computer scientist from Aalborg University, whom has a child with autism, this last test persons use of the system was regarded with respect to that of his education and was mainly used for catching really serious bugs and suggestions for new features.

Lastly, this test was performed on, at the time, older version of the system. Because of some instabilities in the, at the time, current system. The version that was used for the test can be found on github [11].

10.2 Result of the test

Table 10.1 displays the errors or bugs, that we found by the help of the usability test. They are described by first a category, if any, and then by the error itself. They have been sorted by the severity of the error, and we here use 3 categories, critical, serious and cosmetic. Critical depicts an error that either makes a general task impossible or made the user stop all together. Serious depicts an error that made the user stop for some time, or seemed to be unnecessarily distracting. Cosmetic depicts an error of low caliber, that even the users informed us was minor issues that did not affect the flow much more then a few seconds.

Some of these errors was nearly immediately rectified, or had already been solved in the system that was up to date. These can be seen in table 10.1 as "Done". If the done, is marked with parentheses, it is because the problem have been partly solved.

As mentioned before there is errors marked critical, as can be seen by the first 5 entries in table 10.1. These are errors that must be fixed for the system to be able to use the implemented features, as listed in chapter 7 on page 49.
We will here take a moment to explain the critical problems, and in short how they can be rectified.

It is not possible to create new categories of Pictos
This error arose because we had not thought the creation of categories through. This is one feature that we had overlooked, even after having had a meeting with our contact person, and showing all the features we had intended to create.
The way to solve this is to create a minor tool that makes it possible to create and assign categories, that can be created by a simple entry of a text string. For navigation purposes this tool should be put under the Pics Manager category in the navigation menu.

Profile - Restrict QR editing to department manager After having our first test, we was made aware how highly they think of the QR code that they are given for the system. It is supposed to be as important as your house key, and should therefor not be something that can easily be reprinted.
Therefor to accommodate this request the QR changer should not be available from the Own Profile site. Instead the procedure will be to contact the admin

Description	Severity	Done
It is not possible to create new categories of Pictos	Critical	
Profile - Restrict QR editing to department manager	Critical	
DB Problem - Not possible for all pedagogs to fix pictos for all department children	Critical	
Navigation - The site ”Profiles” should link to each profile	Critical	
Missing - Unable to remove relations	Critical	
Profile Pic - Accept button is hard to find	Serious	
Profile Pic - Word ”change” is misleading	Serious	
Navigation - ”Add” and ”Make” under Pics Manager is confusing	Serious	
Navigation - Language Support on navigation did not change to danish	Serious	Done
Missing - No Danish language support on the site ”Profiles”	Serious	
Profile - Department should be a link, not editable	Serious	(Done)
Profile - Links from Own Profile relations is missing	Serious	
Profile Pic - GIRAF Logo as Placeholder is misleading	Cosmetic	Done
Profile Pic - Word ”Edit” is not informative	Cosmetic	
DB Problem - Own Profile takes too long to load	Cosmetic	
Navigation - ”Add Relation” is before ”Create Profile”	Cosmetic	Done
Standardize button names	Cosmetic	
Logout - Can navigate in system without session	Cosmetic	

Table 10.1: Bugs/Errors Found Under Usability Testing

of the system and have him generate a new key with the QR Manager tool and then send this out.

DB Problem - Not possible for all pedagogs to fix pictos for all department children
This problem most likely erupted from a misunderstanding between the DB group, WASTELAND, and ourselves. The problem is that in order for a pedagog to edit anything about a child in its department they need to have a relation between them. But this is not how the user intends to use the system. The relation should only be thought of as a responsibility link, and only adds the feature to change this child's profile data.
There is no easy way to rectify this problem from the admin projects side. This has to be fixed inside the database, because of its security mechanisms. For a solution to this we would like the reader to check out the database project, WASTELAND [6].

Navigation - The site "Profiles" should link to each profile
This is simply a matter of a feature that have not yet been implemented, but it is shown here as an error, because it is necessary for the system to be operated correctly.
It can be fixed by adding normal HTML anchor links to each name in the tables.

Missing - Unable to remove relations
This is again a matter of a feature that have not yet been implemented.
We have thought this feature to be usable only for the department manager and the admin of the system. And should be used directly from each users profile. This means that the admin or department manager, navigates to the users profile, and find the person he or she is related to and clicks the button to remove this relation.

The rest of the errors is not of a nearly as urgent caliber, and should be easy enough to solve, we will therefore not discuss these further in this report.

10.3 New Features

Instead we turn our attention to the possible extra features found during the test, both inspired by the way the users used the system, and their commentary afterward.

As seen from table 10.2 there is a few features that could improve the system. We will here go through some of the features that might be a bit hard to understand without having been on the developing team.

Description	Done
Make the modal window customizable	
Profile - Press enter to save change	
Pics Manager Make - Add recording feature	
Input forms - Automatic first letter uppercase for name and address	
Navigation - Add link to ”add relations” and ”create profile” in Profiles	
Pics Manager Make - Directly create picto for child from profile page	
Pics Manager Make - Auto add ”inline-text” to picto preview	

Table 10.2: Features Found Under Usability Testing

Make the modal window customizable
We have during this project designed our own modal window, because of some issues with the bootstrap library. It seemed that the user sometimes found it confusing with the many cancel options that is offered for closing the modal window, even though it follows the way bootstrap displays their modal windows. But also in the long run it could be useful to have the ability to customize the modal window with more than a title and a text.

Input forms - Automatic first letter uppercase for name and address
While watching the users work in the system, we multiple times witnessed them deleting a full text string because they entered the first letter of an address or a name in lowercase. Therefor to save the user time, we thought that making each word in the address and name field automatically convert to uppercase. Since that is the custom with all names and addresses.

Navigation - Add link to ”add relations” and ”create profile” in Profiles
Some of the users in our test meant that the most meaningful way to handle profiles, would be directly from the menu point ”Profiles”, and did not bother to look just below this menu point, where these options lay. This behavior was seen in 3 out of 4 cases, and should therefor be considered.
It is a simple addition and does not hinder the system in any way. So if it even only helps a few people in the beginning of the learning of the system, this is feature worth having.

Pics Manager Make - Directly create picto for child from profile page
While watching the users work in the system, and also after the, when they were interviewed. We came to understand that some of the pedagogs, in our testcase 2 out of 3. Thought of the child as the central element in the system and therefor wanted to create everything out from the child.
This meant that when they had to add parents or pictograms to the child, they went to try and find the child's profile. Therefor in order to accommodate this behavior we should implement a feature that makes it possible to automatically relate a child and a Guardian as well as a child and a pictogram, when the action is started from that child's profile.

Pics Manager Make - Auto add ”inline-text” to picto preview
Some of the users did not quite understand what the inline-text field in the Pics Manager Make function meant, before we asked them. Which after they immediately understood what it was for. During the test, we even saw one of our test persons fill the field, and then delete the entry again.
If this was a result of the assignment regarding the use of pictograms did not specify that the inline-text field needed to be filled, or something else, we are not certain.
But a way to avoid this problem would be to simply add a JavaScript feature that automatically writes on top of the temporary display of the new pictogram.

The reader should now be as well informed of the structure and shortcomings of the GIRAF Admin system, as the developer group is. But in case the reader wants more information about the system, the reader should try to contact Ulrik Nyman, Associate Professor at Aalborg University and ask about the GIRAF Admin group of the summer semester, year 2013.
In the next chapter we draw conclusions about the system as a whole, and after that we make an evaluation of the process and the multi-project as a whole.

Part IV

Perspective

Chapter 11

Conclusion

This projects focus was on making an administrative interface for the GIRAF system on both a desktop and an android platform. Although the developed system was never deployed on an android platform the system is running on a server and is accessible through common web-browsers.
As it was already shown in chapter 7 on page 49 we did not complete all the features of the system. This was due both to trouble with the WASTELAND API as well as unforeseen trouble with different aspects of the already created system.
However we did not set out to make a 100% complete system, but rather a system that, which parts that was complete, would not have to be worked on again. But also with mind to that the most important features for the Admin system to complete would be user management, since the pictogram features would be implemented on the tablet as well.

But we regret to inform that we did not manage to do this either. As can be seen of chapter 10 on page 75, there is errors and bugs in our system, and even missing parts.

By looking at table 11.1, it can be seen which features that still need implementation in order for the system to be useable. The table displays all the errors and bugs we found under the usability test described in chapter 10 on page 75, with exception to the errors that we already fixed.
In order for the system to be usable the critical errors must be rectified. In order for it to work desirably, both critical and serious errors must be rectified. 4 out of 5, of the critical errors can be fixed by a couple of days hard work, but the last error regarding the DB Problem, can not be fixed within the Admin system. It did not get fixed this semester because of the very late discovery of this error. We did go to the WASTELAND project group when it was discovered, but with only two weeks left, reserved for report writing, neither did they or we have the time to find and fix the problem.

Description	Severity
It is not possible to create new categories of Pictos	Critical
Profile - Restrict QR editing to department manager	Critical
DB Problem - Not possible for all pedagogs to fix pictos for all department children	Critical
Navigation - The site "Profiles" should link to each profile	Critical
Missing - Unable to remove relations	Critical
Profile Pic - Accept button is hard to find	Serious
Profile Pic - Word "change" is misleading	Serious
Navigation - "Add" and "Make" under Pics Manager is confusing	Serious
Missing - No Danish language support on the site "Profiles"	Serious
Profile - Department should be a link, not editable	Serious
Profile - Links from Own Profile relations is missing	Serious
Profile Pic - Word "Edit" is not informative	Cosmetic
DB Problem - Own Profile takes too long to load	Cosmetic
Standardize button names	Cosmetic
Logout - Can navigate in system without session	Cosmetic

Table 11.1: Bugs/Errors Found Under Usability Testing without the fixed bugs

But if we look away from what is lacking and instead turn our attention to the other aspects of our problem statement. Displayed here below to refresh memory.

