Rheinische Friedrich-Wilhelms-Universität Bonn
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Abteilung III

Green Shifting
A mobile application for the efficient usage of renewable energy

Theresa Otte

Master Thesis
Computer Science

Bonn, March 2nd 2015

Supervisor: Prof. Dr. Sören Auer

Abstract

Renewable energy and energy saving have become important topics during the last years. Many efforts have been made to reduce the energy consumption by optimizing devices and making consumers aware of their usage. Though the production of renewable energy has increased, there is still a lot of conventional energy needed, since renewable energy is quite dependent on the weather (as wind power) or on the time of day (solar energy). Also, a large amount of household energy consumption is caused by potential time-shiftable activities such as washing, drying, charging of electronic equipment, etc.
In this thesis, a mobile application is designed and implemented to inform an energy consumer about times with a high amount of renewable energy produced, so he or she shifts the usage of energy consumers to those times and thus use rather renewable energy than conventional. The application is implemented using a cross-platform development tool. To motivate users to actively use it, gamification elements are included. Its usability is evaluated with five test users. The result is Green Shifting, a cross-platform mobile application that encourages users to use the available renewable energy more efficiently.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Statement of the Problem . . . 2
- 1.3 Thesis Organization . . . 2

2 Background and related works **3**
- 2.1 Energy . . . 3
  - 2.1.1 Saving Energy . . . 4
  - 2.1.2 Renewable Energy . . . 5
- 2.2 Mobile app development . . . 5
  - 2.2.1 Cross-Platform . . . 6
  - 2.2.2 Responsive web development . . . 8
  - 2.2.3 Databases . . . 10
  - 2.2.4 Gamification . . . 12

3 Requirement analysis **13**
- 3.1 Functional Requirements . . . 13
- 3.2 Non-functional Requirements . . . 15

4 System architecture **17**
- 4.1 System components . . . 17
- 4.2 Implementation of non-functional requirements . . . 17
  - 4.2.1 Database structure . . . 19
  - 4.2.2 Tool selection . . . 21

5 Implementation **23**
- 5.1 Server structure . . . 23
  - 5.1.1 Data processing . . . 23
  - 5.1.2 Notification Management . . . 24
- 5.2 Application structure . . . 26
  - 5.2.1 Backend . . . 27
  - 5.2.2 Frontend . . . 31
- 5.3 Firebase configuration . . . 34
- 5.4 Prototype . . . 35

5.4.1 Main tab . . . 35
5.4.2 Profile Tab . . . 37
5.4.3 Statistics Tab . . . 37
5.4.4 About Tab . . . 38
5.5 The Web application version . . . 38

6 Evaluation **41**
6.1 Unit testing . . . 41
6.2 Usability tests . . . 42
6.2.1 Test setup . . . 42
6.2.2 Usability evaluation . . . 42

7 Discussion **45**
7.1 Limitations of the system . . . 45
7.1.1 Energy data limitations . . . 45
7.1.2 Application limitations . . . 45
7.2 Future Work . . . 46
7.3 Summary . . . 47

A Screenshots **49**

B Usability evaluation **52**

Bibliography **55**

List of Figures

2.1 Native vs. Cross platform development 7
2.2 Fluid grid example on different screen resolutions 9

3.1 Use Case diagram of energy data usage 14
3.2 Use Case diagram of scoring functionality 15

4.1 Components of the system and their connections 18
4.2 Mockup of the home screen GUI for two screen resolutions . 19

5.1 Interaction of the different components 26
5.2 Sequence diagram of the tweet action 29
5.3 The use device activity 32
5.4 The first impression on a tablet 35
5.5 The different views of the application 36
5.6 Screenshots of the web application 39

A.1 First actions 49
A.2 Setting a reminder 50
A.3 Device usage 51

List of Tables

2.1 Overview of the platforms and the required language skills . 6

4.1 Messaging services for different platforms 18
4.2 Overview on the different platforms supported by PhoneGap and the needed operating system for their SDK 22

5.1 User actions and their resulting scores 29

6.1 Overview on the tests and on which devices they were executed 41
6.2 User opinions about renewable energies 43
6.3 User suggestions for improving the mobile application . . . 44

Listings

4.1 Database structure in JSON syntax 20

5.1 Example CSV file structure 24
5.2 Push Notification Management 25
5.3 Storing data in Firebase 27
5.4 Retrieving data from Firebase 27
5.5 Using the social media plug-in 28
5.6 Firebase query with ordered result 28
5.7 Setting a reminder 30
5.8 Deleting a device ID 31
5.9 Definition of the navigation bar 31
5.10 Creation of a pie chart 33
5.11 Grid layout example 34
5.12 Firebase rule set 34

6.1 QUnit test of scoring.increaseScore() 42

Chapter 1

Introduction

In times of climate change and criticism of nuclear power, renewable energies have become an important topic. Many governments have decided to shift more and more electricity production from the conventional resources coal, gas and nuclear power to the renewable resources sun, wind and water. Nevertheless, especially solar and wind energy are depend highly on weather and daytime, which means that there is still a lot of conventional energy needed. On the other hand, a large amount of household energy consumption is caused by potential time-shiftable activities such as running the washing machine or the dryer, or charging an electronic device.
In this thesis, the mobile application "Green Shifting" is designed and implemented which aims to collect data about current energy production and inform the user about times with a high amount of renewable energy in which the time-shiftable household activities can be executed to optimize the usage of green energy.

1.1 Motivation

In times of growing ecological awareness, the future is dedicated to renewable energies. Sustainability is important to many consumers but with the mix of different energies in the grid it is hard to know when the electricity is generated from renewable resources. An advantage of knowing about that matter is that aware users can shift their energy consumption to times with a lot of renewable energy in the grid. As a positive side-effect this behaviour improves the balance between demand and supply and the grid becomes more stable.
Since computers are available everywhere nowadays and especially the coverage with mobile devices in the population of Europe is very high,

it needs to be a task for the computer researchers to support promising technologies like renewable energies. This can be done by providing applications that assist energy consumers with the handling of energy.

1.2 Statement of the Problem

Energy consumption is a topic that affects almost everyone, but they can also influence the total energy demand by changing their consumption behaviour. The goal of this thesis is to create a system that supports users who are interested in improving their own renewable energy awareness. Everyone can easily shift the usage of some electronic devices like washing machines to time slots when there is a lot of renewable energy produced. They only need to know when these times are. Therefore, data about the produced energy needs to be requested from the providers and published in a way that as many people as possible can access and use it. It is the concept of the Green Shifting system to fulfil exactly these requirements by collecting the energy data, processing it and presenting it to the end user. The publication is achieved via a mobile application that visualises the energy information. To reach as many users as possible this application needs to be available for the most common mobile platforms. This thesis describes the development of this system.

1.3 Thesis Organization

In order to implement such an application, chapter two presents several tools and gives an overview on related works. In chapter three, the system requirements are analysed and in chapter four the system components are introduced. The implementation of the functional requirements is described in chapter five. Chapter six covers the evaluation, which includes unit testing as well as a usability analysis. Finally, chapter seven gives a summary on what this thesis has reached, where its limitations lie and how it can be extended in future works. In the appendix further information about the prototype can be found. Also, details about the usability analysis are listed, covering the questionnaire and an overview on the testers.

Chapter 2

Background and related works

2.1 Energy

Energy nowadays is produced in many ways from many resources. The most common forms are coal, nuclear energy, petroleum, gas, biogas, solar, wind and water energy. There is a lot of criticism about some of these resources:

- The amount of coal, petroleum or gas left is only limited.
- Another problem is that these resources emit a lot of carbon dioxide when they are burned. In the report of the U.S. energy department [Adm09] it is shown that the power sector produced 40% (2,160.0 million tons of 5,425.6 million tons) of all CO_2 emission in the USA in 2009.
- During the last years there have been several incidents with nuclear energy, the most critical and best known ones happened in Chernobyl (Ukraine) 1986 and in Fukushima (Japan) 2011. Both accidents had a big influence on the nearby living people, who had to be evacuated. The areas around the power plants have been forbidden to be entered for a while and people have not been not able to return to their homes.
- Nuclear energy also has the big drawback that it produces nuclear waste that is radioactive for a very long time. This waste has to be stored somewhere save for thousands and millions of years until its radioactivity has reached a low level. It is very expensive to subdue suitable storage areas. Until today, no such areas could be found that will be save for a long enough time and only interim solutions are used that need to be replaced only in a few years.

On the other hand, solar, wind, water and biogas energy are renewable. Still, they do have some drawbacks. Wind power is worthwile only in certain regions. To supply the other regions, it is necessary to extend the current power line system. To store energy, water reservoirs are needed. They cause a huge intrusion in the nature.
However, renewable energies emit less carbon dioxide and are not as risky as nuclear power. Sunlight, wind and water are also not going to be depleted, unlike coal or gas, which exist only as limited resources. These are the reasons why many governments, including the European Union, decided to change the energy market. In the following, two methods to reach this are presented.

2.1.1 Saving Energy

One big idea behind the energy transition is the reduction of energy consumption. With less energy needed, less energy is produced and thus the problems caused by some types of energy are reduced. Energy saving can be achieved by replacing electronic devices with a high need of energy by more efficient ones. Another way is to reduce the time of consuming energy, like turning off the heater when leaving the apartment or opening the window, shutting down a computer after usage or disabling the standby mode on electronic devices. Although these methods are quite easy, consumers are often too lazy to use them. To deal with this issue, there has been a lot of research to support consumers in their energy saving behaviour.
2009 Guinard et al. proposed a system that wirelessly measures the energy consumption of various devices by using sensor nodes of intelligent power outlets [GWT09]. They additionally designed a web based API and web UI for a user to monitor and control the energy consumption via a web browser.
In the same year Kim et al. developed ViridiScope, a fine-grained power monitoring system [KSCS09]. This self-calibrating tool enables a user to monitor the power consumption of every device in the home by using ambient signals from sensors placed near the device. The self-calibration is done by a model-based machine learning algorithm.
In [MSW10] Mattern et al. present the eMeter system, a mobile application that enables a user to log their energy consumption in a household and then evaluate it. For this they use smart electricity meter which measure the energy consumption in a household and are being deployed on a mandatory level in some countries of Europe.
Energy saving is only a limited method. There will be a point when devices cannot be significantly optimized in their energy consumption. This is why further concepts are needed.

2.1.2 Renewable Energy

Renewable energies like solar, wind and water power have the great advantage that they do not consume resources. Nevertheless, their biggest drawback is that they are not always available in the same amount. Especially solar energy and wind power depend a lot on the weather. This can lead to major fluctuations in the energy supplied to the power grid which is dangerous, because the grid frequency needs to be constant. Otherwise interferences in the network can occur and cause major problems. Also, power plants producing conventional energy often cannot be turned off very fast. Especially nuclear power plants need several days to shut down and to be activated again. This makes nuclear energy not very flexible. To maintain a constant power grid frequency, the consumed energy needs to equal the produced energy. One way to use this additional energy is to store it by pumping water into reservoirs for generating water power at a later stage.
Another way to deal with this problem is to regulate the prices for electricity on a base of supply and demand. This means in times with much energy in the power grid, caused by generating a lot of electricity from renewable resources, the energy prices can even turn negative to encourage the consumption of energy. With this in mind, shifting the energy usage from times of low renewable energy production to times with a high level of renewable energy can even save money: The consumers pay less for their usage, while due to an increased energy request the prices can increase again. This might be appealing to the energy providers to enable users to change their energy consumption behaviour by sharing the data of available energy resources. Another approach tries to store this energy by pumping water to reservoirs for later production of water power.
Smart devices can benefit of the data about available energy by consuming or storing energy in peak times, while the device owners can profit by the low prices. This has the advantage of an improved balance of supply and demand. A system that exploits this effect needs many data about the energy, about weather forecasts and about energy consumption behaviour. With this information an algorithm can be developed to provide a real-time forecast about energy supply.

2.2 Mobile app development

Developing applications for mobile devices can be done in two ways. The first way is programming it in the native language of the devices' platforms, which differ in matters of software development kits (SDK), hardware, application programming interface (API) and so on. Table 2.1 shows

an overview of the different platforms and their native programming language.
The second way is developing a web application using a common feature of all platforms: a mobile browser that is programmatically accessible from the native code [CL11]. The drawback of web applications compared to native applications is that they are slower and not so powerful when it comes to system specific features, such as UI or access to hardware components. On the other hand, web applications, which usually consist of HTML5 and CSS for the UI and JavaScript for the programming background, are compatible between the platforms.
Several tools have been developed to close that gap between web applications and native applications. In the next section some of them are presented.

