



# Conception and Realization of a Mobile Crowdsensing Application for Support and Empowerment of Diabetes Patients

Master's Thesis at Ulm University

**Submitted by:**

Emanuele Giannotta  
emanuele.giannotta@uni-ulm.de

**Reviewers:**

Prof. Dr. Manfred Reichert  
Dr. Rüdiger Pryss

**Supervisor:**

Dr. Rüdiger Pryss

2017

Version December 21, 2017

© 2017 Emanuele Giannotta

This work is licensed under the Creative Commons. Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/de/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Satz: PDF- $\text{\LaTeX}$  2 $_{\epsilon}$

## Abstract

Diabetes is an increasingly common chronic disease. Not only in the Western World but across the globe. Statistics show, that 10.3% of men and 9.6% of women in the European Union aged 25 or older are forced to live with this disease, and the numbers are rising [1]. It is a chronic disorder of the metabolism, and a number of dangerous complications can arise if the medical treatment is not supervised carefully. These complications can be lethal, and severely reduce the quality of life of every individual patient. Thoroughly following the medical advices requires considerable discipline of the patient. This project aims to help the patients suffering diabetes, by providing a healthcare application to support them in their daily self-care. The application is called *Track your Diabetes*, and is designed to be a daily companion for the patient. Moreover, it is a mobile crowdsensing application and the collected data can empower researchers to improve the knowledge about the disease. Patient education, empowerment, and practical self-management to help deal with symptoms, is achieved by tracking diabetes related data such as the blood glucose level and weight. Depending on the data, the users can receive automated feedbacks, in order to improve their self-management. The application is questionnaire-based. This means, that the application only tracks data based on the questionnaires published. Patients can then share their data with their medical attendant who, in turn, can use it to improve their treatment methods. Designing this application, and providing prove for its feasibility by implementing it, is the central topic of this thesis. One major requirement was for the application to use the interface of a given RESTful API backend. This backend is planned to be a central database for the relevant data of all users. This work shows, how the application was designed and implemented for the operating system Android.



## Acknowledgments

At this point, I would like to thank all those who supported me during this thesis.

I wish to express special thanks to Dr. Rüdiger Pryss for being my thesis supervisor, for the valuable cooperation, and particularly for his active support which has been a considerable contribution to my work.

Also, I would like to thank Prof. Dr. Manfred Reichert, director of the Institute of Databases and Information Systems, for making this thesis possible.

Very special thanks are directed to my parents Teresa and Rocco Giannotta, who made my studies possible and supported me during this period.

Moreover, I thank my proofreaders for their support, particularly Petra Fabry, for helping me to improve my English skills.

Finally, I want to thank my friends and fellow students for a great time at the University of Ulm.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Objectives . . . . .	5
1.3	Thesis Structure . . . . .	6
<b>2</b>	<b>Background Information</b>	<b>7</b>
2.1	Diabetes Mellitus . . . . .	7
2.1.1	Causes . . . . .	9
2.1.2	Diagnosis . . . . .	10
2.1.3	Complications . . . . .	11
2.1.4	Therapy and Long-Term Prognosis . . . . .	13
2.2	Mobile Crowdsensing . . . . .	15
2.3	Xamarin . . . . .	18
2.3.1	Mono for Android . . . . .	21
2.4	CHRODIS PLUS - Joint Action . . . . .	21
<b>3</b>	<b>Related Work</b>	<b>23</b>
3.1	Healthcare with Smartphones . . . . .	23
3.1.1	Track Your Tinnitus . . . . .	23
3.1.2	KINDEX-Application . . . . .	25
3.1.3	Insulin Dosage Prediction . . . . .	25
3.1.4	Comparison of Healthcare-Apps . . . . .	27
3.1.5	Compliance Self-Monitoring . . . . .	28
3.1.6	Diabetes Digital Coach . . . . .	29
3.1.7	Big data . . . . .	29
3.2	Other Diabetes Apps . . . . .	29
<b>4</b>	<b>Requirement Analysis</b>	<b>33</b>
4.1	Functional . . . . .	33
4.2	Non-Functional . . . . .	35

<b>5</b>	<b>Architecture</b>	<b>37</b>
5.1	Design Objectives . . . . .	37
5.2	Architectural Aspects . . . . .	38
5.3	Application Process . . . . .	44
5.4	Local Data Model . . . . .	49
5.5	Background Threads . . . . .	51
5.6	Restful-API Back End . . . . .	52
<b>6</b>	<b>Selected Implementation Aspects</b>	<b>55</b>
6.1	Notifications . . . . .	55
6.2	Threads . . . . .	58
6.3	Backend Communication . . . . .	62
6.4	Questionnaire Management . . . . .	65
6.5	Basic Functionality . . . . .	67
6.6	Used Technologies and Frameworks . . . . .	67
<b>7</b>	<b>Introducing the Application</b>	<b>69</b>
7.1	Login . . . . .	69
7.2	Main Page . . . . .	70
7.3	Studies . . . . .	72
7.4	Questionnaires . . . . .	74
7.5	Basic Diabetes Tracking Functions . . . . .	76
7.6	Settings . . . . .	77
<b>8</b>	<b>Discussion</b>	<b>81</b>
8.1	Requirements Check . . . . .	81
8.2	Shared Code . . . . .	83
<b>9</b>	<b>Summary and Outlook</b>	<b>85</b>
9.1	Summary . . . . .	85
9.2	Future Work . . . . .	87



# 1

## Introduction

### 1.1 Motivation

Diabetes is a serious chronic disorder that causes a blood glucose level above the normal and healthy value. Once diagnosed it is a great burden on the patient. The treatment takes up a significant part of his everyday life.