> *"Currently there are two different administration interfaces for the GIRAF system. This results in a problem with maintainability and user friendliness. How can we make a single user friendly administration interface for the GIRAF system?"*

The system is actually quite user friendly. If the problems in table 11.1 was to be fixed the system should be free of slowing effects in the matter of daily use. Multiple of our test persons did actually praise the system on the simple structure during the interview after the test.

The system only have one codebase and the maintainability for the new system should therefore be quite a lot higher than that of the former two systems. Also we have made an effort to both comment on our code, and also in this report describe some of the more complex solutions, especially in chapter 7 on page 49. Also this report comes with an install guide in the appendix, see chapter A on page 99, so that this project should be as pain free to continue on, as possible.

With this said, we did not complete what we set out to do. But we did everything we could to ensure that this project is possible to continue on for

whomever is to take over after us. A goal we thought higher then simply completing the system.
In the next sections we will evaluate on the multi-project itself and after that make a note of what could be done in the future to improve on this system as well as the GIRAF system as a whole.

Chapter 12

Future Work

TO BE MADE...

Chapter 13

Project Proposals

To keep a conteius progress of GIRAF is there need for futher development oppeunities there have written down projects we have fund though our time working with the GIRAF project.

Event Calender After meeting with Mette Als at Birken was it clear that there is need for digital adminstration orgenice tool the day basis orginas the guardiens and children main activities. The Schema should also be convertet to a simplifieded version for parants.

Daily Sekvens There is need for a visiule daily sekvens for children that is interaktive.

Child Mode Child mode is needed to ensure childeren is able to play with the tablet without supervion of gauidians. Child mode means that the tablet should be total rescrictet to a special app or special apps in launch mode.

Data and Analysis Create application that gathers information from apps to analyse the childs development.

Scheduled Sekvens Expand sekvens to be triggered by events like time on day or a activity.

Co-op Games Develop a game that that has in focus for children to inter act with other children to accomplish a task.

Development Game Tool Create a tool for easy game development for children with autism.

Language Stimulatioen Create a apps that stimulate children's ability to spell and enunciate words.

Chapter 14

References

TO BE MADE...

Bibliography

[1] Wikipedia, 2013. Looked up: 26-05-2013.

[2] Alexandre Assouad. *PHP QR Code - SVG/EPS Update*. URL: https://github.com/t0k4rt/phpqrcode, 2012. Looked up: 20-05-2013.

[3] American Accreditation HealthCare Commision. Autism, May 2012. http://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0002494/.

[4] Alexis Deveria. Can i use... Website, May 2013. http://caniuse.com/.

[5] Dominik Dzienia. *PHP QR Code*. URL: http://phpqrcode.sourceforge.net/, 2010. Looked up: 20-05-2013.

[6] The WASTELAND GIRAF group. Wasteland api. Website, May 2013. https://github.com/Ezphares/giraf_database/.

[7] The WASTELAND GIRAF group. Wasteland api. Website, May 2013. http://htmlpreview.github.io/?https://github.com/Ezphares/giraf_database/blob/master/source/api/documentation.html.

[8] Simon Jarvis. Simpleimage php libary. Website, November 2006. http://www.white-hat-web-design.co.uk/blog/resizing-images-with-php/.

[9] Jochen. Paw by fun2code. Website, May 2013. http://paw-android.fun2code.de/.

[10] Pyramid Droup Management Services. Picture your student learning. Website, Visited 21.03.2013. http://pecs.com.

[11] SW601F13. Giraf - admin: The test version. Website, May 2013. https://github.com/Zucka/girafAdmin/tree/46e19108a6ca0413e25426212552ac6da99b8962.

[12] Michael Wojciechowski. jquery img libary. Website, May 2013. http://odyniec.net/projects/imgareaselect/.

Part V

Appendix

Appendix A

Installation Guide and Configuration

This section will describe how to install and configure the GIRAF Admin system.

A.1 Requirements

- A suitable operating system (Ubuntu, Windows and OSX was used during development)
- Apache (Not tested on nginx)
- PHP5 with the socket module enabled
- A suitable browser (Chrome,Firefox,Safari,IE10+ is recommended)
- GIT (The project is hosted on GIT)
- Python if auto update is used

A.2 Installation

- Install Apache, PHP5, GIT and Python
- Pull the most current version from https://github.com/Zucka/girafAdmin (although forking the project is advised)
- Copy the contents of source/desktop into the www root of Apache (usually /var/www/ on UNIX systems)

A.3 Configuration

- Enable the socket module of PHP5 if not already enabled
- Enable the header module of Apache if not already enabled
- Modify the Apache configuration file (httpd.conf) to auto-inject a meta tag into every header (used to force Internet Explorer into standards mode). The configuration fragment to be added is listed in Figure A.1.

```
1 <IfModule headers_module>
2    Header set X-UA-Compatible: IE=Edge
3 </IfModule>
```

Figure A.1: The configuration fragment to add to httpd.conf

- Change the variables address and port in source/desktop/db/new.db.php to reflect the IP address and port of the server that hosts the database API

A.4 Auto Update

A python script was made to auto update the server from a GIT. It can be used to auto pull new changes from a GIT and then move the contents of a folder ínto another folder. To get it to work you need either Python 2.X or 3.

- Copy autoupdate.py from source/scripts/autoupdate.py to a seperate folder, to be used as an intermediate folder
- Change all references to `/home/neo/autoupdate` to point to the seperate folder created for auto update
- Change all references to `https://github.com/Zucka/girafAdmin.git` to point to the new GIT for the project (the GIT installed on the system needs to have read rights for the GIT repository)
- Give autoupdate.py execute rights (sudo chmod +x autoupdate.py)
- Make sure that the www folder and all subfolders have the same owner
- Start autoupdate.py as the user that owns the www folder (example: `sudo -u www-data python autoupdate.py`)
- Alternatively start autoupdate.py in a screen (the UNIX program) instance so that auto update keeps running after the ssh connection has been terminated

- The www folder will now be updated every 60 seconds with the newest changes from the specified GIT