Platform	Main Programming language
Google Android	Java
Apple iOS	Objective C
RIM BlackBerry	Java (J2ME flavored)
Windows Phone 7, 8	C#
HP Palm webOS	HTML/CSS/JavaScript
MeeGo, tizen	C, C++, HTML/CSS/JavaScript
Samsung bada	C++
Canonical Ubuntu	C, C++, JavaScript
Symbian	Symbian C++
amazon fire OS	Java
Firefox OS	HTML5/CSS/JavaScript

Table 2.1: Overview of the platforms and the required language skills

2.2.1 Cross-Platform

In figure 2.1 one can see that on the one hand in matters of performance and UI native applications are unbeaten, but on the other hand, if an application needs to be developed for several systems, web applications are way cheaper need less time. Cross-platform applications are some kind of compromise between web and native applications with the support of device functionality and the benefit of portability.
In [OT12] Ohrt and Turau define four classes of mobile applications in the context of cross-platform tools which are mainly grouped in two categories: *Native applications* have compiled source code and are thus SDK dependent, while *interpreted applications* use a virtual machine (VM) running on the platform. The four classes are:

- purely native applications with direct access to the system API,

- purely native applications with a library to access the system on an abstraction level,
- interpreted applications with an included VM in the app itself and
- interpreted applications with a separately installed VM

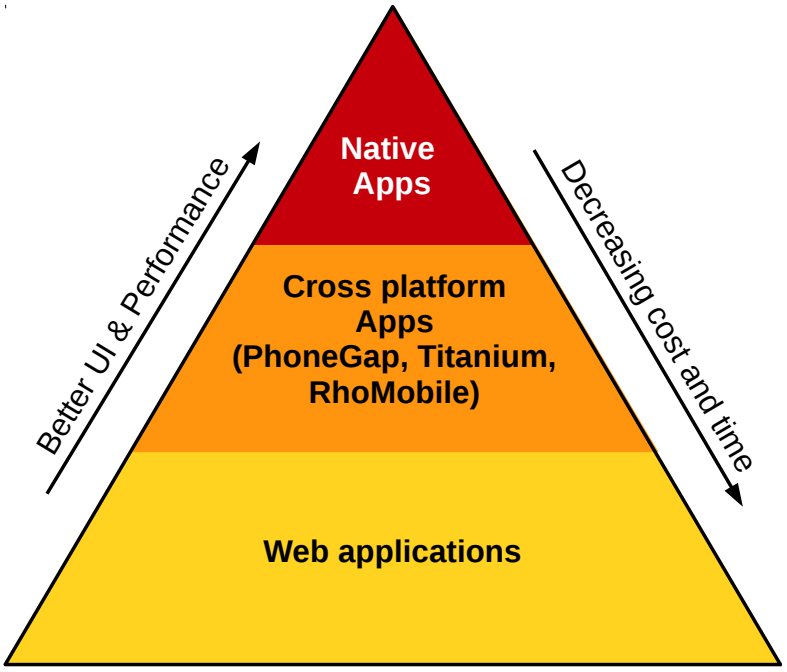


Figure 2.1: Native vs. Cross platform development

In the following some of the cross-platform tools which are compared by Ohrt and Turau are presented.

- **Adobe PhoneGap** [Inc15], also known as Apache Cordova (former name), is an open-source cross-platform development tool for a large number of platforms. It supports Android, iOS, BlackBerry 10, fireOS, Firefox OS, Ubuntu, Windows Phone 8 and 8.1 and tizen. The UI is defined via HTML and CSS, while the functionality is programmed with JavaScript and interpreted by the web view element. For building the application for the different platforms, the platform specific native SDKs are needed. PhoneGap provides an interface for accessing a device's hardware, these functionalities can be integrated with a plug-in. Several plug-ins are already implemented, but it is also possible to develop a custom plug-in as an extension, using native code.

- **Appcelerator Titanium** [app14] provides a stand-alone SDK with an IDE built on the Eclipse platform. All source code for the application is written in JavaScript. Titanium offers several APIs for accessing system functions and creating a GUI. The source code is combined with a JavaScript interpreter and some static content during the compilation. Finally the results are packed into the app. During runtime the JavaScript is processed by the interpreter. Titanium supports Android, iOS and Windows Mobile.

- **RhoMobile** [MS15] by Motorola Solutions relies on the Model-View-Controller system architecture. It uses Ruby for the application functionality, while the GUI is designed with HTML, CSS and JavaScript and is integrated into the app as webpages by using an embedded webserver. eRuby is also supported. With RhoStudio an Eclipse plug-in exists for the development and debugging of the application. The Ruby code is converted into byte code during compilation and then wrapped with a VM into the application package. RhoMobile supports Android, iOS and Windows Phone 8.

In their paper Ohrt and Turau show several tables which list features of these (and more) cross-platform development tools. PhoneGap has the big disadvantage compared to the others that the APIs of the platforms need to be installed while the other presented tools bring their own API. But PhoneGap also supports the most platforms and in the paper it is shown that at least for Android, PhoneGap is faster than the others and also needs less memory.

2.2.2 Responsive web development

An important aspect during the development of mobile applications is the UI. Since mobile devices not only come with different platforms, but also with different screen sizes, varying resolutions have to be taken into account during the designing process. Websites with fixed content optimized for the resolution of desktop screens might be unusable when they are shown on smaller screens like those of smartphones or tablets. Responsive web design (RWD) aims to solve this problem. There are several techniques that are used in RWD which are described by Marcotte in [Mar11] as follows:

- **Flexible/fluid Grid** is one of the most important techniques. Instead of designing web pages with a fixed width and centered content, elements are placed on a calculated percentage width. Although this

makes a website resizable with resolution changes, there might appear some problems if a multiple column design is shown on a smartphone screen. To deal with this, a designer can either make two versions of the website or he can use media queries (see second next item). Figure 2.2 shows how a website using fluid grids is displayed on different resolutions.

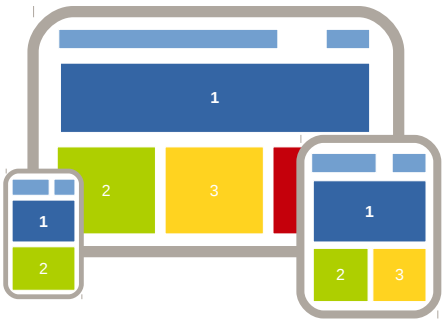


Figure 2.2: Fluid grid example on different screen resolutions

- **Flexible/fluid Images** deals with similar problems with images. Images can be too wide for the screens of mobile devices if a fixed size is used for all resolutions. This might also cause long loading times, since mobile internet connections are often not that fast. These problems can be handled by either using CSS cropping (dynamic resizing), the Max-Width Attribute by Richard Rutter (see [Rut] for some experiments) or with multiple images for the different resolutions.

- **Media Queries** allow to provide several CSS styles and select automatically the best fitting one. This is possible because with media queries information about the user's screen size is available.

Of course there already exist several libraries to support the process of RWD. In the following, some tools based on the fluid grid approach are presented:

- **Twitter Bootstrap** [OTB] is the most popular framework for responsive, mobile first web development. Interface components such as buttons, forms, charts, typography and navigation are supported via HTML and CSS-based design templates. The framework is extensible with JavaScript and supports all common types of browsers, including Chrome, Firefox, Safari, Opera and the Internet Explorer (IE) from

version 8+. Bootstrap uses a dynamic grid system with four different sizes of screens (xs for smartphones, sm for tables, md for laptops and lg for large desktop screens) that rescale with the resolution. The screen is divided into 12 colums. Additionally Bootstrap supports and uses media queries.

- **ZURB Foundation** [ZUR13] is an advanced front-end framework for RWD based on a fluid grid system with 3 different screen sizes. It has on the one hand CSS styles for buttons, forms, images, headings, navigation systems and other interface components, on the other hand jQuery-driven components such as dropdown menus, tooltips or site-tours. All in all it is more style agnostic than Bootstrap. Foundation does not support Internet Explorer 8 and lower versions, but IE 9+ and Chrome, Firefox and Safari.

- **Skeleton** [Gam11] is a responsive web design framework that uses a fixed grid system. It is style agnostic, which means it provides only limited UI tools. The developer is supposed to use Skeleton as a basis and put his own design on top of it. Skeleton supports on desktop devices Chrome, Firefox, Safari and Internet Explorer 7+, on mobile devices only the browsers on iPhone, iPad and Android.

The presented tools are actually developed for designing web applications. With cross-platform development tools they are also suitable for developing mobile applications. Their main differences lie in their provided UI tools and the number of resolutions.

2.2.3 Databases

With cloud computing becoming more and more popular, an increasing amount of data needs to be dealt with. In the past the data was usually stored in relational databases without respect to the actual data model. Relational databases also provide more functionality than often needed which leads to an unnecessary overhead. If for example a database is only used to log information, it will not make use of *join* operations. Especially in the case of big data stored on distributed systems, some of these functionalities slow the processing of requests down while actually a real time response is needed. So the scalability of a database management system (DBMS) becomes more important. With NoSQL [EFH+11] there is now an alternative to relational databases. In the following, a short overview over the features of both systems is given.

- **Relational Databases** (RDB) were introduced by E. Codd in 1970 [Cod70] and are based on relations, a mathematical description of ta-

bles. For queries, the structured query language (SQL) is used which implements the concepts of relational algebra. A table consists of columns and rows where the column names are attributes and the rows describe tuples or records. Each record needs to be identifiable which is achieved by using keys. Keys are attributes which are unique and for every record, for example an ID. They must not be changed. A table can be linked to another by using the other's key as foreign key. The main operations described by relational algebra are: Projection, selection, cross join, rename, union and set difference. An important principle of RDB transactions is ACID which stands for atomicity (only one transaction at a time), consistency (transaction states need to be consistent), isolation (transactions do not influence other transactions) and durability (the result of a transaction cannot be change without making a new transaction).
RDBs are not the best solution for every data model. The most famous problem is the so called object-relational impedance mismatch [Amb00] where objects from object-oriented languages are mapped to relational databases. Objects are often too complex for the flat relational structure which makes the mapping difficult.
Nowadays, data is often stored on distributed server systems. This does not match the approach of RDBs very well.

- **NoSQL** [EFH+11] means "not only SQL" and describes non relational, distributed database systems. Often the ACID rules in transactions are violated. There are several types of NoSQL databases, like column, document, key-value, graph and multi-model, based on the data model.
  In [HJ11] the authors evaluate several NoSQL systems. Their results can be used as advise which system should be used. They have a closer look on what possibilities to make queries a developer has, like REST API, Java API, a supported query language for more complex queries, or MapReduce for calculation intensive queries. They also describe how the systems deal with concurrent transactions (lock, optimistic lock or multiversion concurrency control). The third criteria is the way of partitioning (range based or with consistent hashing). Their last point is replication and consistency.
  Since NoSQL provides distributed data storage, it is often used by cloud solutions and big companies like Google or Amazon.

2.2.4 Gamification

To make an application successful, it is important to motivate users to actively use it. Gamification can improve the user activity and retention by

combining game design elements with non-game context [DDKN11]. In this definition, the term *game* needs to be distinguished from the term *play*. The later one is used for games and toys and describes a free, intuitive, unscripted behaviour, while gaming follows rules to reach a determined goal. Gamification also just describes the implementation of game design elements, so the resulting system is not a full game.
The three most important principles in gamification are *relatedness*, *competence* and *autonomy* [Gro12]. Relatedness means the connection to personal goals, a community of interest, creation of a story and the awareness of social context meanings. To fulfil the competence principle, a system needs to provide interesting challenges, clear, visual, varying, well structured goals, feedback and be aware of unintended behaviours. Autonomy describes that no one is forced to use the gamification elements, autonomy is never lost and that activity is not void.
Gamification elements can be anything connected to challenges, like points, leaderboards, achievements and levels, but also an underlying story or a visualisation of progress and the option to give feedback. In [HKS14] Hamari, Koivisto and Sarsa survey various studies about gamification to find out if it works. Most of them tested points, leaderboards and achievements. As a result Hamari et al. found out that these studies suggest that gamification indeed does work. However, the positive effects exist only due to the considered relationships between gamification elements and studied outcomes according to most of the quantitative studies. Qualitative studies revealed that gamification is more complex as assumed. A great influence is given by the role of the context being gamified and the quality of users. Their literature review also revealed some methodological limitations of the studies.

Chapter 3

Requirement analysis

Software development starts with a requirement analysis. It discusses functional and non-functional goals of a project. Functional goals describe the system behaviour and specific functionality which are important during the *design* of a system, while non-functional requirements specify the system *architecture*.
Green Shifting is based on the results of a lab project by Hajibaba, Golchin and Henk [HGH14]. Their application has similar approaches as this one but they only developed it for Android. That means, their source code is not directly reusable. Nevertheless, this requirement analysis is based on theirs.