According to statistics of the World Health Organization, 60 million citizens of Europe have diabetes. 10.3% of men and 9.6% of women, aged 25 or older, are affected [1]. The prevalence is still on the rising path [1].

The disease cannot be cured and might lead to several severe complications. These complications can not only diminish the patient's quality of life to a significant amount, but they can also be lethal. However, if treated properly, the severity of the disease can be reduced. This kind of treatment demands a lot of discipline and can be highly distressing. One major part of the treatment is to measure the blood glucose level several times a day. Living a healthy lifestyle is very important too. According to studies, overweight is estimated to be responsible for 65-80% of type 2 diabetes cases [1]. Hence, one part of living a healthy lifestyle is following a strict diet. This keeps the blood glucose level from rising and leads to weight reduction if the patient is overweight. Another important part of the treatment is doing physical activities more frequently, as sport activities also lower blood glucose levels.

Patients need assistance and guidance for their treatment. As explained later, in chapter 2.1, a major part of a successful treatment for the patient is providing detailed infor-

## *1 Introduction*

mation and medical advice by professionals. But also, the guidance of dieticians can be important, since being mindful of food intake is crucial for a successful treatment. However, medical appointments are associated with costs and so are appointments with dieticians. Ultimately, it is up to the patient to adhere to the treatment. If the patients' compliance is too low, more appointments with professionals are unlikely to help. Hence, the patient needs to be motivated to take his self-care seriously.

The economic impact of diabetes is not to be underestimated either. Depending on the health care system of the country it is either a burden, the patient needs to bare by himself, one the entire society bares, or both. The articles [2] and [3] show that this is a highly relevant issue. Since the percentage of the population being diagnosed with diabetes is rising, it must be taken care of. If patients can be guided towards better self-care and therefore need less medical attention, the financial impact could be lowered.

A competent coach constantly being at the patient's side, ensuring his compliance and supporting him with feedback, would be of considerable value for the treatment. Since a personal coach is too expensive for most people, a different way must be found. A smartphone could be used to fulfil this function. The capacity of smartphones in terms of computation and memory has risen over the past years. It is now possible to implement applications, which a few decades earlier would have been inconceivable. With the right application smartphones are capable of helping people with their health care. Much work has already been done in the fields of electronic health care. In [4] an eHealth system was developed for type 2 diabetes patients to teach them about their condition and develop their self-care abilities. In article [5] it was found that e-health can be an evidence-based tool to empower the self-management of diabetes patients. Article [6] demonstrates that e-health could even "provide equivalent diabetic control to usual care" [6].

One major advantage of smartphones is that they are widely spread and can be used for multiple purposes. In many cases it is not required to develop custom devices for the personal healthcare. Such custom devices are expensive to develop and therefore

expensive to buy also. It is much more cost efficient to use devices which are already provided. This is not an unknown fact. Just by searching for the keyword "diabetes" many applications can be found in Android's Play Store for tracking relevant treatment data.

This thesis aims to take advantage of the opportunities offered by smartphones. For this reason, the health care application *Track your Diabetes* was designed to support Diabetes patients. The application is based on questionnaires, in order to collect the patient's data. These questionnaires are organized in one or more studies. The application is extensible in terms of studies and questionnaires. This means that the head of the study can add new questionnaires, if required. When a whole new study is needed it can be added as well. For example, the treatment of type 1, type 2 and gestational diabetes (a particular type of diabetes during pregnancy) could be organized in separate studies. This way a type 2 patient will not be bothered with type 1 diabetes questions, concerning the insulin dosage. Also type 1 and type 2 patients who are not pregnant, will not need to lose time reading and skipping questions relating to gestational diabetes. Patients should be able to choose their personal and appropriate study programs in which they need to participate.

The main objectives of *Track your Diabetes* are the following:

- **Improve Self-Care:** The possibility of tracking blood glucose level and weight straight from the smartphone can make the everyday routine of the diabetes therapy easier for patients who would normally use pen and paper for tracking. By using the feedback function patients can be motivated to improve their self-care. For example, a questionnaire could ask the user how much physical activity he does. The answers could then be evaluated, and a proper feedback can be generated. If the patient has done well, he would be rewarded by being complimented. If the contrary is the case, a feedback would motivate him by reminding him of the advantages of physical activity. The article [7] indicates, that such a feedback function can also be a motivating for patients to use such an application, to begin with.