Appendix B

Graphical Mockups of The GIRAF Admin System

Figure B.1: Login

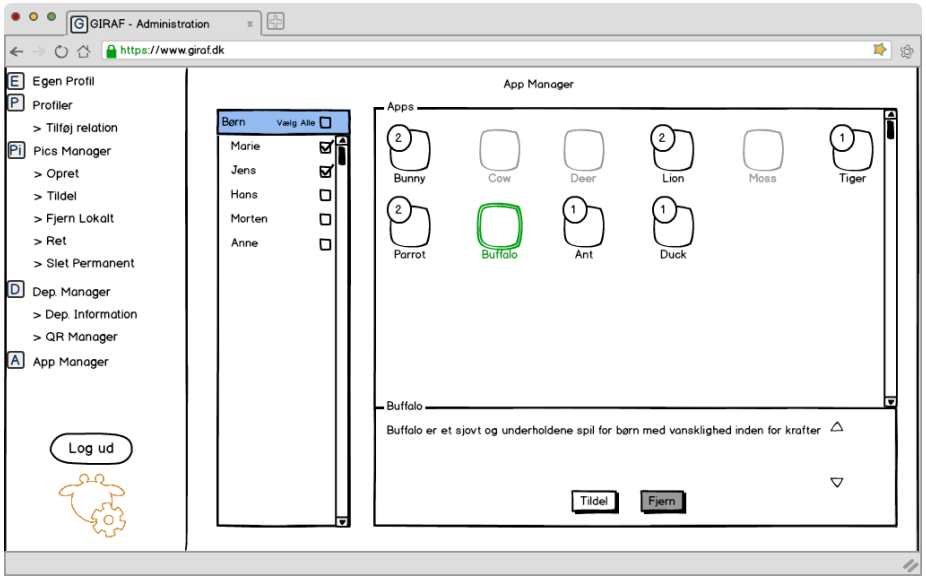


Figure B.2: App Manager

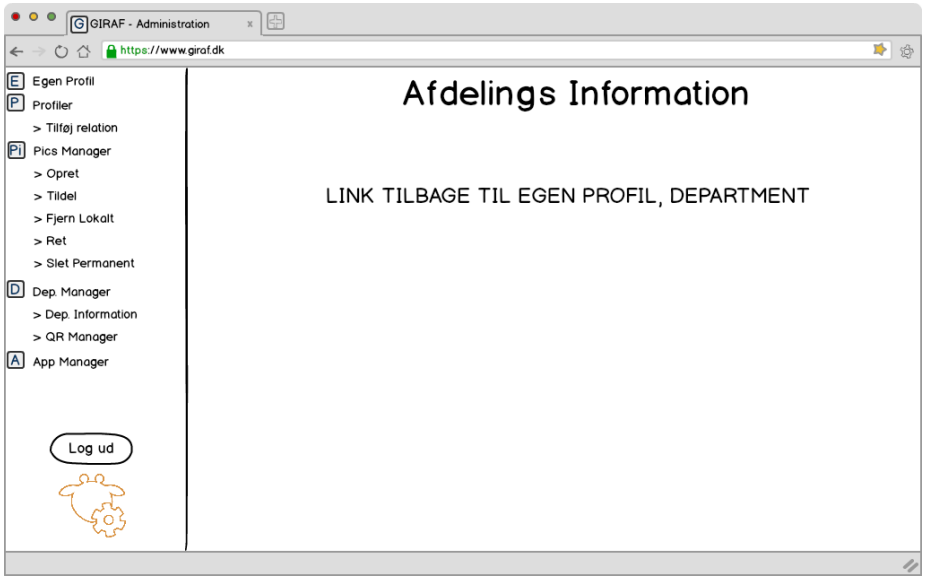


Figure B.3: Department Information

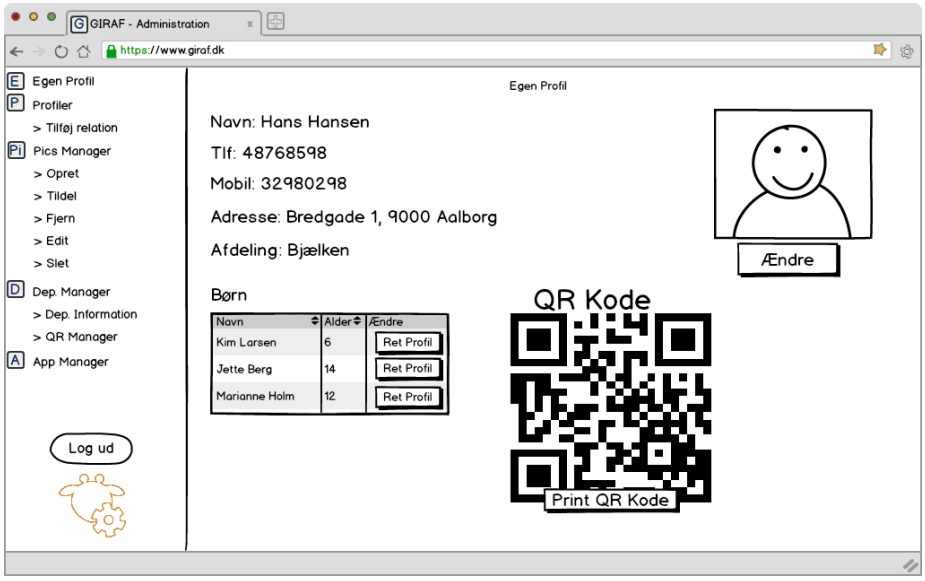


Figure B.4: Own Profile

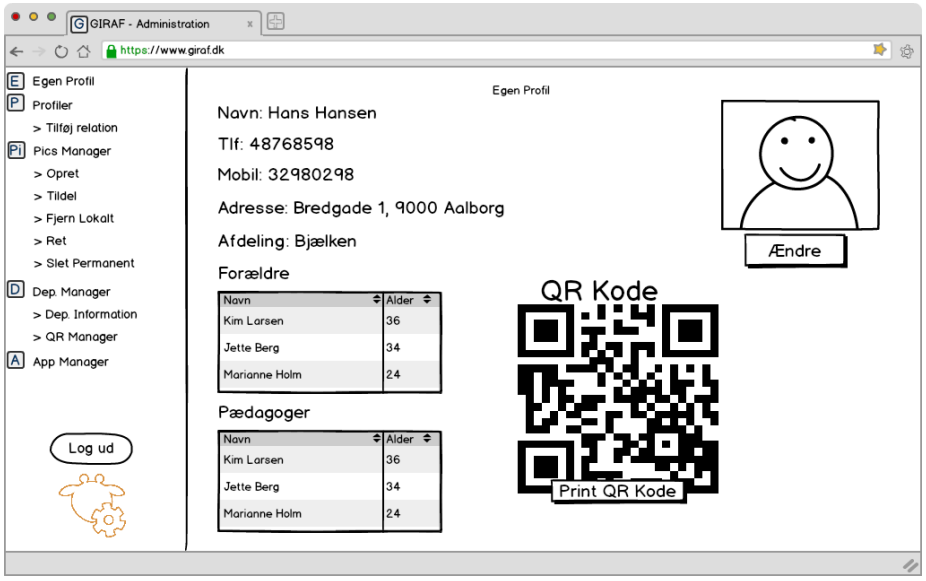


Figure B.5: Own Profile Child

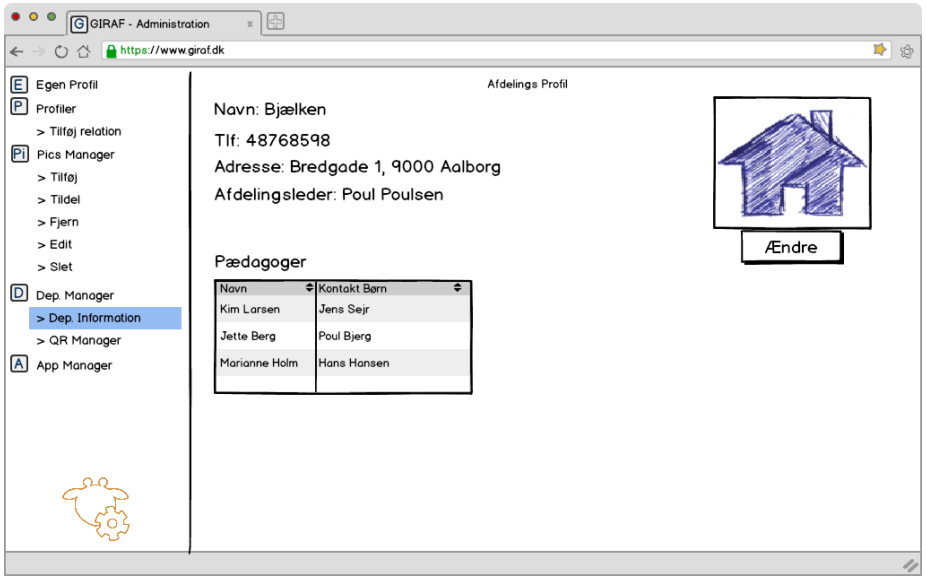


Figure B.6: Own Profile Deparment

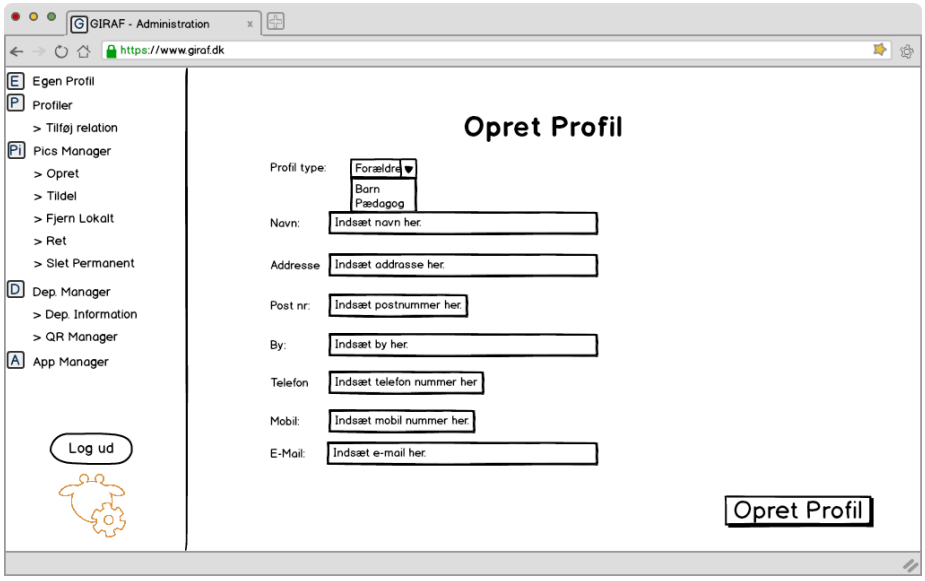


Figure B.7: Own Profile Guardian

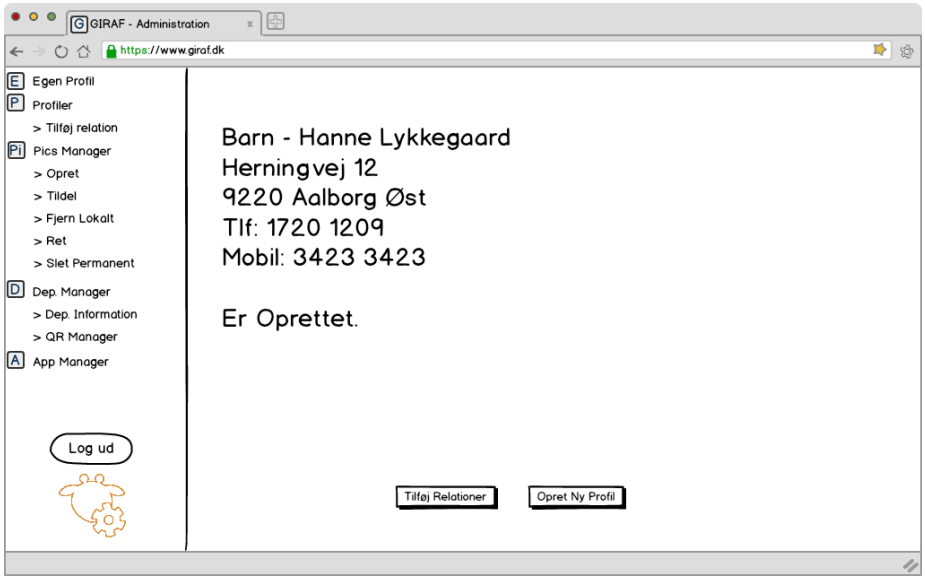