3.1 Functional Requirements

- **Energy Data Processing** is the most important requirement. Without energy data, the application would be worthless. This requirement consists of accessing the data from the energy providers and processing it to make it usable for the application. By analysing the data predictions about future energy production and consumption can be made.

- With **Data Visualisation** the processed energy data is shown to the user. This includes current energy data as well as predictions. Figure 3.1 contains a use case diagram which illustrates the usage of the energy data by the application.

- **Gamification and Account Management** are two important requirements. With gamification the user is motivated to use the application. This is realised by enabling the user to gain points with different ac-

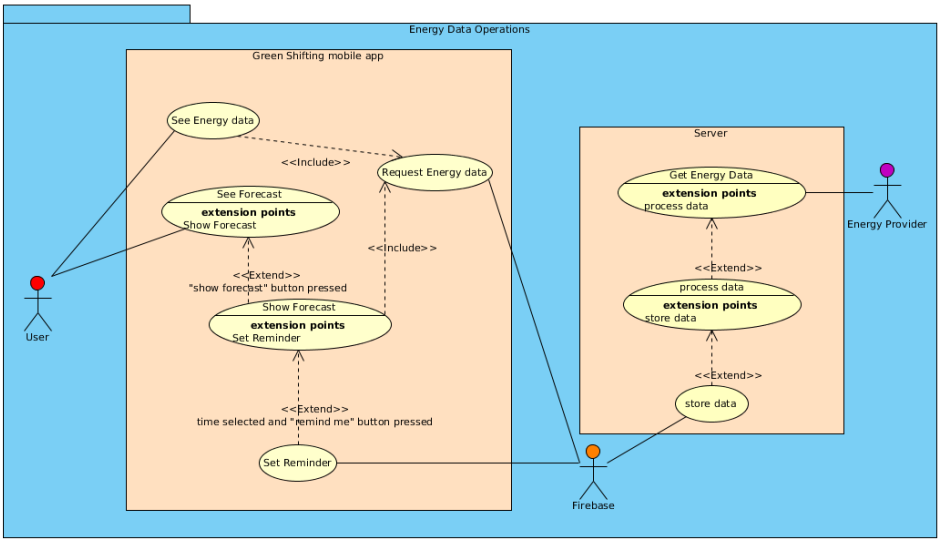


Figure 3.1: Use Case diagram of energy data usage

tivities: He can increase his score by recharging the device at times with much renewable energy in the power grid. Another option is to set a reminder according to the energy prediction to a point in time when the percentage of renewable energy is high. The third way to get points is by sharing the score on social networks. Figure 3.2 shows a use case diagram with the scoring operations.
To save a score, a user needs to be identified. This is achieved by creating an account for each user. Before he can use the gamification elements he needs to be authenticated. A user is also able to select a username.

- **Statistics** that visualise the composition of the user score are shown. Also a leaderboard of the users with the highest scores is included.
- The **User Interface** is required to interact with the user. This is mainly realised with a GUI.
- A **Server** is used to request reminders from the database and send the notification to the recorded device.

3.2 Non-functional Requirements

- **Accessibility** is an important non-functional requirement. It guarantees that the system is accessible from as many platforms as possible.

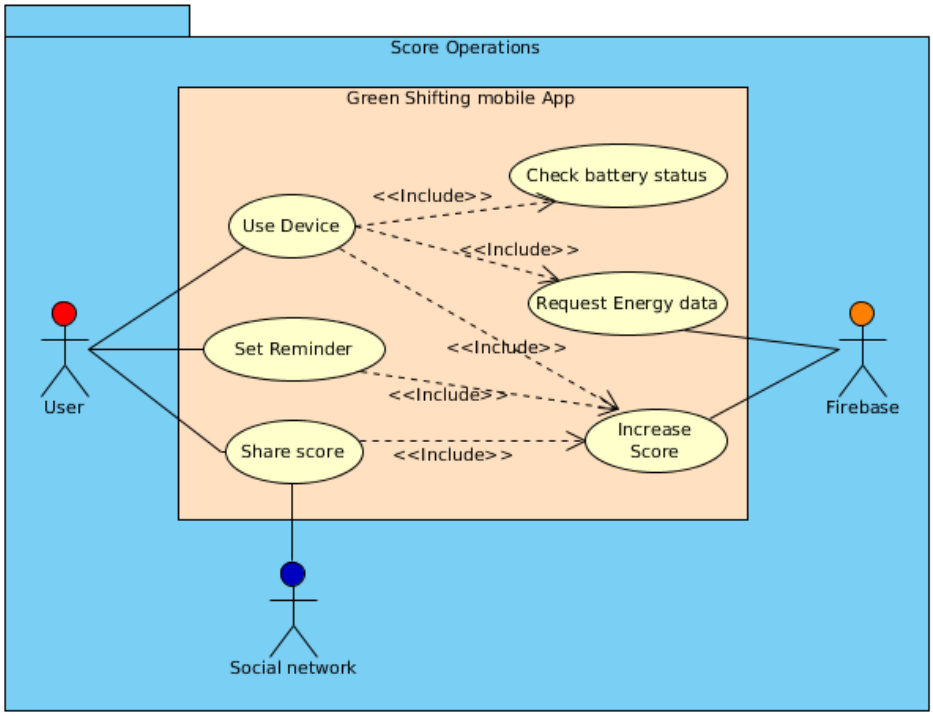


Figure 3.2: Use Case diagram of scoring functionality

- **Usability** influences mostly the UI and means the system is easy and intuitively to use without much background information.
- The **Availability of the internet connection** has effects on the currentness of data. If there is no connection, the data shown in the application might not be up to date.
- With **Reliability** the correct processing and visualisation of the data by the system is ensured. Since the data depends on other resources and processed to match the Green Shifting format, a change in the format of the provided data might cause major problems. The provider might also decide not to deliver their data any more.
- There are several **Interfaces** needed. On the one hand the interface to the data providers to collect the data, on the other hand an interface to social networks for the gamification purpose.

Chapter 4

System architecture

This chapter describes the setting of the Green Shifting system and the implementation of the non-functional requirements. It includes a look onto the different components of the system and their interconnections and the selection of suitable tools according to the results from chapter 2.

4.1 System components

Figure 4.1 shows the different components of the system. The energy data is collected and processed by a server and then stored on a Firebase database (see subsection 4.2.1). The application enables the interaction of the mobile devices with the database. To send push notifications, the server invokes an event on the system specific messaging servers which send the notification to the registered device. Table 4.1 shows the association of platforms and messaging servers.

Platform	Messaging Server
Android	Google Cloud Messaging (GCM)
iOS	Apple Push Notification Service (APNS)
WindowsPhone8	Microsoft Push Notification Service (MPNS)
Windows8	Microsoft Windows Push Notification Service (WPN)
BlackBerry 10	BlackBerry Push Service (BPS)
Amazon FireOS	Amazon Device Messaging (ADM)

Table 4.1: Messaging services for different platforms

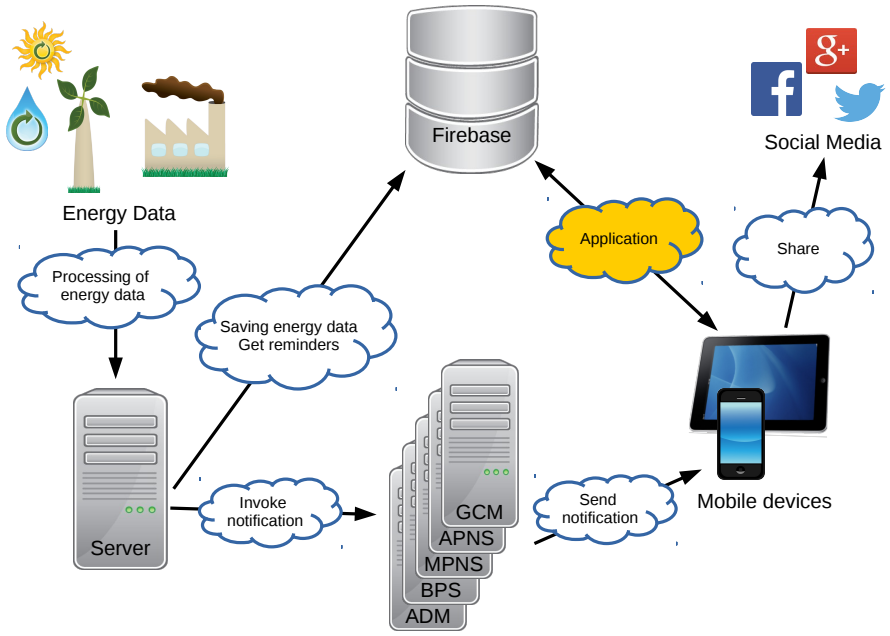


Figure 4.1: Components of the system and their connections

4.2 Implementation of non-functional requirements

The usability is realised by an intelligently designed GUI. It is intuitively usable and self-explaining. Figure 4.2 shows a draft version of the GUI.
Actually, the application itself cannot guarantee that the data is always up to date if there is no internet connection for the mobile device. Anyway, if there is a connection, the data is refreshed periodically. It can also be reloaded manually.
A server is used as interface to the providers. It is meant to fetch and process the data and upload it to the database. The connection to the social networks is realised by using the accounts of the mobile device.
The following subsections give a deeper view into the database organisation as well as into technical details about the used tools and which platforms are supported.

4.2.1 Database structure

The database management system used in the Green Shifting system is Firebase [TLNL]. It is a ”powerful API to store and sync the data in realtime”. Its libraries run on all major web and mobile platforms and can

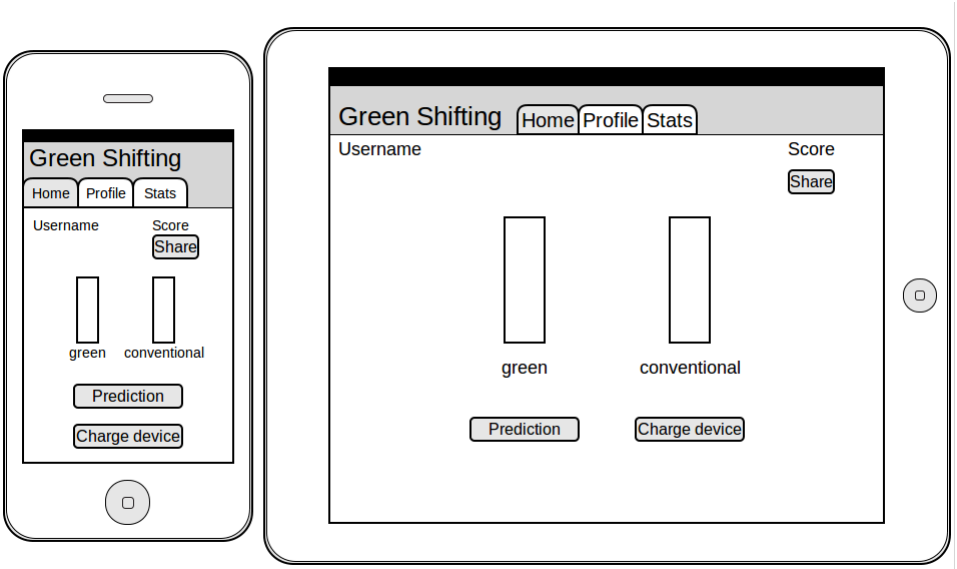


Figure 4.2: Mockup of the home screen GUI for two screen resolutions

be extended using the REST API. User authentication can be done via an included service using only client-side code. Besides authentication with email and password, also login with the social providers Facebook, GitHub, Twitter or Google is supported. Each Firebase interaction is transmitted using SSL encryption. It is also possible to host a static website using only HTML, CSS and JavaScript. The database model follows a tree structure stored as JSON objects. This means that keys are stored as parental nodes to its values, whereby a value again can be a parent for other values. For example, in listing 4.1 ”greenshifting” is the main key with the values ”profiles”, ”providers” and ”reminders”. The value ”profile” is parent to more values and so on.
Since Firebase offers not only a realtime database solution, but also web hosting and is free for small projects, it is used in this thesis. The system needs three different databases which are realized by three subtrees. Listing 4.1 shows the JSON structure of the databases. Lines 3 to 25 show the profiles database with the different fields, like username, devices and reminder. To be able to analyse the way the user gathers points the score is differentiated into the categories of which it is composed. In line 27 to 41 the energy data is saved, grouped by provider. Each provider has an ID and a name. The energy data is stored per date and hour and differentiated by type. A precalculated percentage of green energy is also included. Line 43 to 47 shows the database about the reminders. Since the list for each time is cleared by the server after invoking the notifications only the time needs to be stored, while the date is redundant. Notifications are stored

by the device's registration ID and its platform, so the server can select the correct messaging service.