- **Patient Empowerment:** As it was stated above, doctor's appointments can be expensive. The user of *Track your Diabetes* is able to export his data and share it with his physician. The physician is then better informed and can decide better on necessary therapies. If the right questionnaire exists, he is able to determine what the patient might have done wrong and therefore give him better advises. By that, the appointments can be made more effective.
- **Research:** One major advantage of this application is for researchers to be able to create studies and gather valuable information. This can help their research to find new treatments and improve old treatments. From this point of view *Track your Diabetes* is a mobile crowdsensing application. These studies can be deployed in large parts of society and have the possibility to reach far more people compared to standard studies. This way more data can be gathered in a more cost-effective way. The infrastructure needed, is a backend server which can cope with all requests, the application, and mobile devices. As soon as the infrastructure is provided, studies can be deployed with very little effort.  
Studies can be protected by a password or the user can require an invitation for subscription. By this, it is assured that only the authorized patient answers the questionnaires.

This model can help patients in many ways. For example, a study for complication prevention could be deployed. A questionnaire asking the right questions could find out whether the patient shows symptoms of a secondary disease. If this is the case, via feedback, he could be sent immediately to see his doctor.

With these questionnaires the personal risk for complications could also be calculated. This is a big advantage, since the early identification of complications is of great importance for the treatment.

Also, side effects of the medication can be identified. The patient can then be notified via feedback to consult his doctor.

Using mobile crowdsensing in this field can have many advantages. With crowdsensing applications, neuropsychiatric symptoms of patients can be documented in real-time instead of retrospectively [8]. In [8] the authors have compared real-time assessment

data gathered by the crowdsensing platform *Track your Tinnitus* (TYT; see [9], [10], [11]) with assessment data that was collected retrospectively. They have shown that the results of these assessment types vary, and they see a chance for crowdsensing systems to improve the treatment of diseases involving neuropsychiatric symptoms. Another advantage of crowdsensing is that the data collected can be used for research. This can contribute in expanding the knowledge of the respective disease. Data collected by the TYT system has already been used for studies such as [12], [13] and [14]. In the future, diabetes researchers could also be empowered by data collected by *Track your Diabetes*.

In [15] it was investigated whether the crowdsensing platform TYT can make the tinnitus symptoms worse for patients using the application. They showed that this is not the case. For *Track your Diabetes*, this could also be a justified concern. Chapter 3.1.1 presents the TYT project in more detail.

In short, there is a high potential of helping diabetes patients with a smartphone application. Whether it is because the users can more comfortably track their data in their everyday life, get valuable medical feedback or because their contribution to studies improves the overall therapy possibilities.

## 1.2 Objectives

In context of the EU project CHRODIS PLUS (see section 2.4) and the research by the Institute of Databases and Information Systems, it is the objective of this work to design the above-mentioned application *Track your Diabetes*. A backend server is already provided in form of a RESTful API. Hence, the application needs to be able to use its interface and adapt to its conditions. The application should download all relevant data from the server. Therefore, the questionnaires and feedbacks needed for this project must be created, using the backend interface directly on the server. For the development of the application a test study and test accounts are provided. Questionnaires can be uploaded with the proper editor account and the app can be used with test accounts. The application uses the backend server not only for downloading data. The server can

## *1 Introduction*

also be used to upload the answers of the user to the questionnaires. This is done for two reasons. One is, to store the data for the user. If he buys a new smartphone, he can therefore synchronize his data. The second and more important reason is to enable researchers to use his data for scientific purposes.

Finally, the application will be implemented to prove the practicability of the concept. The targeted operating system is Android and the implementation will be done by using the cross platform Xamarin (see section 2.3).

### **1.3 Thesis Structure**

The rest of this work is structured into 8 more chapters. The second chapter introduces the basic knowledge for this work. There, the disorder diabetes is explained, as well as the framework Xamarin and the theoretic part of crowdsensing. Chapter three gives an overview of related work referring the topic of healthcare with smartphones. Then a few applications for diabetes patients are presented, that can be found in the stores of Android and Apple. The fourth chapter names the requirements for the application. The fifth chapter presents the architectural design. Chapter six will show how some parts of the design were implemented. In chapter seven the application is introduced with some screen shots. Chapter eight shows whether the requirements were met. Finally, chapter nine concludes this work with a summary and an outlook.

# 2

## Background Information

This chapter introduces basic background information required for this work. The first section provides an overview of the disease *diabetes mellitus*, the complications patients can have, and current therapy methods. Crowdsensing will be introduced in the second section. Finally, in the third section, the framework Xamarin, which was used for the application, will be discussed.

### 2.1 Diabetes Mellitus

Diabetes mellitus identifies metabolic disorders which are accompanied by a reduced production of the hormone insulin or by a reduced effectiveness of hormones. This kind of metabolic disorders are followed by a blood sugar level too high for the human body. If the before mentioned disorder is not treated properly, it might lead to severe complications. This chapter gives an overview to this disease. The information presented in this section was - if not cited otherwise - derived of [16].

First of all, it is essential to know that glucose plays an important role as energy supplier for most of the cells in the human body. Digestion carbohydrates are metabolized mainly into glucose, fructose and galactose. The percentage of glucose is about 80%. Fructose and galactose are almost completely absorbed in the intestinal tract and transformed into glucose in the liver. Finally, the blood sugar concentration value that circulates in the blood consists of more than 95% of glucose. This makes glucose "the final common pathway for the transport of almost all carbohydrates to the tissue cells" [17]. The amount of sugar in the blood is called the *blood sugar level*. Usually it is kept from the body at a

## 2 Background Information

level of between 70 and 100mg/dl [16].