Figure B.8: Create Profile Accept

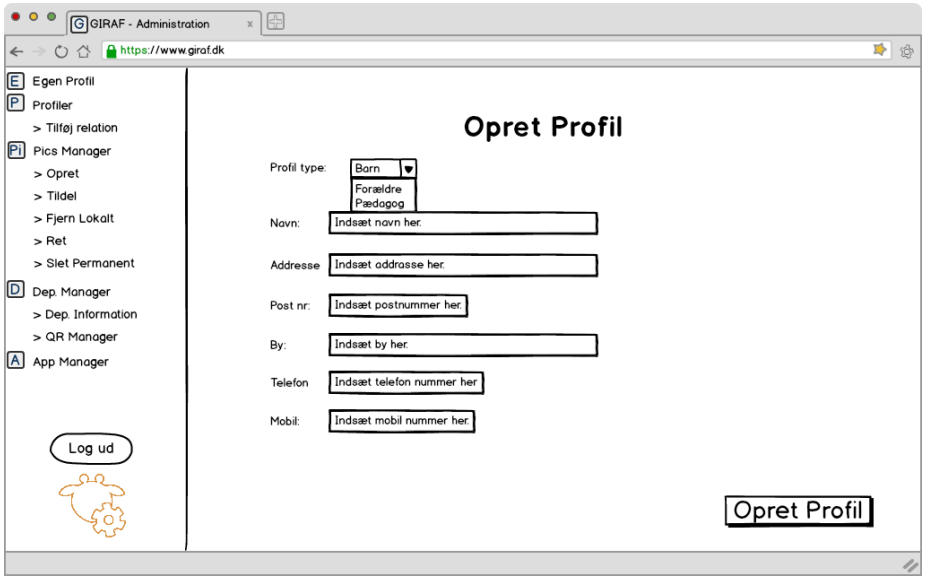


Figure B.9: Create Profile Child

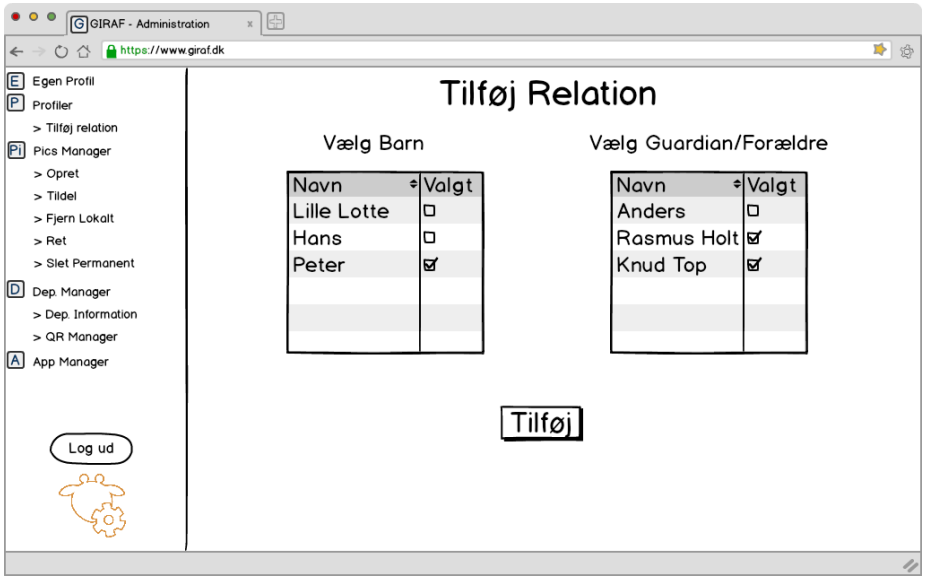


Figure B.10: Create Relation

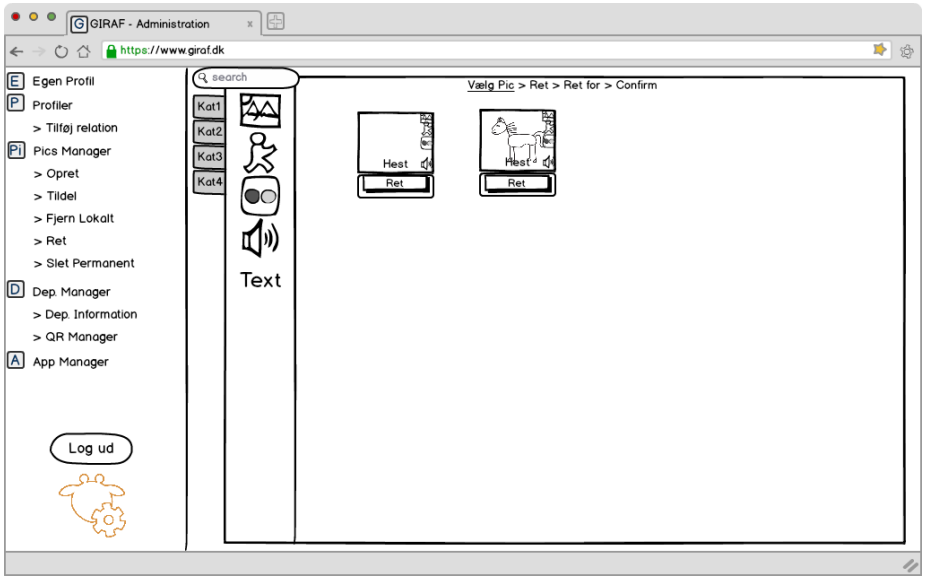


Figure B.11: Pics Manager Edit

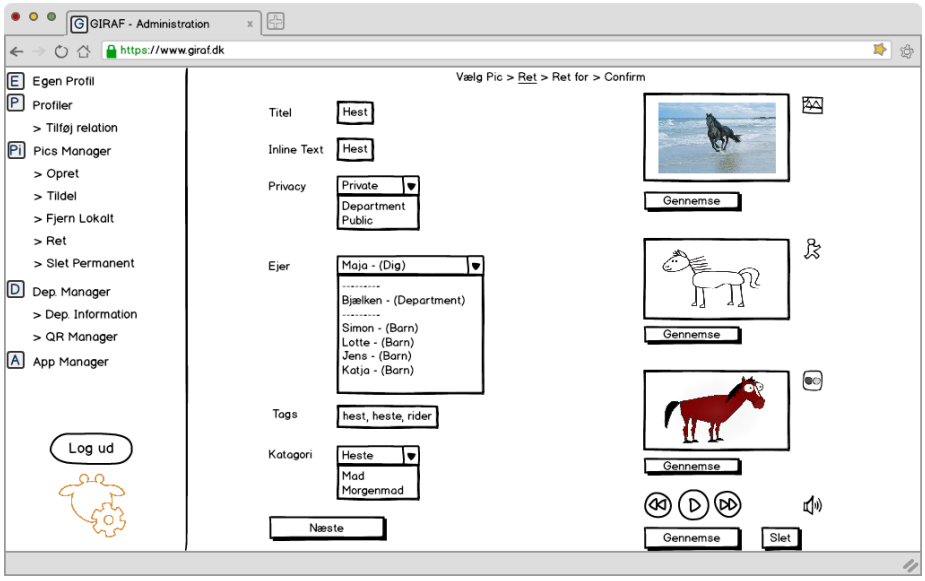


Figure B.12: Pics Manager Edit 2

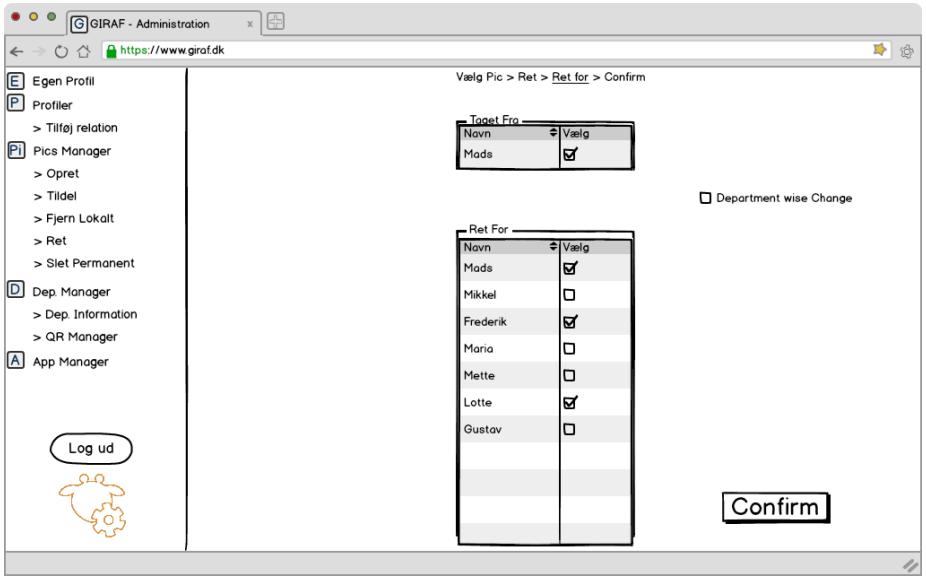


Figure B.13: Pics Manager Edit 3

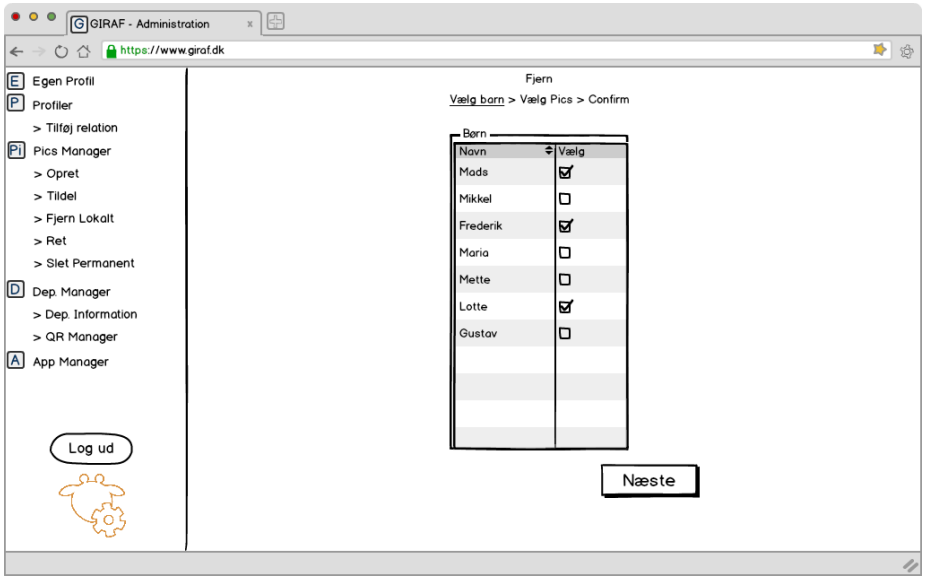


Figure B.14: Pics Manager Remove

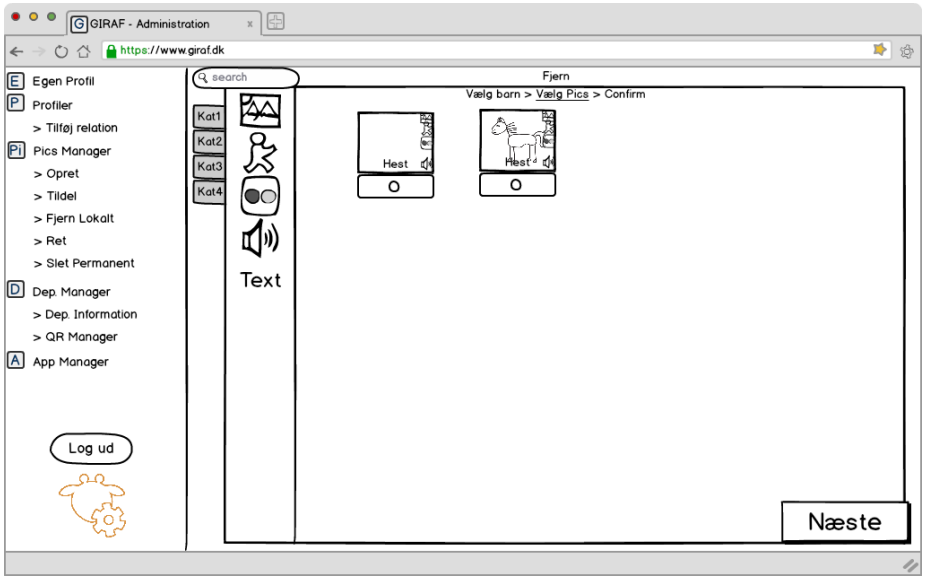


Figure B.15: Pics Manager Remove 2