Listing 4.1: Database structure in JSON syntax

```
{
"greenshifting": {
    "profiles": {
        "email@host:de": {
            "devices": {
                "regID" : {
                    "name": "name",
                    "type": "Platform"
                }
            },
            "interval": 300000,
            "lastReminder": 0,
            "lastUsage": 0,
            "provider": 1,
            "reminder": "",
            "scores": {
                "device": 0,
                "mobile": 0,
                "reminder": 0,
                "social": 0
            },
            "totalscore": 0,
            "usedMobile": false,
            "username": "name"
        }
    },

    "providers": {
        "1": {
            "provider": "EEX",
            "data": {
                "[date]": {
                    "[time]": {
                        "all": 0,
                        "greenPercentage": 0,
                        "solar": 0,
                        "wind": 0
                    }
                }
            }
        }
    },

    "reminders": {
        "time": {
            "regID": "platform"
        }
    }
}
```

4.2.2 Tool selection

For the development of the system several tools are needed. In the following, tools for cross-platform development (CPD) and responsive web development (RWD) are compared and one tool for each purpose is selected.

Adobe PhoneGap

One of the most important non-functional requirements is the accessibility of the system. Cross-platform development makes it possible to make the system available for many platforms with less programming effort as by developing the system for each platform specifically. In section 2.2.1 CPD is introduced and three tools (PhoneGap, Titanium and RhoMobile) are presented. This system makes use of PhoneGap since it supports the most platforms and is highly extensible. It is also the most common tool and thus provides many plug-ins for native functionality, like information about the battery status, geolocation, camera, contacts, network information and so on. Plug-ins are not only provided by PhoneGap itself, there also exist several third-party plug-ins. Another advantage is its detailed documentation. The biggest drawback is that for the installation on the different platforms the platform specific SDK is needed, which are not always available for all operating systems (OS). For example, platforms based on Windows need a Windows OS to be build, while software for the iPhone needs iOS. Others, like Android SDK, are available on all systems. Table 4.2 gives an overview on the platforms and the supported OS.

	Windows	iOS	Linux
Android	✓	✓	✓
iOS		✓	
WindowsPhone 7	✓		
WindowsPhone 8	✓		
Windows 8	✓		
BlackBerry 10	✓	✓	
FireOS	✓	✓	✓
Firefox OS	✓	✓	✓
Ubuntu			✓(Ubuntu)

Table 4.2: Overview on the different platforms supported by PhoneGap and the needed operating system for their SDK

Twitter Bootstrap

App development with PhoneGap is based on JavaScript, HTML and CSS. Not all devices have the same resolution, so RWD becomes important to maintain the usability of the system. Section 2.2.2 proposes three RWD tools (i.e. Twitter Bootstrap, ZURB Foundation and Skeleton). The most common and best documented tool is Twitter Bootstrap, which is the reason why it is used for designing the GUI of the Green Shifting system. With Bootstrap is it also possible to create a web application with reduced functionality, which can be used inside browsers, by just adapting the existing code. This makes the application also available from desktop computers and laptops.
Together with Adobe PhoneGap, Twitter Bootstrap ensures the availability of the system on as many platforms as possible.

Chapter 5

Implementation

This chapter describes the implementation of the functional requirements and contains a first view on the resulting prototype of Green Shifting. The system consists of two active components, the server that processes the energy data and manages the notifications and the mobile application which acts as an interface to the user.

5.1 Server structure

The server is a virtual machine with Ubuntu 14.04 installed and is used for two different tasks: processing the energy data and sending notifications. In the following, both tasks are described in more details.

5.1.1 Data processing

The main task of the server is to gather the energy data from the providers and process it in a way that it matches the Firebase data structure. During the work on this thesis, no current provider data was available, so the system makes use of some archived energy data of European Energy Exchange (EEX) [Eur12]. EEX is a leading energy market place in Europe where amongst other things energy, coal and emission allowances are traded. To increase the transparency on the wholesale market, EEX provides a transparency platform, where market-relevant generation data is published. In matters of energy production, they provide free example transparency data from Germany/Austria and the Czech Republic. This data comes in the CSV format and contains information about the type of energy, the expected and actual generated amount of energy (total), timestamp and lo-

cation. All in all, there is data for two days that can be evaluated by this app as example data. For this, the values for produced solar and wind energy are compared to the value of generated energy at each timestamp. This is also done for the prediction data. Listing 5.1 shows an example CSV file from the EEX. The first lines are comments and tell something about the structure of the file. The first uncommented line contains a timestamp of the creation of the file and always starts with the key "FCRT". Then the actual data is recorded. The only interesting values here are the "TimeStamp" and "PlannedGeneration" entries, where the later one varies from file to file, depending on the energy type, and if it is planned or produced data. The last line starts always with the key "TELI" and contains the number of lines in the file. The data extracted from the lines between "FCRT" and "TELI" is converted to the JSON schema known from the listing in 4.1, line 28 to 42, and then uploaded to the Firebase database.

Listing 5.1: Example CSV file structure

```
1  # ExAnteInformationPlannedGeneration
2  #
3  # FCRT;[CreationTimeStamp]
4  # CPGL;[Country];[TimeStamp];[PlannedGeneration];[PublicationTimeStamp];[
       ModificationTimeStamp]
5  # TELI;[LineNumbers]
6  #
7
8  FCRT;2012-05-10T18:05:00+02:00
9  CPGL;DE;2012-05-11T00:00:00+02:00;7506.8;2012-05-10T18:01:48+02:00;
       2012-05-10T18:01:38+02:00
10 CPGL;DE;2012-05-11T01:00:00+02:00;7135.5;2012-05-10T18:01:48+02:00;
       2012-05-10T18:01:38+02:00
11 CPGL;DE;2012-05-11T02:00:00+02:00;7101.0;2012-05-10T18:01:48+02:00;
       2012-05-10T18:01:38+02:00
12 # [...]
13 CPGL;DE;2012-05-11T23:00:00+02:00;7489.5;2012-05-10T18:01:48+02:00;
       2012-05-10T18:01:38+02:00
14 TELI;32
```

5.1.2 Notification Management

Since the mobile application runs mostly using JavaScript, it is not able to run in the background. That implies, as soon as a functionality needs to be available when the app is not in foreground, native code is needed which increases the programming effort for each additional platform that is supported by the application. For several of these functions PhoneGap provides plug-ins. To send push notifications to a device, a developer has two possibilities: Either to implement the push notification natively for each platform, or to set up a server that initiates push notifications by calling the platform specific notification services (see table 4.1). The existing plug-in for PhoneGap for this feature uses the later implementation. To save

time, this plug-in is also used in the Green Shifting system. This means, there needs to be a server that initiates the messages sent by the notification systems. Listing 5.2 shows a Python script that sends these messages to Android and iOS devices. It uses the Firebase Python library by Michael Huynh [Huy13]. It runs once per hour and retrieves each time the current registered notifications from the Firebase (see listing 4.1 line 44ff) which are deleted afterwards. The registration ID is used to identify the device.
The procedure with Android and GCM can be seen in line 21f and has the following mechanism. A GCM object is created with a project ID as parameter. This project is used to send push notifications to all registration IDs from the database. iOS and APNS work similar, but instead of using a project ID, a certificate and a key file are used (see the lines 25 to 28).

Listing 5.2: Push Notification Management

```
1  # get current time
2  now = datetime.datetime.now()
3  hour = now.hour
4
5  # get reminder entries
6  ref = Firebase("https://greenshifting.firebaseio.com/")
7  reminders = ref.child("/reminders/%i:00" %hour).get()
8
9  # message to be sent
10 data = {"message": "Your Green Shifting Reminder!", "title": "
       GreenShifting"}
11 reg_ids_android = []
12 reg_ids_ios = []
13 # get device IDs for reminder
14 for entry in reminders:
15     if reminders[entry] == "Android":
16         reg_ids_android.append(entry)
17     elif reminders[entry] == "iOS":
18         reg_ids_ios.append(entry)
19
20 #send Android push notifications
21 gcm = GCM("AIzaSyDPEX921YShmMoao7a27lbAczkn2hiUKA0")
22 gcm.json_request(registration_ids=reg_ids_android, data=data)
23
24 #send iOS push notifications
25 apns = APNs(use_sandbox=True, cert_file="cert.pem", key_file="key.pem")
26 payload = Payload(alert:"Your Green Shifting Reminder!", sound="default",
       badge=1)
27 for reg in reg_ids_ios:
28     apns.gateway_server.send_notification(reg, payload)
29
30 # delete entries from database
31 ref.child("/reminders/%i:00" %hour).remove()
```

5.2 Application structure

The mobile application defines the user interface and is the heart of the system. Its functionality is implemented using JavaScript, while the GUI is designed with Twitter Bootstrap in HTML and CSS.

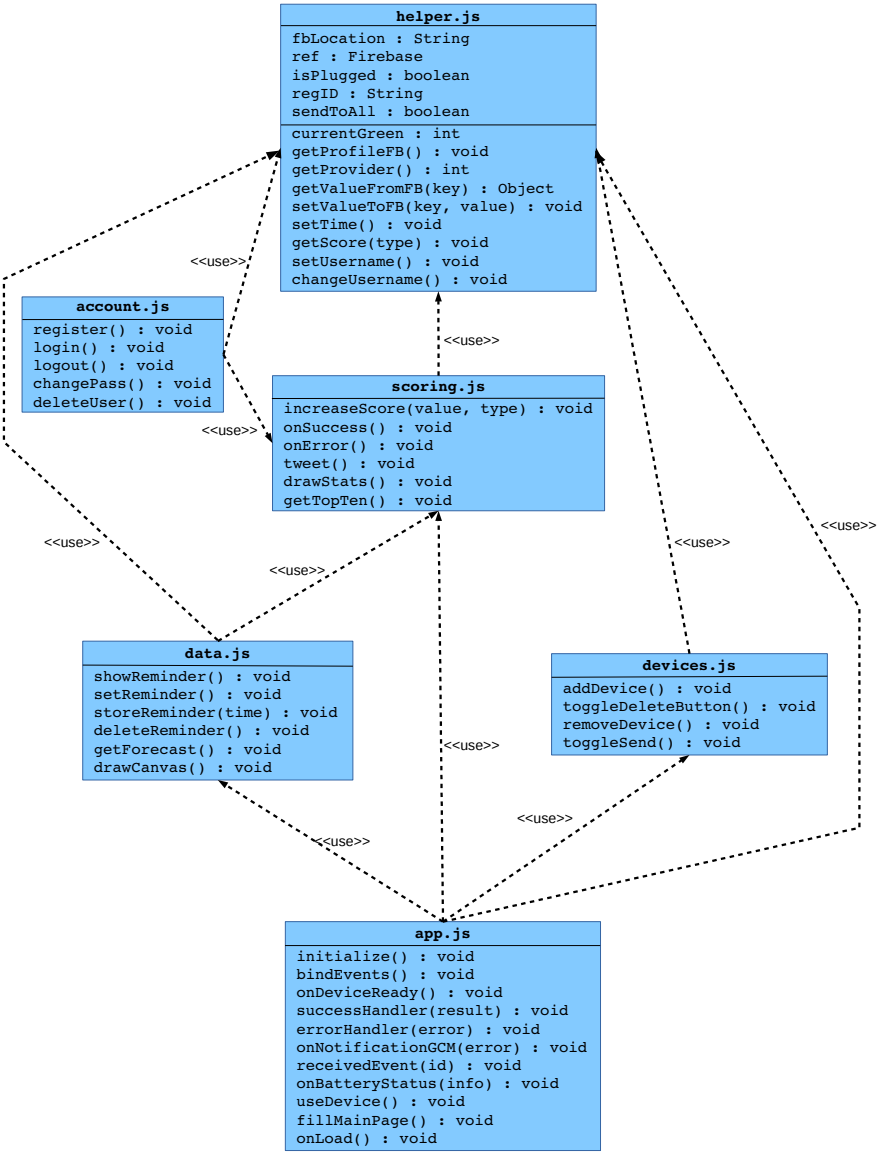


Figure 5.1: Interaction of the different components

5.2.1 Backend

The backend of the application consists of five components which are distinguished by their functionality. Figure 5.1 shows how these components interact and which functions they provide.

Helper

The helper class contains basic functions which are often called by the others. It also defines some global variables which are listed in the diagram 5.1.
The method *getProfileFB()* returns the Firebase address of the profile for the currently authenticated user. With the methods *getValueFromFB(key)* and *setValueToFB(key, value)* profile specific values are read from and written to the database. See the listings 5.3 and 5.4 to see how a query is done with Firebase. *getProvider()* returns the selected provider of a user, or a default provider if no user is authenticated. It defines the energy provider from which the energy data is queried from.
A user can gather points by using different functionalities of the system. To evaluate which functionality is used most, the score is stored for each type. The method *getScore(type)* returns the score that is specified by the *type* parameter.