Since diabetes is a metabolic disorder that affects the blood sugar level, it is a serious threat for the health and can be a life-threatening disease for the patient, if not under control. In a healthy human body, the blood sugar level is regulated by various hormones. There is only one hormone which can lower the blood sugar level. This hormone is called *insulin*. To increase the blood sugar level there are also several hormones:

- Glucagon
- Catecholamine
- Somatotropin (growth hormone)
- Cortisol

Not only a high blood sugar level can be dangerous. A low blood sugar level is called *hypoglycemia* and can have serious consequences too. A patient suffers from hypoglycemia if either his blood sugar level reaches values below 40mg/dl or values below 50mg/dl whilst having symptoms of hypoglycemia [18]. Symptoms of hypoglycemia are anxiety, shivering, paleness, sweating, nausea, hunger, weakness, fatigue, confusion and cramps. In severe cases the patient can even enter a coma [19].

Since both, a high and a low blood sugar level are dangerous, the body uses the above-mentioned hormones to keep it on a healthy level. By that it takes care of the energy supply of the central nervous system which depends on glucose. After a short fasting period, when the blood sugar level becomes too low, the antagonistic hormones of insulin mobilize the saved glycogen in the liver and muscles. If a longer fasting period is detained, glucose is formed within the liver and the kidneys using amino acids that can be transformed. When the blood sugar level becomes too high, insulin is produced by the body to lower it again. The hormone stimulates the absorption of glucose in muscle and fat cells. Hence insulin has anabolic properties, while the absence of insulin leads to a catabolic situation in which the body metabolizes proteins, fat and glycogen [16].



### **2.1.1 Causes**

In Germany about 8% of the population is estimated to suffer of diabetes with alarmingly rising tendency. About 90% of the diabetes patients have type 2 diabetes.

#### **Type 1**

Type 1 diabetes is characterized by a lack of beta cells in the pancreas. These cells are responsible for producing insulin and are destroyed by an autoimmune attack. Only when 80 to 90% of the beta cells are destroyed type 1 diabetes manifests. Predominantly the manifestation happens in the adolescent age of 15 to 25 years.

Patients with type 1 diabetes have a genetic predisposition. However, the concordance in identical twins is only 30 to 50%. Hence, there still has to be an exogenous factor - which is not yet identified - that activates an autoimmune attack at the beta cells. Possible exogenous factors like virus infections (mumps and rubella) are hereby discussed as well as toxins and chemicals.

Symptoms of type 1 diabetes are increased production and passage of urine, excessive thirst, weight loss and fatigue. If the disease remains undetected for a longer period, symptoms like itching, skin infections, and pains in feet and lower legs can come along. Even falling in coma is possible [16].

#### **Type 2**

A person who suffers from type 2 diabetes has developed a resistance to insulin. The hormone has a reduced capability or is no longer able to lower the blood sugar level. Genetic predisposition has a higher influence on type 2 diabetes. The concordance in identical twins is higher than 50%, and therefore higher than the concordance in identical twins for type 1 diabetes. The two main factors that cause type 2 diabetes are genetic predisposition and lifestyle of the patient. Malnutrition with obesity is observed in most cases when type 2 diabetes emerges. The constantly heightened food supply raises

## *2 Background Information*

the insulin production and generates a peripheral resistance to the, glucose regulating, hormone. Hence, more insulin is needed, as its efficiency is decreased.

Usually type 2 patients are older than 40 and almost all are overweight. It is likely that the insulin resistance already existed before the diabetes manifested. The patients rarely suffer from symptoms of insulin shortage. Still, sometimes they have symptoms such as itching, balanitis, decrease of the libido, and pain in the legs. Often the disorder is diagnosed by coincidence during a routine checkup when the blood sugar level is measured. At this point the patient can already have complications characteristic for type 2 diabetes. [16]

### **Gestational diabetes**

During pregnancy a gestational diabetes can manifest because of a peripheral resistance to insulin provoked by hormones. This form of diabetes passes off after pregnancy.

#### **2.1.2 Diagnosis**

The first step to a diagnosis is always an in-depth health history interview with the patient, whereby the family physician speaks in detail with his patient and asks relevant questions. After the interview the physician decides if there is a reasonable suspicion, that the patient might have diabetes. Usually a fasting blood glucose test is done to confirm the indication. If the blood sugar level then repeatedly exceeds  $126\text{mg/dl}$  on empty stomach, the diagnosis of diabetes mellitus is confirmed. With the help of an electrical glucose meter one is capable to measure his blood sugar level within 5 to 30 seconds. This glucose meter is an indispensable assistant for the patient's daily management of diabetes, however is not suitable for diagnostic purposes [16].

### 2.1.3 Complications

If not treated properly, all forms of diabetes can be followed by several diseases, most of which have a serious impact on the quality of life of the patient. In cases of absolute emergency, a patient can enter a so-called *diabetic coma*. It rarely results into an actual coma, but it makes intensive monitoring and treatment necessary. It has to be treated as quickly as possible.