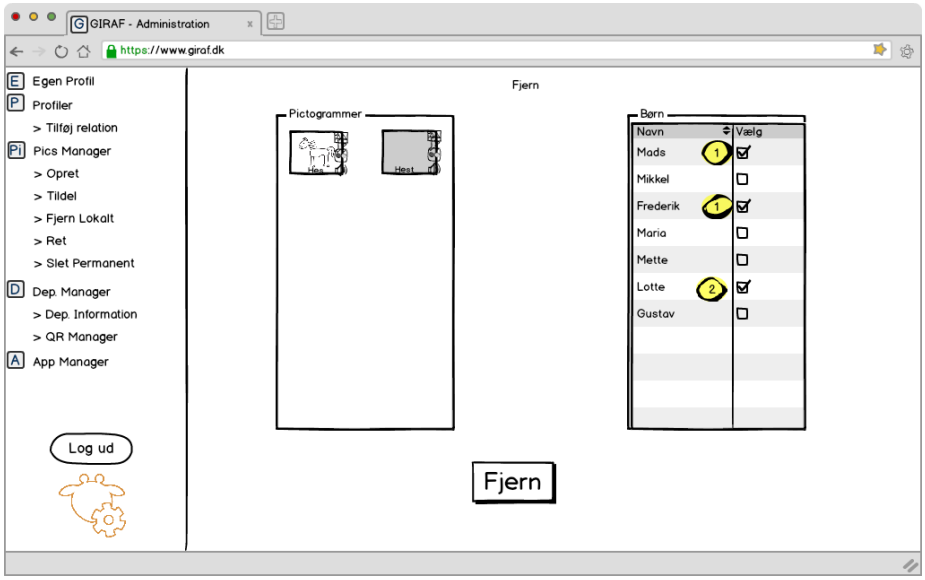


Figure B.16: Pics Manager Remove 3

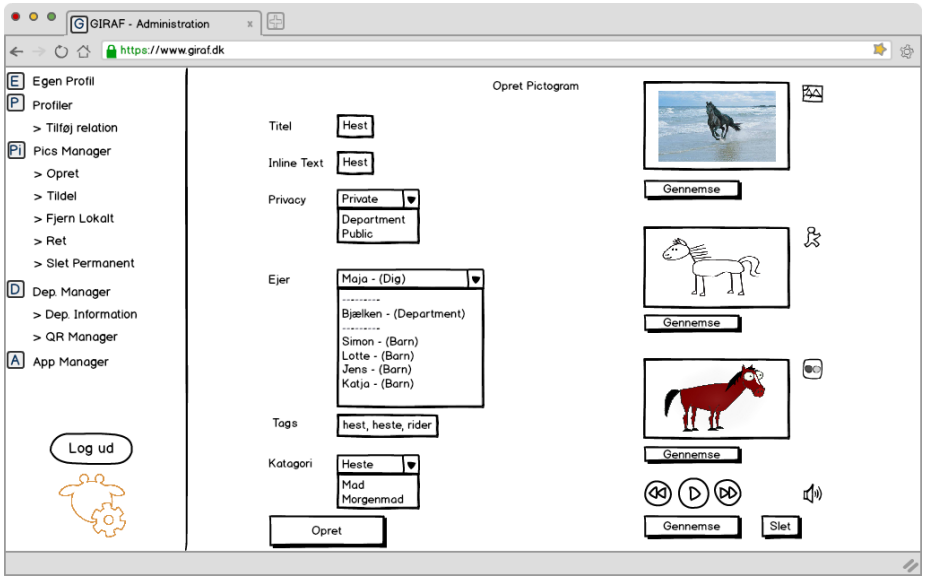


Figure B.17: Pics Manager Create

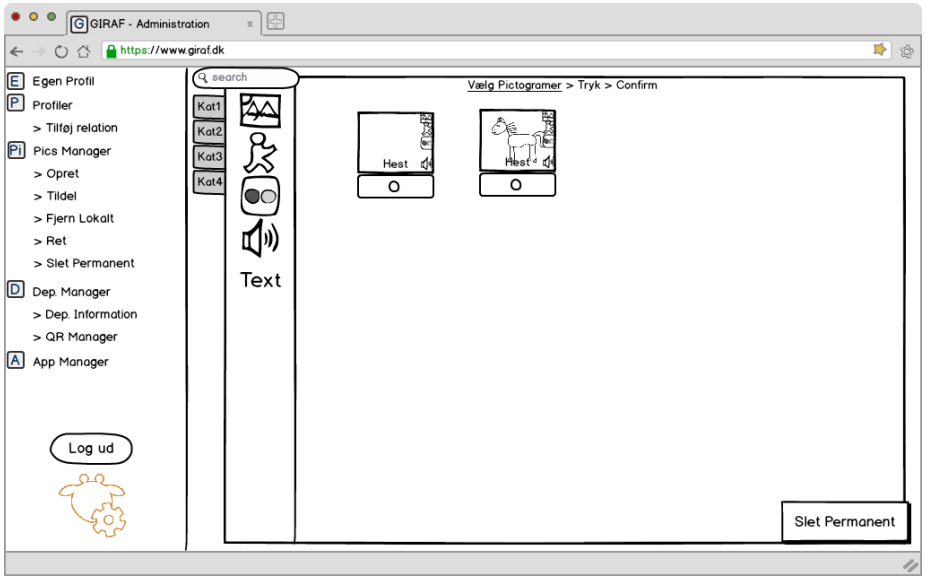


Figure B.18: Pics Manager Delete

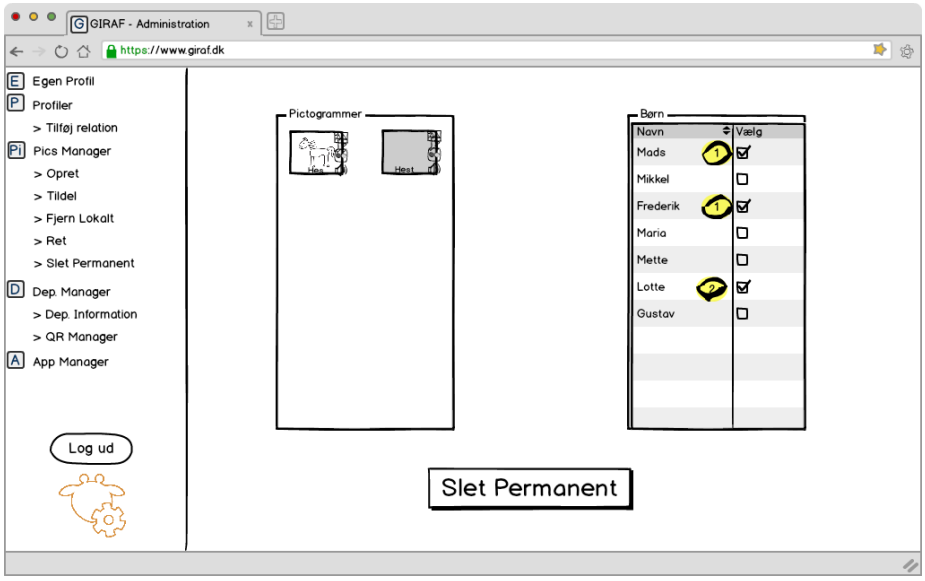


Figure B.19: Pics Manager Delete 2

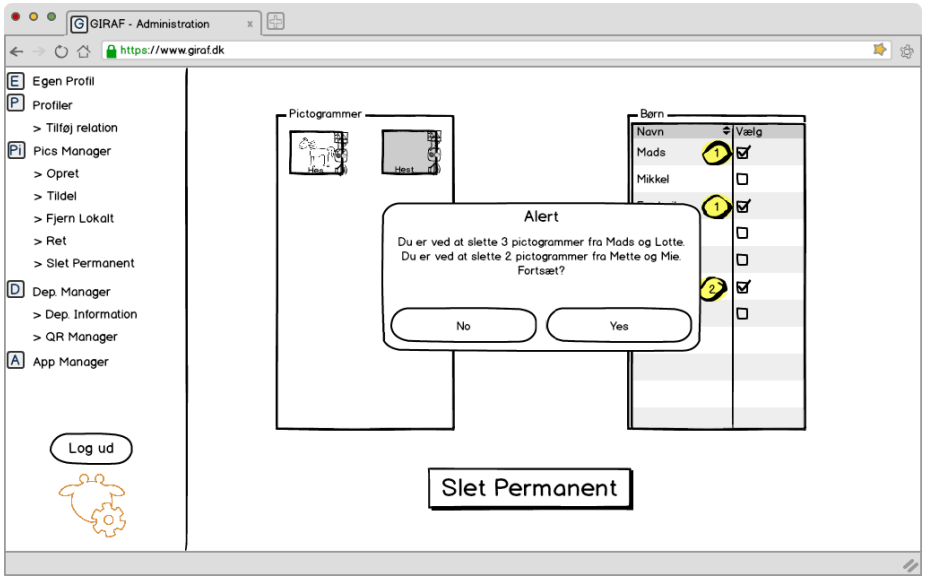


Figure B.20: Pics Manager Delete 3

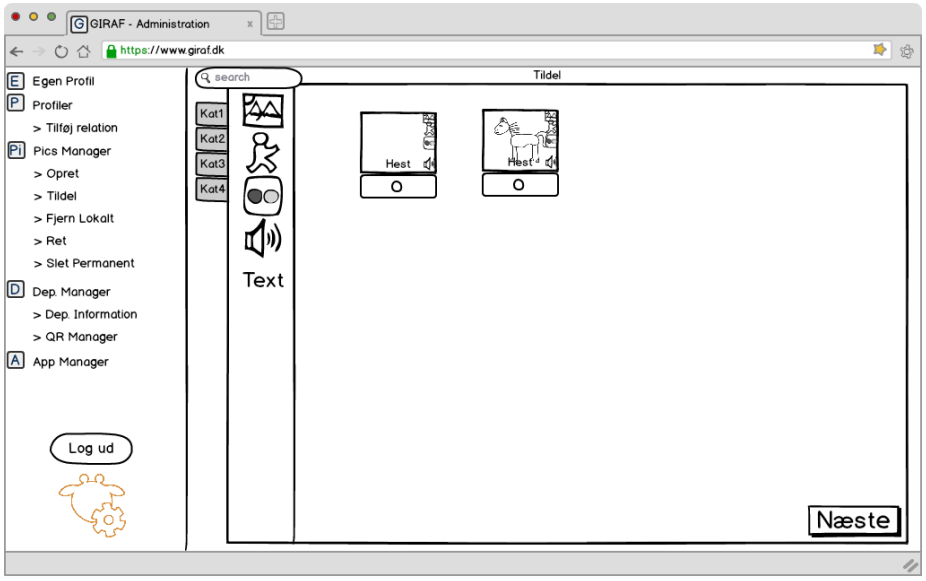


Figure B.21: Pics Manager Assign

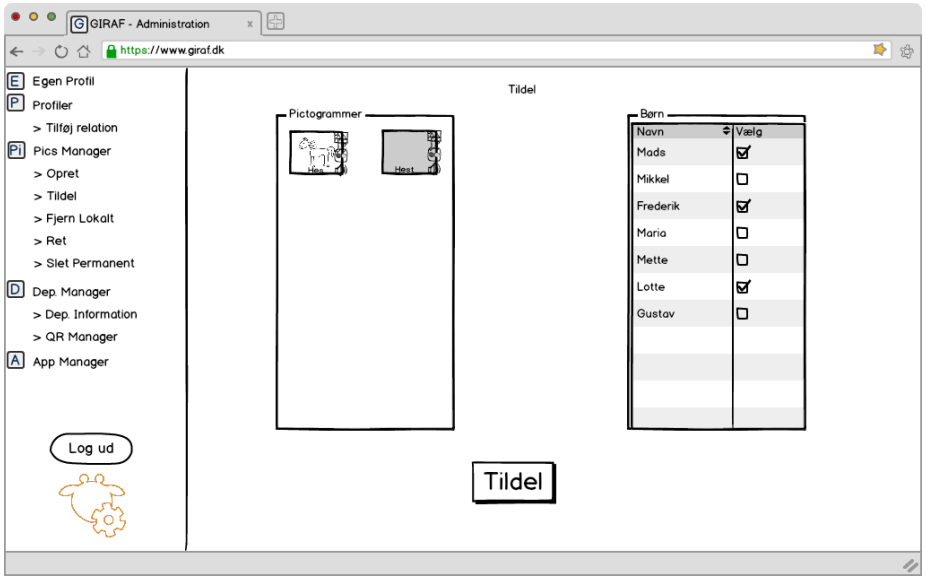


Figure B.22: Pics Manager Assign

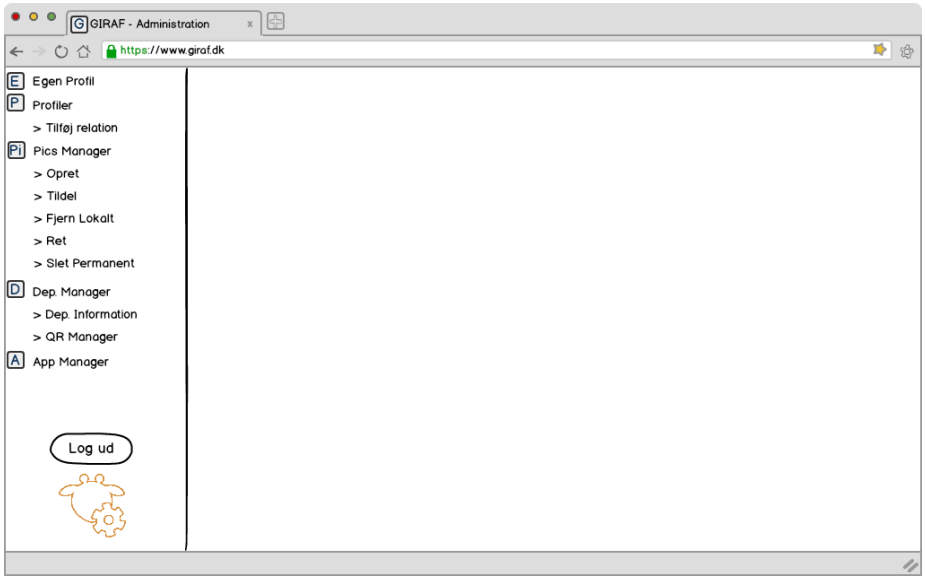