Listing 5.3: Storing data in Firebase

```
1  setValueToFB : function(key, value) {
2    var fb = new Firebase(this.getProfileFB());
3    entry = {};
4    entry[key] = value;
5    try {
6      fb.update(entry);
7    } catch (e) {
8      alert(e);
9    }
10 },
```

Listing 5.4: Retrieving data from Firebase

```
1  getValueFromFB : function(key) {
2    var userRef = new Firebase(this.getProfileFB());
3    var value = null;
4    try {
5      userRef.child(key).on('value', function(snap) {
6        value = snap.val();
7      });
8    } catch (e) {
9      alert(e);
10   }
11   return value;
12 },
```

Scoring

The class *scoring.js* contains methods relevant for gaining and evaluating the user scores. *increaseScore(value, type)* is used to increase the score for a specified type by a given value. Depending on the action, the score a user gains differs. Table 5.1 gives an overview on the actions that result in an increasing of the score, if some requirements are fulfilled.
If a user wants to publish his score, he can share it on social networks. For this, a third party Phonegap plug-in by Eddy Verbruggen [Edd15] is included. This plug-in accesses the accounts of a device to share some content. This means, the user also needs other applications for Twitter, Facebook and so on installed on his device. At the moment, only publishing on Twitter is implemented by the Green Shifting system, but it is easily extensible to other services. The snippet 5.5 shows how this feature is implemented, while the sequence diagram in figure 5.2 visualises the different events in chronological order. The callback function *onSuccess()* increases the score by three points, but only if it was able to find a Twitter account on the device. Tweeting the score is limited to one tweet per day to avoid spamming and unlimited score gain.

Listing 5.5: Using the social media plug-in

```
1            window.plugins.socialsharing.shareViaTwitter(textToTweet, null
                 , null, this.onSuccess, this.onError);
```

Firebase supports the ordering of values during a query. Listing 5.6 shows how this feature is implemented in the method *getTopTen()* to retrieve an ordered list of the users with the highest total score.
In the *drawStats()* method, the composition of the user score is visualised.

Listing 5.6: Firebase query with ordered result

```
1      ref.child('profiles').orderByChild('totalscore').limitToLast(10)
2        .on('child_added', function(snap) {
3            //...
4          });
```

Account

Firebase provides several authentication methods. Besides using email and password, authentications with the user's Facebook, Twitter, GitHub or Google+ account are also possible. For Green Shifting only the email and password method is used so far. Additional user data cannot be stored in the same list as the authentication information, so a second entry for the profile data is created, with the email address as common key. Since a key

Action	Preconditions	Score
using the mobile app	first login	10
recharging the device	high green energy level, device plugged in, one usage per hour	5
setting a reminder	reminder only for high green energy level, one reminder per hour	5
tweeting the score	Twitter account on device installed, one tweet per day	3

Table 5.1: User actions and their resulting scores

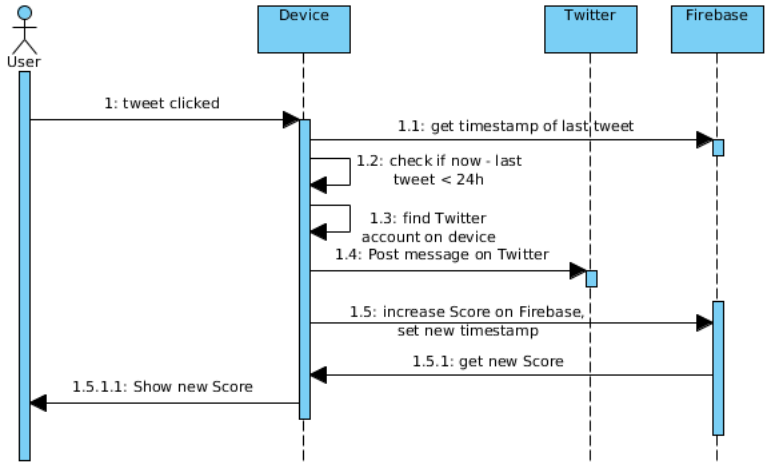


Figure 5.2: Sequence diagram of the tweet action

cannot contain the signs ".", "{", "}", "[" and "]", the dots from the email address are replaced by a ":".
The class *account.js* contains methods that register an account, deal with the logging in and out processes, enable a user to change his password and to delete an account if its no longer needed.

Data

The *data.js* class deals with the energy data. It shows the current data (*drawChart()*), collects the energy prediction information (*getForecast()*) and enables a user to set a reminder. Listing 5.7 shows the process from getting the selected reminder time to storing the reminder in the reminder database. In line 2 the selected time is retrieved. The lines 12 to 16 check

if the device is already registered. Otherwise, a dialog is shown that asks the user to register the device (line 18). Then the function checks if the previous time the reminder was set was more than an hour ago. Since a user gains points by using this feature, this restriction is implemented to avoid the unreasonable increase of the score by abusing this functionality. It is only enabled on Android and iOS devices because push notifications are only implemented for these platforms so far.

Listing 5.7: Setting a reminder

```
1  setReminder : function() {
2    var time = $("input[name='forecastRadios']:checked").val();
3    if (time == null) {
4      alert('You have to select a time first!');
5      return;
6    }
7    last = -1;
8    var now = new Date().getTime();
9    var fb = new Firebase(helper.getProfileFB());
10   last = helper.getValueFromFB('lastReminder');
11   found = false;
12   devFB = fb.child("devices/" + regID);
13   devFB.once('value', function(snap) {
14   if (snap.val() !== null)
15     found = true;
16   });
17   if (!found) {
18   var r = confirm("You need to register your device first. Do you want
         to register it now?");
19   if(r){
20     devices.addDevice();
21   }
22   else
23       return;
24   }
25   // only one Reminder per hour
26   if (last == 0 || last == null || now - last > 3600 * 1000) {
27   scoring.increaseScore(5, 'reminder');
28   fb.update({
29     lastReminder : now,
30     reminder : time
31   });
32   fb.off();
33   this.storeReminder(time);
34   } else {
35   alert('You can only set one reminder per hour.');
36   }
37   $('#forecastModal').modal('hide');
38 },
```

Devices

To set a reminder the registration ID of a device is needed, otherwise no push notifications can be sent. The *devices.js* class enables a user to register his device's ID in the database and thus enable the reminder feature. If he

uses several devices, an entry for each device is created. He can also remove the currently used device if he does no longer want to receive notifications. Listing 5.8 shows how an entry is deleted from Firebase.

Listing 5.8: Deleting a device ID

```
1 new Firebase(helper.getProfileFB() + "/devices/" + regID).remove();
```

App

The *app.js* class defines on the one hand methods for the receiving of push notifications (*onNotificationGCM()*, *receivedEvent()*), but also for handling the GUI which depends on the authentication state (*fillMainPage()*). Another important functionality implemented in this class is the *useDevice()* method that can the user can invoke by recharging the device when the green energy level is good. Figure 5.3 shows an activity diagram that visualises the process of this function from the pushing of the button to the gaining of points.

5.2.2 Frontend

The HTML file that defines the GUI consists of six different parts: The navigation bar, the four views and the included libraries. Listing 5.9 contains the definition of the navigation bar. Line 1 shows the Twitter Bootstrap parameters for the navigation bar design. ”navbar-fixed-top” means the navigation bar is on the top of the page and stays visible during scrolling. ”navbar-default” defines the color style to be default. The default Bootstrap design is adapted to a custom design and stored in a CSS file. In the lines 17, 21, and 25 so called ”Glyphicons” are defined. These are icons that behave like font and come with the Bootstrap stylesheet. The used navigation style are tabs.

Listing 5.9: Definition of the navigation bar

```
1  <nav class="navbar navbar-default navbar-fixed-top"
2      id="top" role="banner">
3      <div class="container">
4          <div class="navbar-header">
5              <h4 style="color: white; font-weight:bolder">
6                  <img src="img/icon.png" height=30 width=30>
7                  Green Shifting</h4>
8          </div>
9          <nav role="navigation">
10             <ul class="nav nav-tabs" role="tablist" id="myTabs">
11                 <li id="homeTab" class="active" >
12                     <a href="#home" data-toggle="tab" role="tab">
13                         <span class="glyphicon glyphicon-home"></span>
```

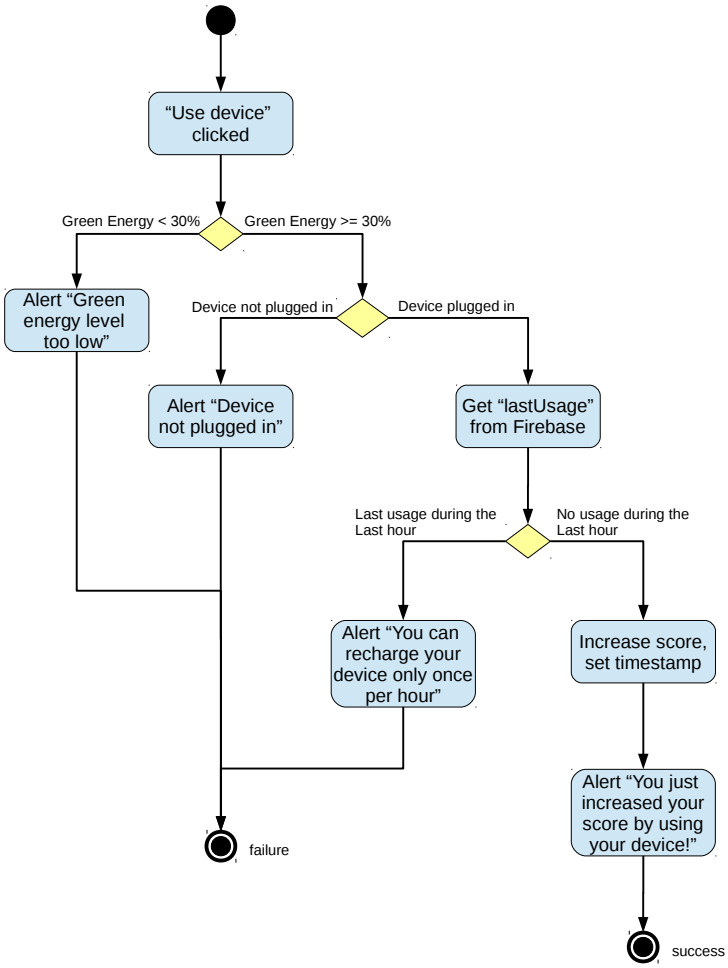


Figure 5.3: The use device activity

```
14                  </a></li>
15              <li id="profileTab" style="display: none">
16                  <a href="#profile" data-toggle="tab" role="tab">
17                      <span class="glyphicon glyphicon-user"></span>
18                  </a></li>
19              <li id="statTab">
20                  <a href="#statistics" data-toggle="tab" role="tab">
21                      <span class="glyphicon glyphicon-stats"></span>
22                  </a></li>
23              <li id="aboutTab">
24                  <a href="#about" data-toggle="tab" role="tab">
25                      <span class="glyphicon glyphicon-question-sign"></
                            span>
26                  </a></li>
27          </ul>
```

```
28          </nav>
29      </div>
30  </nav>
```

Then the content of the three tabs is defined. The main view shows a pie chart of the energy composition (green and conventional) and the energy prediction for the next five hours. For the chart the JavaScript library Chart.js [Dow] is used. In listing 5.10 taken from the *scoring.js* the usage of this library is shown.

Listing 5.10: Creation of a pie chart

```
1  data = [{value: green, color: "#090", highlight:"#0c0", label:"green"},
2          {value: conv, color:"#000", highlight:"#666", label:"conventional"
               }];
3
4  context = $('#pieEnergy').get(0).getContext('2d');
5  Chart.defaults.global.animation = false;
6  var pie = new Chart(context).Pie(data,{});
```

The login and a register buttons above the chart open a modal where a user can enter his login/registration data. If a user is authenticated, a row with the username, his score and a button for publishing the score is displayed above the chart instead of the login and registration buttons. He is then also able to see details about the energy prediction where he can set a reminder and push a button if he is charging the device at that moment.
The other tabs are only displayed if the user is authenticated. On the profile tab a user can set specific preferences, like selecting his energy provider, registering and unregistering his device for notifications, changing the displayed username and his password and deleting his account. The statistics tab shows the composition of the user's score as a staple bar chart and a list of the ten users with the highest score. In listing 5.11 the Twitter Bootstrap grid layout is used to place the staple bar chart and its legend next to each other. For this they are defined to be in the same row. Then the width of both elements is defined with the *col-xs-4* statement. In Bootstrap a page is divided in 12 columns. The statements of the format *col-AB-X* is interpreted as follows: the *X* stands for the number of columns taken by the element, while with *AB* the resolution is defined. Possible values for *AB* are "xs" (smartphones), "sm" (tablets), "md" (laptops) and "lg" (desktop monitors). The column width depends on the resolution. Here the application is optimized for the view with smartphones.