There are two different kinds of diabetic comas:

- **Diabetic ketoacidosis:** By this mostly type 1 diabetes patients are affected. It is triggered by hyperglycemia which is the medical term for a blood sugar level that exceeds normal values. In order to get this kind of diabetic coma the blood sugar level is  $300\text{mg/dl}$  or higher. In comparison, a blood sugar level between 70 and  $100\text{mg/dl}$  is considered to be normal.
- **Hyperosmolar nonketotic coma:** This diabetic coma affects mostly type 2 patients. In this case the blood sugar level of the patient is by  $1000\text{mg/dl}$  or higher. The diabetic coma is also a consequence of hyperglycemia. Causes for this are dietetic mistakes or derailment of medication.

Patients suffering from a diabetic coma can have different states of awareness. Only 10% are unconscious, 20% are completely clear. The remaining 70% have an awareness state in between [16]. After a few days most patients enter a real coma. Symptoms of diabetic comas are among others an increased production and passage of urine, thirst, fatigue, sickness and vomiting. The general practitioner makes the diagnosis by finding out the symptoms. The diagnosis will then be checked in the laboratory.

Other than diabetic coma, which is a case of absolute emergency, diabetes can cause plenty of complications and secondary diseases. Only one percent of diabetes patients die of coma, but 70% to 80% die of *vascular diseases*, caused by diabetes. Hence, mainly the vascular complications [16] lead in fact to a shortened life expectancy and a reduced quality of life. In theory, the chronic hyperglycemia plays an important role

## *2 Background Information*

for the development of these diseases and some changes can even be reversed with a lower blood sugar level. In the following a few of them will be presented [16]:

- **Diabetic retinopathy and maculopathy:** This disease is characterized by damages of the eye's retina. If it is not treated, it can result into serious sight disorders and ultimately into blindness. Blindness of people between 30 and 60 years in developed nations is mostly caused by diabetes. After 20 years of diabetes up to 95% of the patients are affected by this disease. Type 1 patients are affected less than type 2 patients. This may be the case, because many type 2 patients live with the disorder for a longer period of time without knowing and hence without an appropriate therapy. Diabetic retinopathy and maculopathy can be treated by a timely laser therapy. It is important for diabetes patients to get their eyes checked regularly once or twice a year.
- **Diabetic nephropathy:** A diabetic nephropathy ranges from damaged kidneys up to a terminal kidney disease, that makes regular dialysis necessary. It affects type 1 patients as well as type 2, and 40% of the diabetes patients develop this disease. The life expectancy of a diabetes patient is determined by the emergence of nephropathy. After 20 years with diabetes, 25% of the patients are affected, and after 40 years 40% will be affected. For the treatment an early diagnose is important, since only in the early states something can be done. The therapy is based upon normalizing the blood sugar level and keeping it on a healthy level.
- **Diabetic neuropathy:** A patient has a diabetic neuropathy if he has damaged motor nerves, that have a restrained functionality and structure. After 10 years with diabetes about 50% of the patients suffer from such a disease. Especially feet and lower legs are concerned. Affected people have symptoms like numbness, prickle, sensation of cold or burning pain. Even cerebral nerves can be affected. The possibilities for therapy are unsatisfactory. It is important to normalize the blood sugar level.
- **Diabetic foot:** A diabetic foot can be triggered by minor local traumas such as inadequate nail care, or shoes that are too small. It can also be triggered by infections like the athlete's foot. Very often a diabetic foot does not cause any

symptoms, which is why an adequate prophylaxis is advised. The prophylaxis for example consists of keeping a healthy blood sugar level, rigorous foot care and the right shoes. It is important to detect a diabetic foot early, which is why regular examination is required. The therapy of a diabetic foot can require months of time and hence very much patience. The patient needs to relieve the foot completely. If not treated adequately, an amputation can become necessary.

- **Diabetic macroangiopathy (atherosclerosis):** This is the most common complication of diabetes patients. Not only diabetes patients can have this disease, but it is worse for them and it starts sooner. Risk factors are hypertonia, smoking and hyperinsulinism. It manifests as coronary heart disease, myocardial infarction, stroke, and peripheral artery disease. Around 65% of diabetes patients die of a macroangiopathy, and 75% of all deaths are caused by coronary heart disease. For diabetes patients a silent heart attack happens more often which is why a stress EKG should be done every one to two years.

### 2.1.4 Therapy and Long-Term Prognosis

If the patient complies strictly with the rules of the therapy, there is a good chance for him, not to have major restrictions of his quality of life. The resume is therefore, not to avoid or delay medical check-ups and treatments, some of which are mentioned in the section before (2.1.3). The overall goal is to keep the blood sugar level at a state between 80 to 120mg/dl and have regular check-ups. [16]

There are five fundamentals of the treatment:

- **Patient education:** The patient should be trained to be independent. They should make regular doctor's appointments for checkups. Furthermore, they should be motivated to take their diabetes care seriously in order to live as long a high-quality life as possible.
- **Self-management of the blood sugar level:** The patient needs to control his blood sugar level regularly.