Figure B.23: Profiles

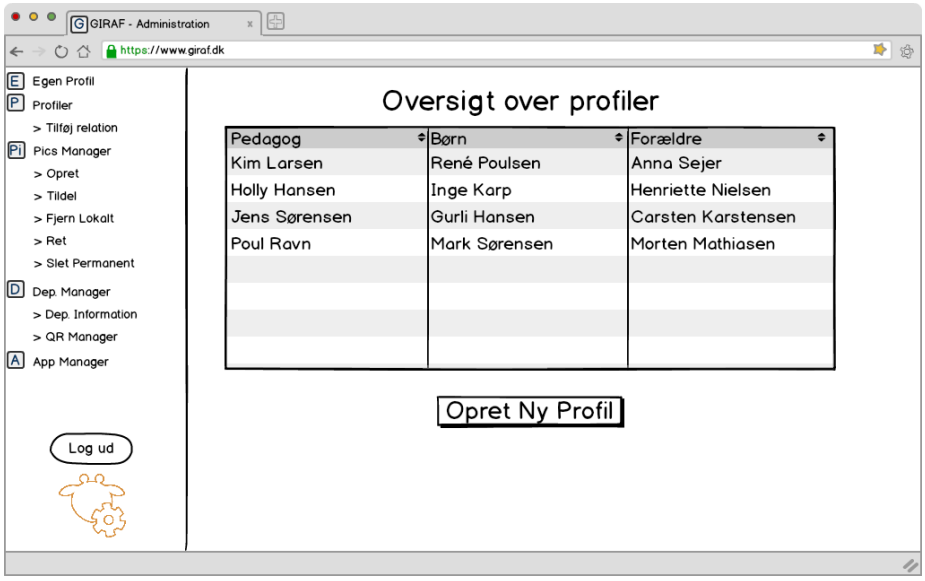


Figure B.24: Profiles Department Manager

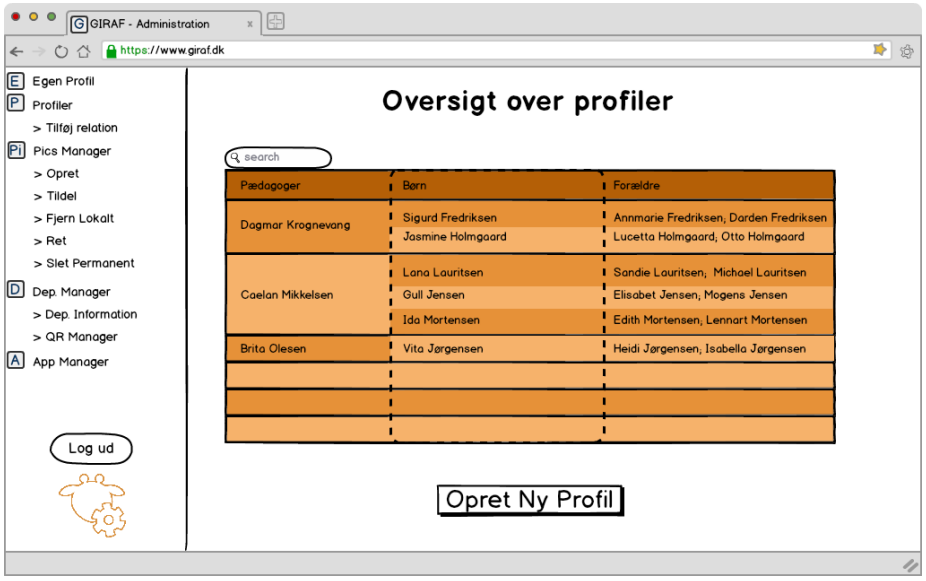


Figure B.25: Profiles Department Manager 2

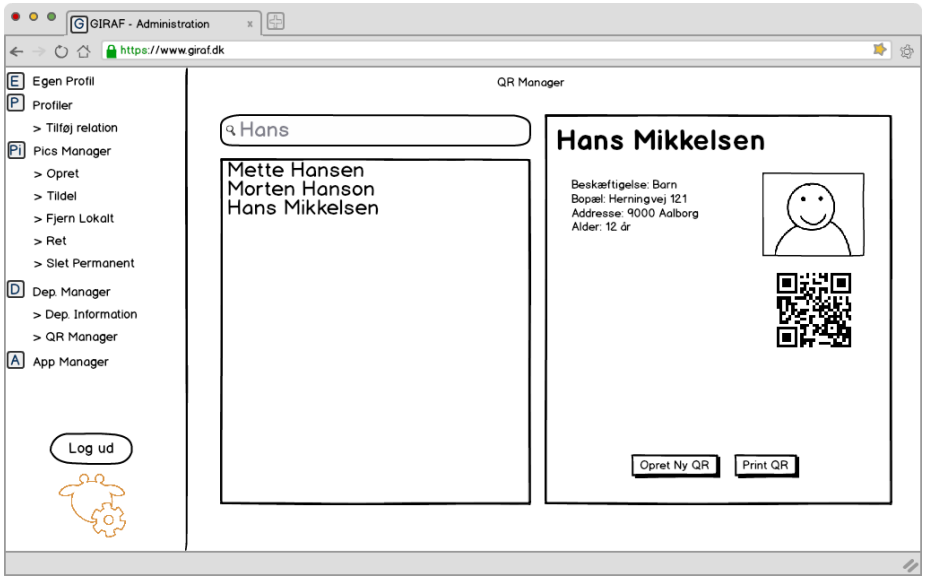


Figure B.26: QR Manager

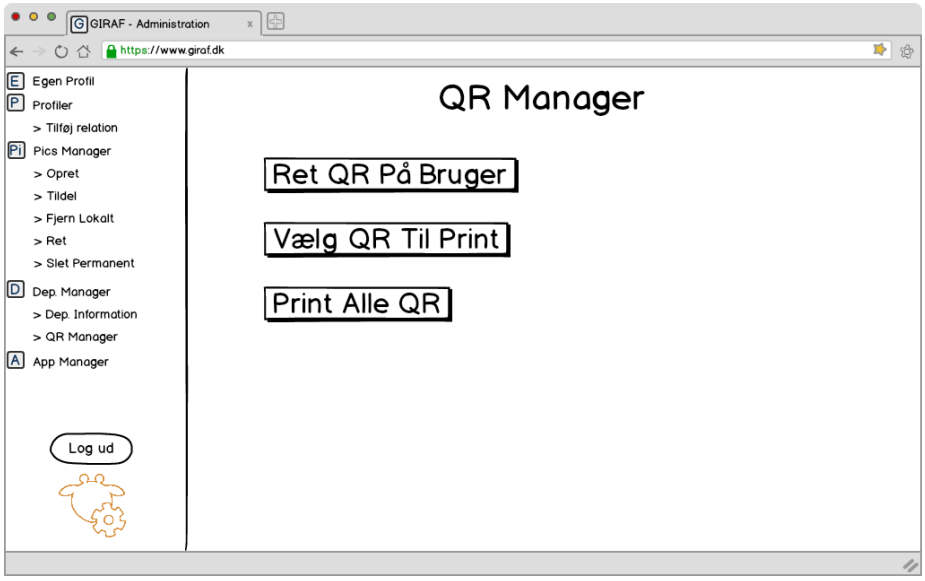


Figure B.27: QR Manager Frontpage

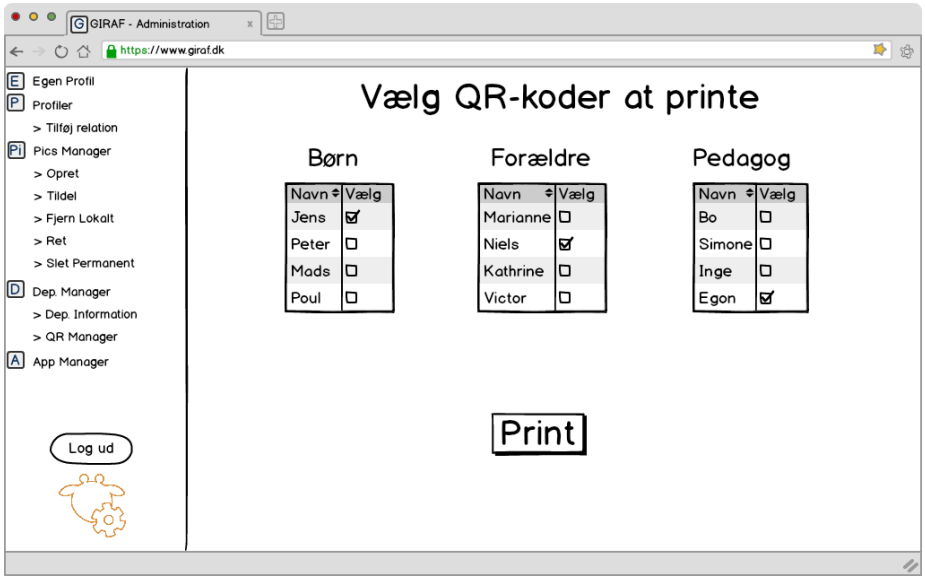


Figure B.28: QR Manager Pick

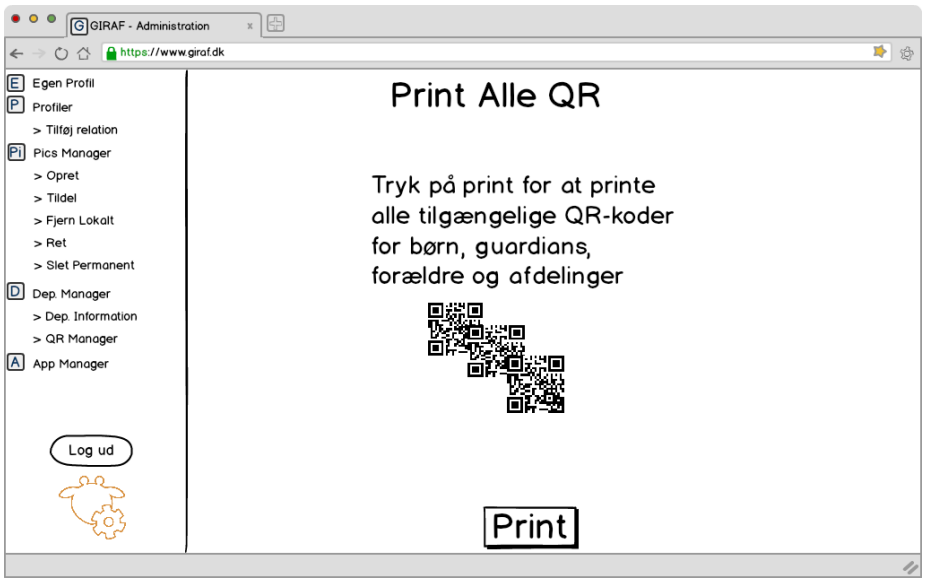


Figure B.29: QR Manager Print All

Appendix C

Transcript of Meeting with the head of the kindergarten (1/3-2013)

The questions we brought with us:
A: Does the parents have a common contact list between each other, like the ones used in the, Danish, elementary school?
B: How often do you change the kids pictograms? - As in all the way out of the scrap book
C: How many kids does a pedagogue take care of, on average?
D: How is the parents allowed to influence the daily life of their kids, when the kids are at the kinder garden?
E: Does the department manager have a list of the e-mails of each parent and are they easy to find? - If not, how easy would it be to gather them?