Listing 5.11: Grid layout example

```
1  <div class="row">
2      <div class="col-xs-4 text-center">
3          <canvas id="scoreChart" width="40" height="100"></canvas>
4      </div>
5      <div class="col-xs-8">
```

```
6           <div style="color: #0c0">Posting the score on Twitter</div>
7           <div style="color: #090">Setting a reminder</div>
8           <div style="color: #060">Using the mobile app</div>
9           <div style="color: #030">Charging the device</div>
10      </div>
11  </div>
12                  <hr />
```

In the end of the HTML file the JavaScript files are imported. They are put to the end of the file to enable a faster loading of the page.

5.3 Firebase configuration

Firebase is configured using the web interface. For Green Shifting email and password authentication is enabled. To provide also third-party authentication, the domain must be whitelisted for OAuth redirects. This means to register a new application for each of these services. This is not done for Green Shifting yet but might be an option for the future.
Firebase also provides the possibility to secure the database. This is done by defining read and write rules. In listing 5.12 the current rules for Green Shifting can be seen. For example the ”write” access for everybody on the users and profile entries are not the best solutions. They are at the moment implemented that way because the app registers an user and thus writes to these databases while no authentication is given. A workaround for this problems is to authenticate the application with a default account as soon as it starts. This is not yet implemented.

Listing 5.12: Firebase rule set

```
1  {
2      "rules": {
3          "profiles": {
4            ".read": true,
5            ".write": true,
6            ".indexOn": ["totalscore"]
7          },
8          "providers": {
9            ".read": true,
10           ".write": "auth != null"
11         },
12         "reminders": {
13           ".read": true,
14           ".write": "auth != null"
15         },
16         "users": {
17           ".write": true,
18           "$uid": {
19             ".read": "auth != null && auth.uid == $uid"
20           }
21         }
22     }
23 }
```

5.4 Prototype

The application is devided into four pages: the starting page (main view), profile, statistics and about tab. They can be reached by using the symbols on the navigation bar on the top. The following subsections navigate through these tabs and their options.

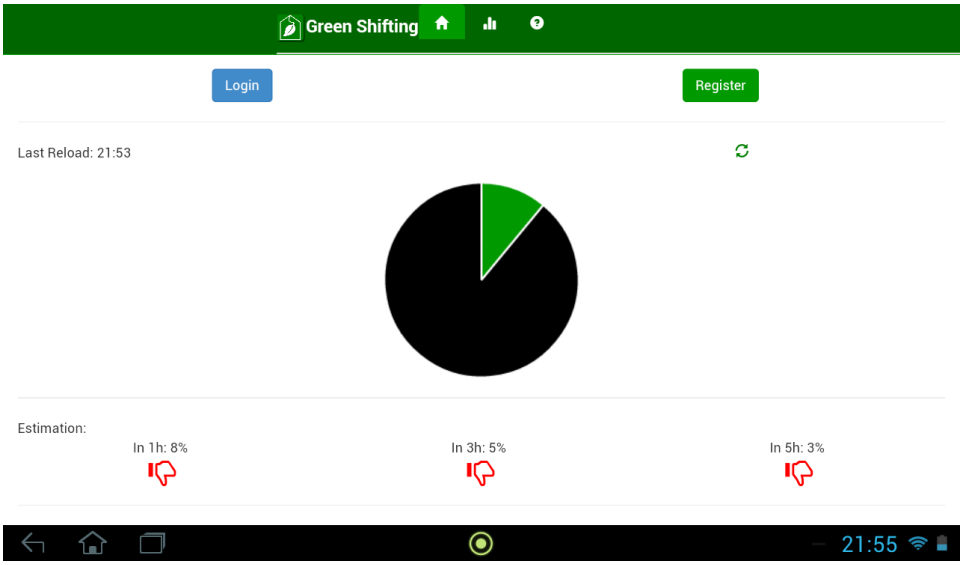
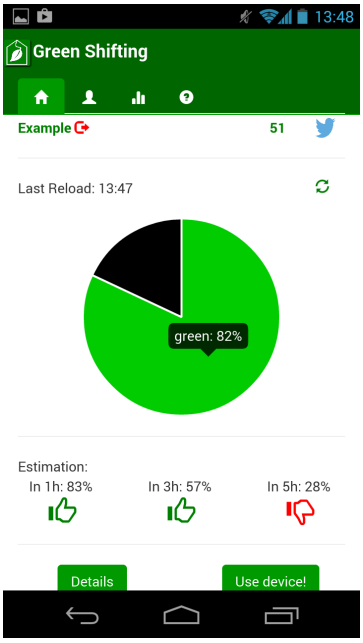


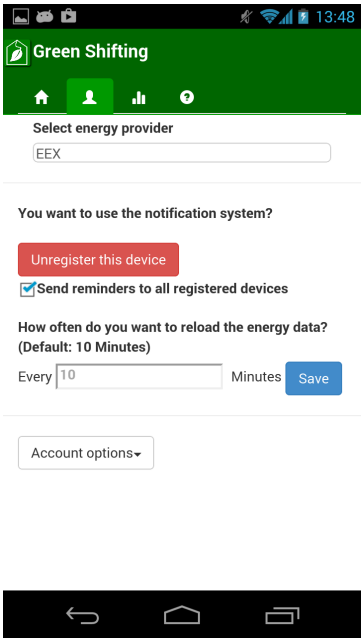
Figure 5.4: The first impression on a tablet

5.4.1 Main tab

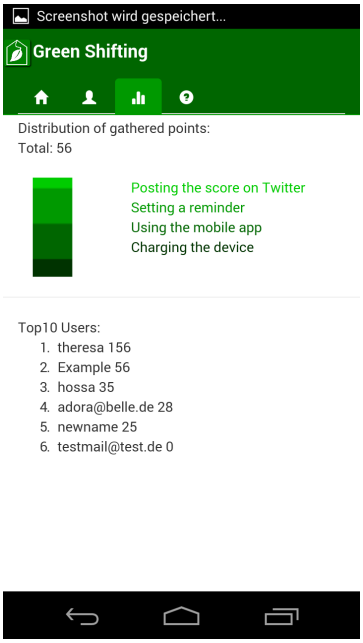
The first impression matters, so the main page is the most important part of the application. Figure 5.4 shows what a user sees first in Green Shifting on a tablet, while in figure 5.5 the views from a smartphone are shown. Further screenshots of almost all actions can be found in the appendix A. The most important information - the energy data - is shown to everyone, independent of the existence of an account. As provider a default one is selected, at the moment this is EEX, since there is no other data available. By tapping on the chart area (see figure 5.5a), a user can obtain detailed information about the percentage. The timestamp shows when the page was refreshed for the last time. To use more functionality, a user has to register an account and then sign in.
After authentication, the user can see two buttons, one for the details, and one for using the device. Pushing the details button opens a dialog with prediction details for the next 12 hours. If the renewable energy level is



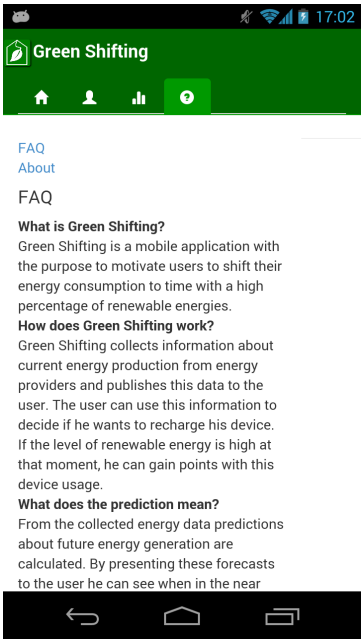
(a) Main view



(b) Profile view



(c) Statistics view



(d) About view

Figure 5.5: The different views of the application

higher than 30%, a time can be selected and a reminder can be set. If the user has set a reminder before, it is also shown and it can be deleted.

5.4.2 Profile Tab

Figure 5.5b shows what an authenticated user sees after he tapped on the profile tab. As first option he can select an energy provider. At the moment only EEX is selectable, but as soon as data by more providers is available, they can be selected here.
If a user wants to use the notification system to receive reminders with this device, he can register it here. A button to unregister this device is shown afterwards. If a user owns several devices, he can register each and then decide if he wants the reminders on all of them or only on the one from which he set it.
With the next field, the user can define the update interval for the data. The default interval is 10 minutes.
The last entry on this tab is a drop down menu. Here the user can select account preferences, like changing the username and the password, logging off or deleting the account.

5.4.3 Statistics Tab

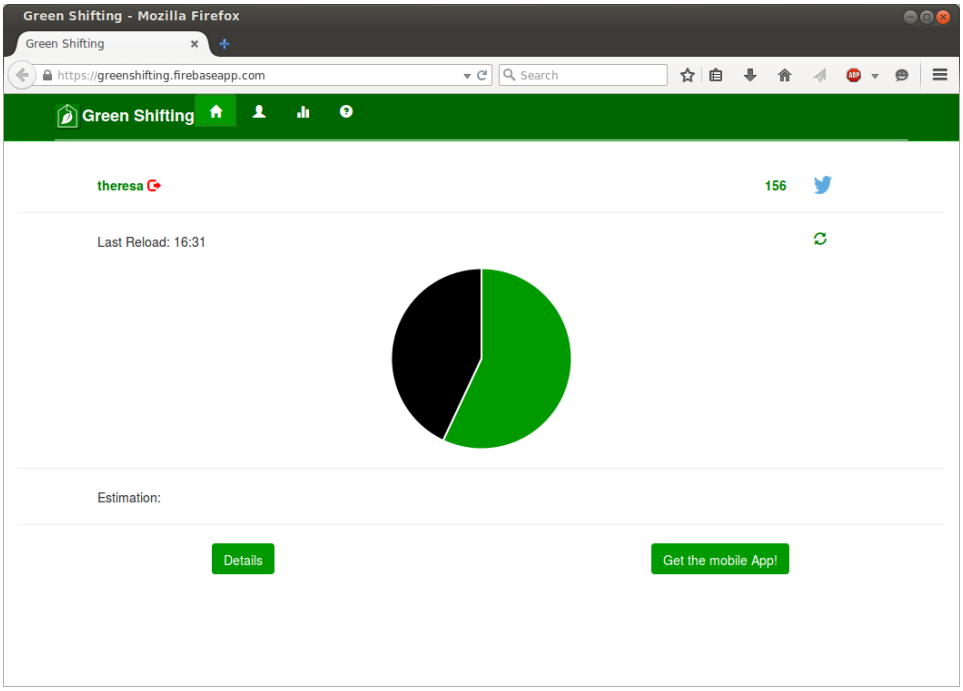
This tab is already shown in the unauthenticated mode with reduced content. If a user is not logged in only the top ten list is displayed. Otherwise, he can also see a staple bar chart that shows the composition of his gathered points (see figure 5.5c).
The staple bar chart can be interpreted as how the app is used. If the user often publishes tweets, the score for this usage increases and thus its part in the total score. The chart can motivate a user to try to balance the score by using the device more often or setting more reminders. The top ten list on the other hand increases the competition between users which also increases the usage of the app. This greatly fulfils the system's goal: to increase the renewable energy awareness.

5.4.4 About Tab

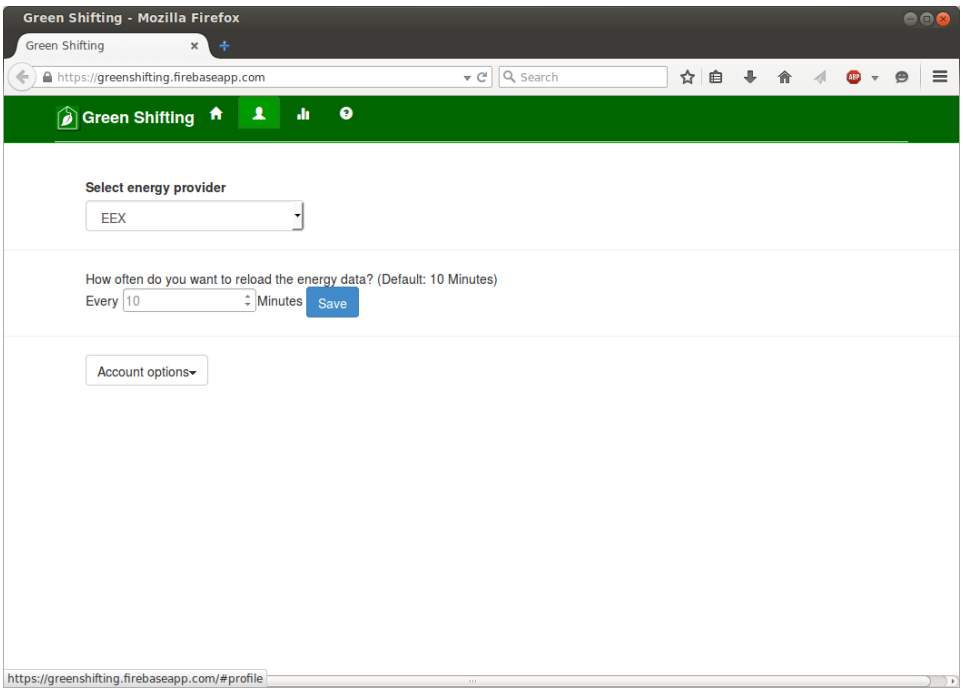
Behind the question mark some information about the application and its developers can be found (see figure 5.5d. A frequently asked questions (FAQ) section is also contained.