## 2 Background Information

- **Diet:** It is crucial for the treatment to know what to eat and comply with it. A treatment without the according diet is pointless. Hence, a nutrition counseling is essential. For example, a patient should know, that alcohol has a lowering effect on the blood sugar level, which can result into a hypoglycemia. Moreover, if the patient is overweight, a weight reduction should be aimed. For type 2 diabetics it is even possible to have their metabolism normalized, once they have lost their extra weight.
- **Life style:** Regular exercise should be part of the patient's life style as well as to quit smoking.
- **Medication:** The medication should be taken carefully and punctually. Type 1 diabetes patients need to take their insulin regularly and type 2 patients may need to take antidiabetics. Furthermore, the patient should be informed of side effects. Type 1 diabetes patients can use an insulin pump instead of a pen. With that they can program the needed insulin and change the programming any time. They get a basal rate of insulin every hour, and they can give themselves a bolus by the push of a button, if needed. This spares them the four to six daily injections. However, to measure the blood sugar level regularly, will still be needed. The advantage here is that the pump can be customized to the individual insulin need over the day.

Whether the patient will get late damages as mentioned in section 2.1.3 or not and how severe they will become, depends on his personal diabetes care. Apart from that, genetic factors can contribute to how grave the vascular complications can become and how fast they manifest [16].

The total mortality of type 1 diabetes patients is at a rate of 50% after 40 years with the disease. Type 2 diabetes patients have a reduced life expectancy of about six to ten years [16].

## **2.2 Mobile Crowdsensing**

Mobile crowdsensing is the expression for collecting and sharing data via mobile devices, such as smartphones and tablets with various types of sensors. With these sensors a vast amount of useful data can be collected [20]. Not only smartphones can be used for crowdsensing. All consumer-centric devices, able to gather potentially important information for any purpose can be used. For example, in the area of transportation the users might be interested in roads with high traffic congestion and alternative roads. Data for this application can be gathered from smartphones as well as in-vehicle sensing devices. Also, many other internet-of-things sharing devices, like music players, can be used for crowdsensing.

One important aspect that makes smartphones so valuable for crowdsensing, is the fact that it has plenty of built-in sensors. Sensors like Accelerometer, Gyroscope, Magnetometer, GPS and ambient light sensors are only a few examples of current sensors of smartphones. In the future, smartphones could even have sensors for measuring the air quality [21]. Often sensors of internet of things devices are connected with the smartphone. Even a coffee machine connected to a smartphone might have potentially interesting data to share with the community about the surrounding environment.

Another advantage for the use of smartphones as a device for crowdsensing is the fact, that it already is widespread and versatile. People do not need to buy extra devices only for the sake of one crowdsensing application.

Furthermore, the data collected can easily be transmitted to the backend server through the internet.

The resourcefulness of today's smartphones in terms of computing power is undoubtedly one more advantage. It expands the possibilities of applications, compared to what was possible in the past. In addition, being a widespread device, it is possible to make large-scale case studies in a cost-efficient way. In many cases the buying of extra hardware will not be required.

Another advantage is that people can be leveraged to collect more complex data that

## 2 Background Information

could otherwise not be collected by a sensor. [21]

Many applications can be supported by crowdsensing. One example was already mentioned in the paragraph above. Another possibility could be to show and foresee traffic jams by using the data of smartphone maps and constantly update them. Drivers could then be lead through alternative roads to relieve the traffic situation. In the past, drivers had to listen to the radio news if they needed traffic information. Some of these news were even collected by people in traffic helicopters. In the future, all the information about traffic could be collected by one's smartphone, then shared with other users.

Another trend is to use crowdsensing in order to improve the health care sector. As will be explained in chapter 3, there are already many different approaches only in the sector of diabetes. Sometimes the aim is to improve the life quality of people with chronic diseases or to collect data in order to learn more about a disease for improving prognosis of therapy. One example is the *Track Your Tinnitus* Project, introduced in section 3.1.1.

In [21] the crowdsensing applications are divided into two categories:

- **Personal sensing:** This category includes applications collecting data of an individual person, e.g., for health care.
- **Community sensing:** Here something is measured by a collective which would be difficult to measure by one person alone. Detecting traffic congestion is an example for this category. Community sensing can also be referenced to as one of the following:
  - **Participatory sensing:** The user is required to actively contribute data.
  - **Opportunistic sensing:** The collection of data is done more autonomously without the user taking action.

The advantages of using smartphones for crowdsensing have already been discussed. Challenges will be discussed in the following.

Although it is a great advantage that smartphones are already wide spread through



the population, a challenge is, that a very large variety of devices exist. Not only many old devices are still in use, they are produced from different companies. Therefore, the resources and sensors can also vary. As a result, the collected data could have different quality features. For some applications it might be necessary to determine the right set of devices first. [21]

The computing power of smartphones may have risen constantly in the last years. Still, the devices have limitations, not only in respect to computation but also in terms of energy and bandwidth. Various crowdsensing applications run independently of one another, however sometimes even collect the same information from the same sensor. There is a lack of communication between applications - and the result is, that the limitations of the smartphone are reached more quickly. In order to make it more efficient, developers need to address those limitations while designing their application. [21]

Another point is the users' motivation to use the application in the first place. If users are not motivated to use or even install the application, data cannot be collected at all. Hence, a developer of a crowdsensing application needs to offer incentives in order to recruit and engage users [21].