Answers to these questions:
A: The parents are not allowed to know anything about each other or the other parents kids.

B: It doesn't happen that often, but they still need to be able to change the pictures really fast. Since sometimes the kids come in with a new jacket, and cannot put their jacket back on, or take it off because of the wrong visual stimuli from an old picture.
The same goes if the milk carton suddenly changes to a Christmas theme or such.

C: A pedagogue usually have the responsibility for 2-3 kids. - These are called "kontakt børn".

D: The parents have no influence. Actually if their kids cannot participate in outdoor activities because of illness or tiredness, the parents are told to keep

their kids at home if the plan for the day involves outdoor activity, (which they at Bækken do almost every day).

E: She has their e-mail, but they are not public accessible. This still makes it possible for her to send invites to the system. - We were informed that the registration forms for when a kid enters the kinder garden has been updated so that the parents are asked to fill in their e-mail address.

Information gathered for PictoCreater group:
- The text should not be stationary.
- There should be an option for drawing lines and removing parts of a pictogram.
These two lines were derived from what she told us of their current pictogram editor system.

Possible future work:
1. A planning tool. The department manager uses a certain amount of time on scheduling the kids and pedagogues day. One of the most serious issues could be that one of her pedagogue had to take the day off, then she must make sure that all her schedules get taken care of. A feature that they could really use would be that of fixing an assignment for each second week.
This planning tool should also be able to print two different versions of the schedule. One for the parents and one for the pedagogues. - The parents are not supposed to know which kids do what, or with whom. They only need to know what general activities is taking place in the kindergarten.
She also suggested that the taxa schedule could be incorporated in this system.

Other observations:
"Profiler" - We noticed that the tab "Profiler" from our original design was of no use to anyone but the department manager. Since everyone else only had access to a certain list of persons, that we already displayed on "Egen Profil".

"Profiler" - Should be ordered in a table, where first the pedagogues name is, the next <td> should then contain the kids they are responsible for, the next <td> should then contain the parents of this kid, and the last <td> should then contain other relevant persons or information about the kid. (<td> is the same as a cell in a table)

"Parent/Pedagogue contact" - They don't communicate all that much in person. Since a department can have kids from basically all over the country. The kindergarten do host some 'parent, come and see the kindergarten nights'. But outside that they only communicate via phone and this black book the kids bring to and from the kindergarten. In it important information like, if the kid did something special today, or the reason why he arrived home with a new pair of pants on, and so forth. - The pedagogues always check this book when the kid arrives and so does the parents.

”Privacy setting” - The privacy setting private is known as ”mine tavler” by the pedagogues.

”Privacy setting” - We need 5 different privacy settings:
Pedagogues only
Parents only
Guardians (”værger”) - We need to confirm this phrasing with the head of the kindergarten
Department
Public

”Searching” - When searching through tags or categories, it should search on synonyms as well.

”Breadcrumbs” - There should be breadcrumbs on every screen, or if there is only one action on the screen, it should contain its title.

”Pictogram ’Edit’ and ’Opret”’ - We forgot to add category and tag adding to these functions.

”App Manager” - She found it intuitive to perform special settings on the apps in the app manager. - This could be an interesting design idea, but hard to implement.

Appendix D

Usability Testing Appendix

This chapter contains the introduction given to the test persons and the tasks they had to perform. This chapter is written in danish, as was the test.

Opgaver:

- Ændre egen profils information, her i blandt: Navn, mail, tlf., profil billede
- Opret nyt barn
- Opret barns forældre
- Opret pædagog
- Opret relation mellem forældre og barn
- Opret relation mellem pædagog og barn
- Ret information om afdeling
- Opret pictogram
- Opret kategori

Første Login:
Du har nu fået en helt ny bruger i Giraf systemet, og opdager at dit navn er stavet forkert og der mangler tlf. nummer og et profil billede.
Vi vil derfor bede dig om at opdatere følgende: Dit navn, dit tlf. nummer og uploade dit profil billede.
Vi har lagt et billede på skrivebordet med navnet "profilBilledePige.jpg" og "profilBilledeDreng.jpg"

Nyt barn i afdelingen:
I morgen starter det nye barn, Thorsten Jensen, i afdelingen og han har derfor

brug for at få oprettet en profil. Hans forældre hedder Lotte og Mads Jensen. For at Lotte og Mads også skal have mulighed for at benytte systemet derhjemme skal de også oprettes i systemet.
Lottes e-mail: lotte.jensen@test.dk
Mads' e-mail: mads.jensen@test.dk
Vi vil bede dig om at oprette en profil til Thorsten, Lotte og Mads.

Hvems forældre er det?
Nu har Thorsten, Lotte og Mads fået oprette en profil. Men før at Lotte og Mads kan bruge deres profiler til noget skal de forbindes med Thorsten.
Vi vil derfor bede dig oprette en forbindelse mellem Thorsten og Lotte, samt en forbindelse mellem Thorsten og Mads.

Den nye Pædagog:
I dag starter den nye pædagog Henriette Poulsen også i afdelingen, hun skal have ansvaret for Thorsten.
Henriette Poulsens informationer:
E-mail: henriette.poulsen@test.dk
Tlf: 88 44 55 66
Add.: Mark Poulsen Vej 23 – Aalborg Øst 9220
Vi vil derfor bede dig om at oprette en profil til Henriette og dernæst oprette en relation mellem Henriette og Thorsten.

Ny Tlf. i afdelingen
I har fået besked ovenfra på at skifte jeres telefon nummer i afdelingen. I har i dag modtaget det nye telefon nummer.
Det nye nummer er: 87 78 23 44
Vi vil nu bede dig om at rette informationen om jeres afdeling så den passer.

Pictogram af de nye shorts:
Thorsten har lige fået nye shorts og I er blevet tilsendt et billede af Thorsten hvor han er iført sine nye shorts.
I vælger for at være på forkant med Thorstens pictogram kartotek at oprette et pictogram af hans shorts.
Vi vil derfor nu bede dig om at oprette et pictogram som hedder Shorts.
Billedet til pictogrammet kan findes på skrivebordet: "shorts.jpg"

Introduktion:
Velkommen til testning af GIRAF admin systemet.
Vi vil stille dig en række opgaver, skrevet ned på papir, som vi gerne så dig udfører på den fremstillede PC. Testens formål er for os at drage indsigt i din måde at arbejde på, vi vil derfor bede dig tænke højt imens du udfører opgaverne. – Vi vil her gøre opmærksom på at intet er for småt til at vi gerne vil hører det.
Igennem testen vil <navn> sidde med dig i testrummet mens du udfører op-

gaverne. Hvis du har spørgsmål til opgaverne vil <navn> hjælpe dig. Han vil også give dig den næste opgave når du har gennemført den du var i gang med. Vi optager testprocessen på film og har derfor brug for at du skriver under på at dette er okay. <Udlever underskrifts side og giv tid til at underskrive>

Appendix E

Color Theme

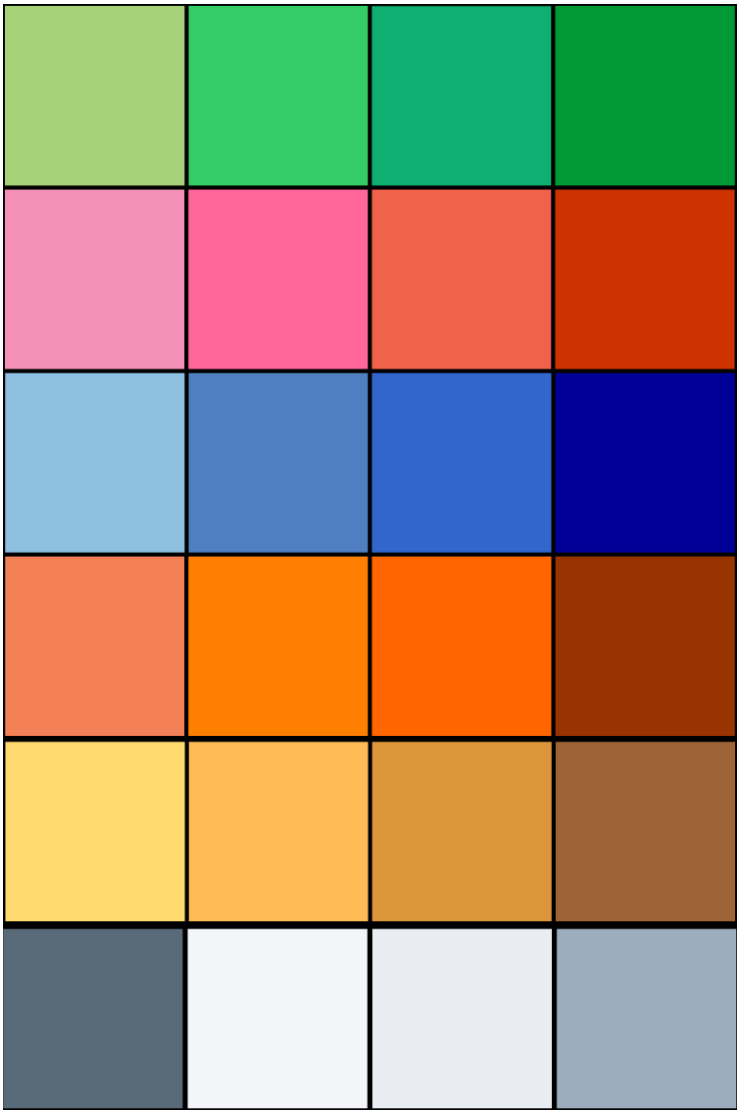


Figure E.1: Color theme for the GIRAF system.

Part VI

Fixme

List of Corrections