5.5 The Web application version

Twitter Bootstrap enables the system to be easily migrated between different platforms without heavy losses regarding the design. Since the mobile application uses some device sensors and features, its functionalities need to be slightly reduced for the adaption as web application. This implies that the setting of a reminder is not possible as well as the device usage, since a web application cannot gain any information about the battery status of the device it is running on. Since the main task of the web application is to increase the accessibility of the energy information data, this goal is reached. Figure 5.6a shows a screenshot of the main view on the web application after a user is logged in. Via the button "Get the mobile App!" the mobile application can be downloaded. Compared to the view on the mobile app in figure 5.5a this button in the only difference. Figure 5.6b contains the profile view of the web application. The reduced functionality is easily seen compared to the screenshot in figure 5.5b. The web application is hosted with Firebase and can be reached from *https://greenshifting.firebaseapp.com*.



(a) Main view



(b) Profile view

Figure 5.6: Screenshots of the web application

Chapter 6

Evaluation

During the development of software it is important to ensure its quality. This is achieved with testing. The quality of the code consists of its functionality and is guaranteed by unit testing, while for the GUI it is important to be usable. This chapter describes the application of both types of tests. Table 6.1 lists the platforms on which the system has been executed.

Type of test	Platform	Versions
Unit testing	Ubuntu Firefox	35.0.1
Execution web app	Ubuntu Firefox	35.0.1
	Android Browser	4.3-776638
		4.1.1-1351049447
Usability mobile app	Android	4.1.1
		4.1.2
		4.2.2
	Android CyanogenMod	4.3

Table 6.1: Overview on the tests and on which devices they were executed

6.1 Unit testing

The unit testing of the JavaScript source code was done using the QUnit framework [jQu15]. All self implemented functions were tested if they return the correct result. Listing 6.1 shows the example test setup for the *scoring.increaseScore()* method. In line 2 the correct handling of a wrong parameter is checked. Line 3 tests, if the code execution was successfully and if the method returned the correct value 0. The last test makes sure that the database entry ”account” at ”totalscore” was updated correctly.

Listing 6.1: QUnit test of scoring.increaseScore()

```
1  QUnit.test("test increaseScore", function(assert) {
2    assert.ok(increaseScore(23, "wrongtype") == -1, "Wrong type considered")
         ;
3    assert.ok(increaseScore(13, "social") == 0,
4      "social Score successfully increased");
5    assert.ok(account["totalscore"] == 123, "score correctly added");
6  });
```

6.2 Usability tests

The usability test was done with five testing persons. The number seems low, but in [NL93] Nielsen and Landauer show that already with five persons about 84% of all usability problems can be found. The formula to gain this value is $1-(1-L)^n$, where n is the total number of test users and L is the fraction of usability problems discovered while testing a single user. L is typically set to 31%.

6.2.1 Test setup

The test persons were 24 to 31 year old Germans who are somewhat active mobile device users. They first had to answer some questions about their energy consumption and mobile device usage. Then they had to complete some typical tasks with the prototype. Afterwards, they were asked for their opinion about the system. In the end, the technical details about the used device were queried. The full questionnaire including the tasks to fulfil can be found in appendix B.

6.2.2 Usability evaluation

For all test users sustainability is somewhat important (value from the questionnaire: 4). Their opinion about renewable energies is positive but most of them think their usage needs to be improved. They think they are now often too inefficient, further research need to be done. New innovative method shall be considered, too. Table 6.2 shows the answers in detail.
Most of the users (60%) are interested in data about the current energy generation. All of them try to save energy. They are all active users of mobile devices (average value: 7.8) and have between 10 to 20 apps installed themselves. Four of the users would shift the usage of time-flexible energy consuming gear to times with a high level of green energy. They also would be interested in an application that shows the current energy composition.
The evaluation of the application had the following results. The test users

User	Opinion
User 1	If it makes sense they should be extended. Especially small, decentralised solutions and new technologies such as wave power plants.
User 2	Renewable energies are important. However, the conversion should take place carefully and must not be rushed. It is important to get away from resources that are limited and used for other purposes as well, and instead use renewable energies. Also, the ecological damage needs to be taken into account.
User 3	Important innovations, but too inefficient. More research needed before further mass production.
User 4	Need to be extended with more storing options.
User 5	Good thing. Need to be paid attention to more often.

Table 6.2: User opinions about renewable energies

thought the application to be rather intuitive (3.8 of 5). Two users reported some smaller problems: The app needs to refreshed after start to load all information. After changing the username the app needs to be refreshed to show the new username in the leaderboard. They suggested several improvements, like better design, better structure or realistic data. Table 6.3 lists the complete answers of the users. The users were all using Android smartphones.
In summary, most of the users liked the concept of the app, but they think it needs further improvement. Their suggestions have to be considered when extending the application.

User	Opinion
User 1	Nicer dialog for setting the reminder. A timeline of 24h with pie charts: how is the energy composition at what time? Would be nicer than the current list. Setting the reminder not for a specific time but for a percentage of green energy. The app should recharge the mobile device automatically if the electricity is cheap or the level of green energy is high.
User 2	Recognize automatically when device is recharging and green energy level is high. Automatic notification for good energy level instead of reminder. Usage with other devices, especially large equipment, if possible, or SmartMeter.
User 3	Design needs to be more appealing so it is more fun to use the app.
User 4	Direct login after registration. Setting a reminder directly after registration in reminder menu. App needs real data with a higher time resolution.
User 5	After deleting a reminder, no new can be set immediately (only after one hour). Bad luck in the case of a wrong click. The visualisation of the gathered points is hard to read. A pie chart would be better readable. The different shades of green do not make it any easier. For these reasons the deciphering of the view itself was not easy, too.

Table 6.3: User suggestions for improving the mobile application

Chapter 7

Discussion

7.1 Limitations of the system

During the development already some limitations of the functionality of the system appeared. The usability test revealed some further improvements that need to be considered.

7.1.1 Energy data limitations

The main problem with the application is that it does not provide current energy data, but only example data. It is important to contact energy suppliers to convince them to make their data about energy production available. The biggest challenge then will be to process the data from the different suppliers, since they will most likely not be provided in the same format.
A user might also be interested in seeing details about the energy, such as the amount of the different resources that are part of the renewable energy. At the moment, only the total green energy is compared to the total conventional energy.

7.1.2 Application limitations

At the moment the system does not provide a lot of functionality. A user is only able to gain points by using the device the app is running on during times with a high renewable energy level, while mobile devices do not consume as much energy as large electronic gear like washing machines. The challenge here is the connection of the Green Shifting system to other

devices. Probably in the future when there are more smart devices available in households this will be realised. Also a connection to smart energy meters that measure the total energy consumption of a household is thinkable, as it was done by Mattern, Staake and Weiss in [MSW10].
Another problem is that the system is running with JavaScript and thus not able to run in the background when the app is closed. Otherwise, the system could notice itself when a device is plugged in and could increase the score if the green energy level is high. This kind of functionality needs the usage of native code and raises the implementation effort for each platform that is supported.
At the moment, the application is only built and tested for Android devices. Some functionalities like the sending and receiving of push notifications need to be added for other platforms.

7.2 Future Work

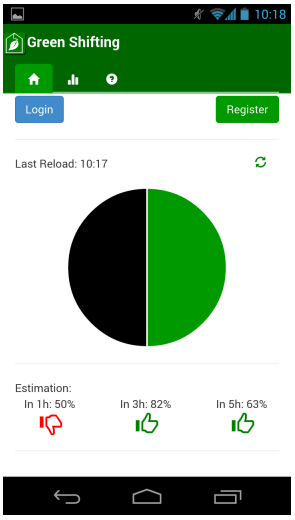
It can be seen from the previous section that the system needs some enhancements. For future development especially the exploitation of current energy data is important to provide a more up to date application. This also contains further work in the data processing and visualisation.
The application itself can be improved in several ways. Since Firebase provides authorization not only by email and password, but also by existing accounts for several social networks like Facebook, Twitter, GitHub and Google+, this feature can be supported. Also, the sharing of the score can be extended from just Twitter posts to all accounts that are registered on a device.
These are some smaller enhancements, but also more complex ones can be imagined. As already mentioned, it is possible to connect smart energy meters to an application. This data could be used to reward a user if the main energy consumption lies during times of a high renewable energy level. Also, as soon as it is possible to communicate with the electronic devices themselves it is thinkable to control them with the app and shift their usage to times of good energy levels.
Another rather complex enhancement affects the predicition of future energy generation. It is at the moment taken from EEX with actually provides prediction data. Still, it might be also interesting to calculate own forecasts based on previous energy data and weather forecasts. Other energy suppliers might also not provide their predictions which makes a custom calculation necessary.

7.3 Summary

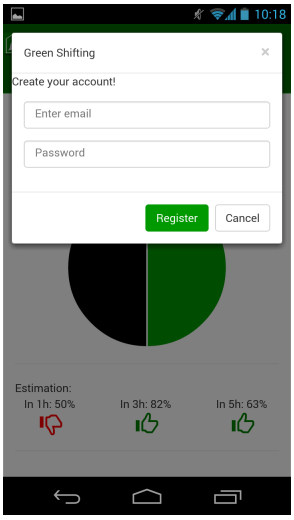
Renewable energies have an increasing part in the electricity generation of Europe. Also, many people become more and more aware of sustainability. They a interested in using renewable energies rather than conventional ones. Since mobile devices are becoming more and more prevalent they can be used to distribute information.
In this thesis a system has been presented that processes energy data from suppliers and make it accessible for everyone. To reach as many users as possible the data is published via a cross platform mobile application running on most kinds of mobile devices. To motivate users to an active usage of the application gamification elements are implemented. A user is able to gain points for different actions and share this score on Twitter. With Firebase the system uses a powerful database that is able to store and synchronize data in realtime.
Usability tests revealed that Green Shifting still has some flaws, but altogether the feedback was positive. Nevertheless, several improvements need to be done by future developers.

Appendix A

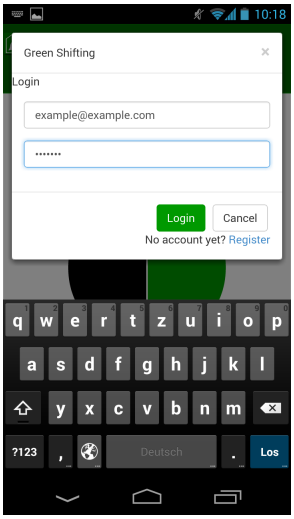
Screenshots



(a) Main view unauthenticated

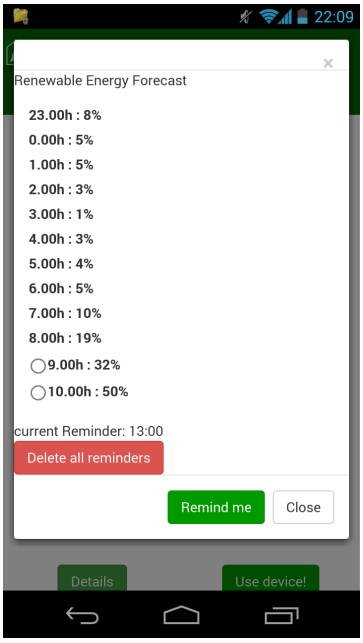


(b) Prompt for account creation

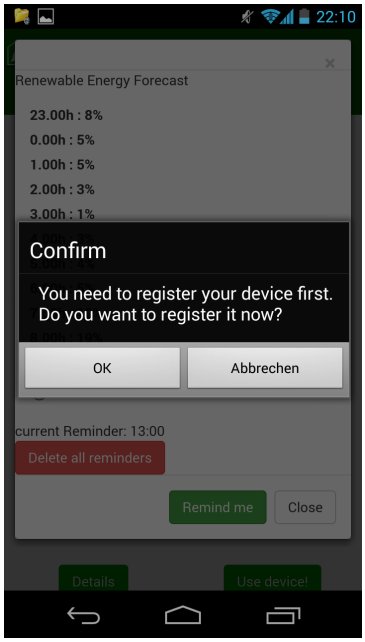


(c) Prompt for log in

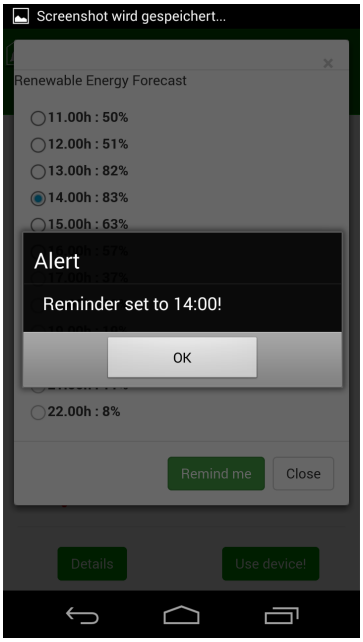
Figure A.1: First actions



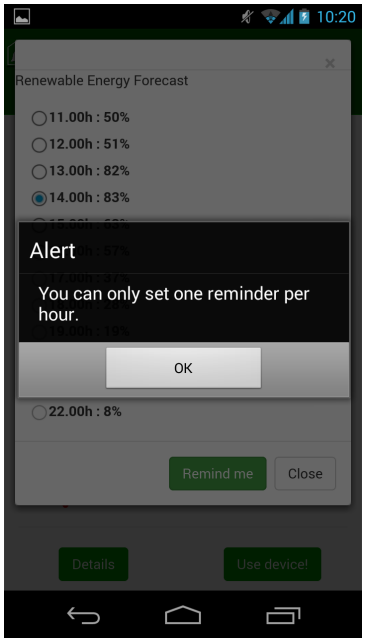
(a) Reminder menu with previous reminder