The authors of [22], for example, considered to pay the users for their involvement. Among other things they compared the following three schemes:

- **Uniform scheme:** Everyone gets paid 4 cents per task.
- **Variable scheme:** Users are paid 2 to 12 cents per task judged by their performance.
- **Hidden scheme:** The users can win a prize if they participate.

The authors found, that the variable scheme is 50% more cost effective in terms of completion rate, and performance compared to the uniform scheme. The hidden scheme on the other hand was the least effective.

One crucial point is the privacy concerns of the users. Different users have different views of what is too private for them to share. Some may have no problem constantly

## 2 Background Information

sharing their current location, others do. The main point is to ensure privacy and security on the one hand while, on the other, keeping the data useful for analysis. In [21] the following three techniques are mentioned:

- **Anonymization:** Remove all identifying characteristics.
- **Multiparty computation:** Use cryptography to preserve privacy.
- **Data perturbation:** Randomly modify the collected data. When the arithmetic average of all data is computed, the random modifier can be removed, and an accurate result is gained. An example would be the measuring of the average population weight. A random number of 100 to 200 can be added to the individual measurement. If every number between 100 and 200 has the same probability to be chosen, then the average of 150 is added. After computing the average, 150 has to be subtracted. The result is accurate, if the data set is big enough.

In [22] the authors found out that privacy awareness can enhance the performance of crowdsensing applications by eliminating the privacy concerns of potential users. If successfully eliminated, more users may contribute. The authors explained the *coalition strategy* by which the users share their information within a group. This adds a k-anonymity privacy protection.

In the article [23] a state of the art analysis of various privacy techniques is made. They found an "orthogonal relationship between data quality, privacy and resource consumption". That is the case, because effective privacy mechanisms consume more resources.

Another challenge of mobile crowdsensing is the possibility to test the application within a crowd before releasing the application to the public. For that Fiandrino et al. introduced a simulation platform, called CrowdSenSim [24]. The platform has been designed especially for simulations of crowdsensing in realistic urban environments.

### 2.3 Xamarin

Xamarin is a cross-platform programming framework for mobile apps. It supports the three platforms Android, iOS and Windows Phone. With this, framework developers

are able to write native applications for the various operating systems in .NET, all using C#. APIs for each platform are provided and also wrappers for functions, so that they can use the platform-specific components such as the user interface, notifications, animations, and can access sensors with the proper access rights. Developers are able to program applications without knowing every detail of the respective operating system since Xamarin takes care of the aspects, which are not important for development [25]. In addition, since it is a cross-platform, if someone wants to program an application for two or all three of the supported operating systems, there is a section of code, which can be shared among all platforms. This code is called *shared code*. How big it can become, depends on the application type and on the developer, himself. If OS specific libraries are used in a code section, this section cannot be shared. [25]

Development environments which can be used for Xamarin, are *Xamarin Studio* for Windows and Mac and *Visual Studio* for Windows.

A distinction is made between *Xamarin.Android*, *Xamarin.iOS* and *Xamarin.Forms*. Xamarin.Android and Xamarin.iOS are "C# bindings to the native Android and iOS APIs for development on mobile and tablet devices" [25]. For each new release of Android or iOS a release of Xamarin follows to include the new APIs. At the same time developers can use .NET features like data types, garbage collection and many more.

Xamarin.Forms on the other hand is an additional "layer on top of the other user interface bindings and the Windows Phone API" [25]. Figure 2.1 shows these layers. While the User Interfaces (UI) in Xamarin.Android and Xamarin.iOS can only be used for the respective operating system, Xamarin.Forms has a cross-platform UI library. The set of UI controls provided by Xamarin.Forms is then mapped to the native bindings of Xamarin.Android and Xamarin.iOS. By this the developer can create one UI for all three operating systems. Thereby, the developer only needs little knowledge of Android and iOS if he just wants to create basic UIs. Furthermore, this obviously increases the quantity of shared code. However, developers can use UI controls of the native bindings in Xamarin.Forms by using *custom renderers*. For example, this can be used,

## 2 Background Information

if something is missing. Figure 2.2 shows the application architecture, when custom renderers are used. [25]

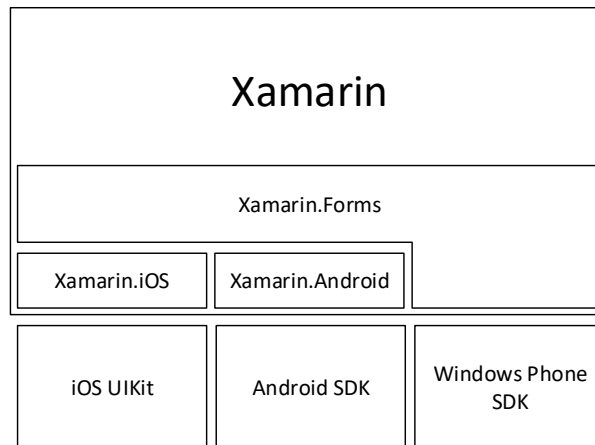


Figure 2.1: "Xamarin libraries bind to native OS libraries" [25].