(b) Alert: Register device first

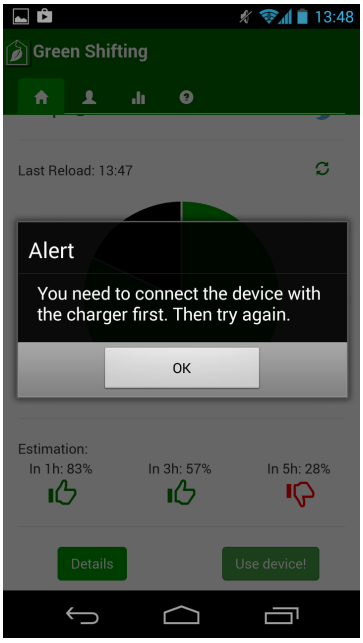


(c) Alert: Reminder successfully set

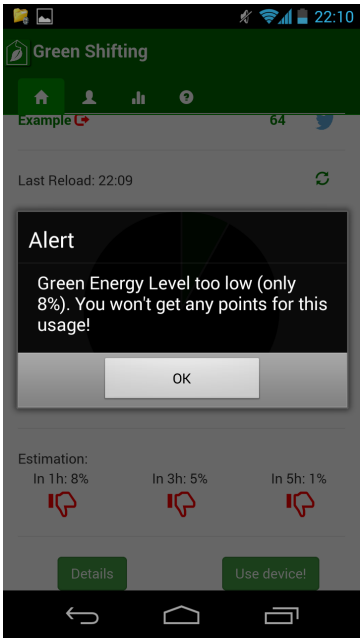


(d) Alert: Only one reminder per hour

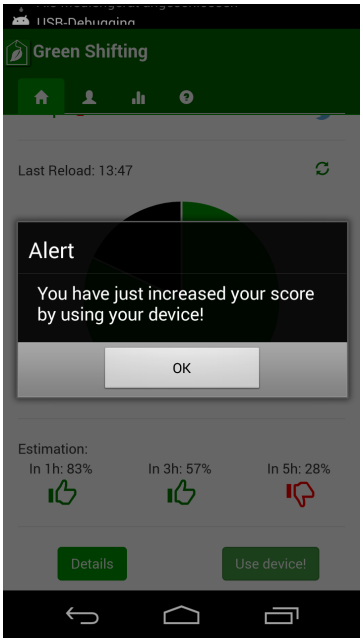
Figure A.2: Setting a reminder



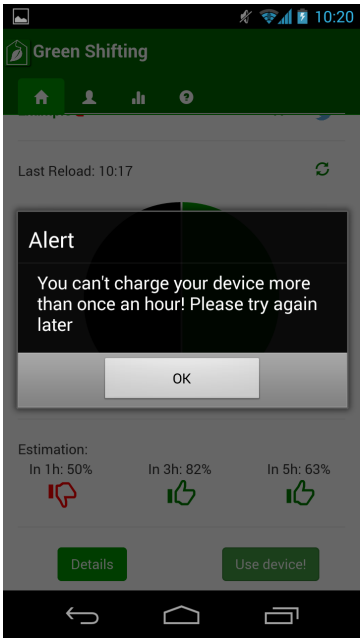
(a) Alert: Plug device in before usage



(b) Alert: Green energy level too low



(c) Alert: Device successfully used



(d) Alert: Only one device usage per hour

Figure A.3: Device usage

Appendix B

Usability evaluation

Questionnaire

Personal data

- Age:
- Education level:

Questions about energy consumption and app usage

- How important is sustainability to you?
  1 (not important) to 5 (very important)
- What do you think about renewable energies (solar, wind, water power)?
- Would you be interested to know the amount of produced renewable energy compared to conventional energy (coal, gas, nuclear) right now?
- Do you try to save energy? (E.g. turning the heater off if a window is opened, turn off the lights when leaving a room, keep the fridge closed,...)
- How often do you use your smartphone or tablet?
  1 (never) to 10 (whenever possible)
- How many apps did you install yourself (not counting the preinstalled ones)?
  - 0
  - < 5
  - < 10
  - < 20
  - > 20
- Would you shift the usage of time-flexible energy consuming gear like washing machine or dish washer to times with a high level of renewable energy?

- Would you be interested in using an app that shows you the current energy production for renewable and conventional resources?

Using the app - tasks

- install and start
- refresh
- create an account
- log in
- have a look around
- change the username
- set a reminder (including the registration of the device)
- tweet your score (if possible)
- recharge your device (if possible)
- log off

Using the app - evaluation

- How intuitive is the usage of the app in your opinion?
  1 (not at all) to 5 (totally intuitive)
- Did any errors or problems occur?
- What improvements do you suggest?
- Did the concept of the app convince you?
- Further comments

Technical data

- Kind of device: Smartphone or tablet
- Operating system
- OS version
- OS anomalies (e.g. CyanogenMod, Kies,...)

User description

- User 1 is a biology PhD student and 28 years old. She uses her Android smartphone regularly (8 of 10) and has 10-20 applications installed herself. She thinks renewable energies are important, but the technology needs further improvement and new technologies need to be supported better.
- User 2 is also 28 years old and a computer science Master student. She has rooted her smartphone and flushed it with a CyanogenMod Android version. On it 10-20 apps are installed, not including the preinstalled ones. She uses the devices regularly (7 of 10). She thinks that the growth in renewable energies is important but needs to be done carefully.
- User 3 is a 24 years old engineering Bachelor student. He has his Android smartphone flashed with an official HTC Sense that is officially not supported by his device. He uses the device regularly (8 of 10) and has less than 10 apps installed himself. He thinks renewable energies are innovative but too inefficient and need further development.
- User 4 is 27 years old and studies Astro Physics on a Master's level. He owns an Android smartphone and tablet (usability was tested on the smartphone). He uses these devices very often (9 of 10) and has 10-20 apps installed himself. Renewable energies are important to him.
- User 5 is a computer science Bachelor student and 31 years old. He owns an Android smartphone with 10 apps installed himself. He thinks renewable energies are important and need better consideration.

Bibliography

[Adm09] U.S. Energy Information Administration, *Emissions of Greenhouse Gases in the United States 2009*, Tech. report, U.S. Department of Energy, 2009.

[Amb00] Scott W Ambler, *Mapping objects to relational databases: What you need to know and why*, Ronin International (2000).

[app14] appcelerator, *Titanium*, http://www.appcelerator.com [visited: 22.02.2015], 2014.

[CL11] Andre Charland and Brian Leroux, *Mobile application development: web vs. native*, Communications of the ACM vol. 54 (2011), no. 5, 49–53.

[Cod70] E. F. Codd, *A Relational Model of Data for Large Shared Data Banks*, Commun. ACM vol. 13 (1970), no. 6, 377–387.

[DDKN11] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke, *From game design elements to gamefulness: defining gamification*, Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, ACM, 2011, pp. 9–15.

[Dow] Downie, Nick, *Chart.js*, http://www.chartjs.org [visited: 22.02.2015].

[Edd15] Eddy Verbruggen, *SocialSharing PhoneGap Plugin*, https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin [visited: 22.02.2015], 2015.

[EFH+11] Stefan Endlich, Achim Friedland, Jens Hampe, Benjamin Brauer, and Markus Brückner, *NoSQL*, Carl Hanser Verlag GmbH & CO. KG, 2011.

[Eur12] European Energy Exchange, *EEX transparency data*, http://www.eex.com/de/marktdaten/marktdaten-download/produktinhalte/transparenzdaten [visited: 22.02.2015], 2012.

[Gam11] Dave Gamache, *Skeleton*, http://www.getskeleton.com [visited: 22.02.2015], 2011.

[Gro12] Fabian Groh, *Gamification: State of the art definition and utilization*, Institute of Media Informatics Ulm University vol. 39 (2012).

[GWT09] Dominique Guinard, Markus Weiss, and Vlad Trifa, *Are you energy-efficient? Sense it on the web*, Adjunct Proceedings of Pervasive, 2009.

[HGH14] Mahnaz Hajibaba, Sarvenaz Golchin, and Veronika Henk, *Green Shifting*, Lab Project documentation (Unpublished), 2014.

[HJ11] Robin Hecht and Stefan Jablonski, *NoSQL Evaluation*, International Conference on Cloud and Service Computing (2011).

[HKS14] Juho Hamari, Jonna Koivisto, and Harri Sarsa, *Does gamification work?–a literature review of empirical studies on gamification*, System Sciences (HICSS), 2014 47th Hawaii International Conference on, IEEE, 2014, pp. 3025–3034.

[Huy13] Michael Huynh, *python-firebase*, http://github.com/mikexstudios/python-firebase [visited: 22.02.2015], 2013.

[Inc15] Adobe Systems Inc., *PhoneGap*, http://phonegap.com [visited: 22.02.2015], 2015.

[jQu15] jQuery Foundation - jquery.org, *QUnit: A JavaScript Unit Testing framework*, http://qunitjs.com [visited: 22.02.2015], 2015.

[KSCS09] Younghun Kim, Thomas Schmid, Zainul M Charbiwala, and Mani B Srivastava, *ViridiScope: design and implementation of a fine grained power monitoring system for homes*, Proceedings of the 11th international conference on Ubiquitous computing, ACM, 2009, pp. 245–254.

[Mar11] Ethan Marcotte, *Responsive web design*, A book apart, 2011.

[MS15] Inc. Motorola Solutions, *RhoStudio*, http://rhomobile.com/products/rhostudio/ [visited: 22.02.2015], 2015.

[MSW10] Friedemann Mattern, Thorsten Staake, and Markus Weiss, *ICT for green: how computers can help us to conserve energy*, Proceedings of the 1st international conference on energy-efficient computing and networking, ACM, 2010, pp. 1–10.

[NL93] Jakob Nielsen and Thomas K Landauer, *A mathematical model of the finding of usability problems*, Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems, ACM, 1993, pp. 206–213.

[OT12] Julian Ohrt and Volker Turau, *Cross-platform development tools for smartphone applications*, Computer vol. 45 (2012), no. 9, 0072–79.

[OTB] Mark Otto, Jacob Thornton, and Bootstrap contributors, *Twitter Bootstrap*, http://getbootstrap.com/ [visited: 22.02.2015].

[Rut] Richard Rutter, *Clagnut*, http://clagnut.com/sandbox/imagetest3/ [visited: 22.02.2015].

[TLNL] James Tamplin, Andrew Lee, Vikrum Nijjar, and Michael Lehenbauer, *Firebase*, http://www.firebase.com [visited: 22.02.2015].

[ZUR13] ZURB, *Foundation*, http://foundation.zurb.com/ [visited: 22.02.2015], 2013.

Declaration of Authorship

I hereby declare that this thesis submitted was formulated by myself. All direct or indirect citations are marked as such. I have used no other sources than those that are referenced.

This thesis was not previously presented to any examination office in the same or a similar form and has not been published.

Bonn, March 2nd 2015