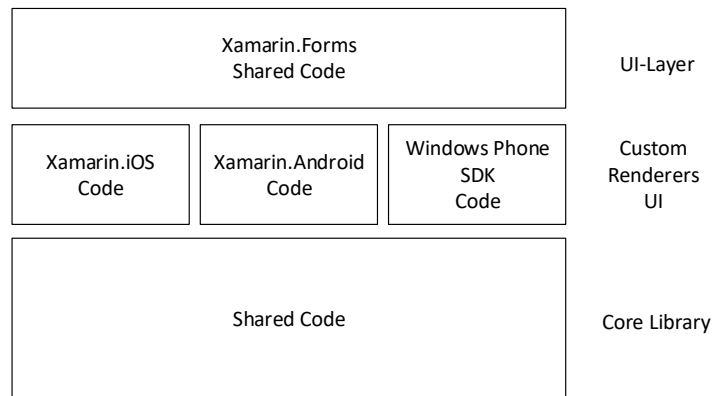


Figure 2.2: "Xamarin.Forms architecture with custom renderers" [25].

One major advantage of the native Xamarin bindings is, that they are better established since they are existing for a longer time and still provide more of the features provided by

the original platforms [25]. Furthermore, the access to native UIs is closer to the original UIs.

In [25] it is recommended to use Xamarin.Forms, if the developer is getting started with development in C#, for basic business apps, and for apps with basic design. The author advises to use the native Xamarin frameworks for complex screens, apps with complex visual design and for single platform apps. By using custom renderers in Xamarin.Forms, "virtually everything" is possible that can also be done with the native Xamarin frameworks. [25]

### 2.3.1 Mono for Android

Xamarin.Android uses *Mono for Android* to run the applications natively "without a major performance trade-off" [26]. While Android apps usually run within the *Dalvik VM* or its successor, the *Android Runtime*, Xamarin.Android uses the Mono CLR in cooperation with the Dalvik VM or Android Runtime. Mono CLR classes allow the developer to access the OS features that cannot be accessed by .NET itself.

## 2.4 CHRODIS PLUS - Joint Action

Joint Actions are designed and financed by the EU member states and aim to improve health. CHRODIS PLUS Joint Action ([27]) is a three-year EU initiative against chronic diseases, such as diabetes, cardiovascular disease, cancer, and mental disorders. It was started 2017 and runs until 2020.

In [27] it is stated, that chronic diseases cost EU economies about 115.000.000.000€. Worldwide it takes up to 70%-80% of healthcare budgets [27]. Thus, the Joint Action (JA) CHRODIS PLUS was started as successor of JA-CHRODIS (2013-2016). The objective of CHRODIS was to promote health and prevention. Also, diabetes management and multi-morbid chronic conditions were a priority.

## *2 Background Information*

CHRODIS PLUS aims to make use of the "wealth of knowledge" ([27]) on ways to prevent and manage chronic diseases. It should empower patients and improve their quality of life. By that, costs can be reduced, and health systems can be made "sustainable and responsive" to the demographic change of an aging population.

# 3

## Related Work

This chapter gives an overview to work done in the field of smartphone supported health-care. The focus is on mobile crowdsensing and diabetes.

### 3.1 Healthcare with Smartphones

In this section, some articles how health-care can be improved by computers and smartphones, will be introduced.

#### 3.1.1 Track Your Tinnitus

An important project by the *Institute of Databases and Information Systems* is *Track your Tinnitus* (TYT). Pryss, Reichert, and others present this project in their articles [9] and [10]. They developed a concept for a crowdsensing system, to help people suffering from tinnitus and, at the same time, collect relevant data for better understanding of the symptom. This is highly relevant to people who are affected because, due to the variety of tinnitus forms, the symptom is very difficult to treat. It can only be measured subjectively, and the awareness of it changes in different situations. The aim is to collect "large and qualitative longitudinal data sets" [10], not only concerning the symptoms, but also the treatment and its effectiveness. This data can then be evaluated by professionals to improve treatment possibilities. To reach these goals, the authors chose the approach of using mobile crowdsensing. Mobile devices collect the data needed by an application while the backend server stores it. Since smartphones are wide spread, the authors can reach many participants, and the desired large-scale data can be collected.

### *3 Related Work*

When the user starts interacting with the application, he fills out three questionnaires regarding the "stable tinnitus characteristics" [10]. Then, the regular tinnitus measurement starts. Since perception of tinnitus is subjective, it is measured by a questionnaire also. The user is notified to fill in the questionnaire at random points in time. Moreover, the environment sound is recorded by a microphone, while the user is filling out his questionnaire. The user's privacy is secured by using pseudonyms.

Users of TYT can receive feedback from the application. This feedback function was presented in [7]. Feedback is generated automatically by using the patients' data that was captured in real time and by considering his inputs in the past. His feedback can be sent to the respective physician, also. Such a feedback function seems to motivate patients to use the application [7].

This way, a global tinnitus study can be conducted to help affected people. Compared to regular studies, this approach collects larger amounts of data in a more cost-efficient way.

The study [14] compared three data sets, collected differently of people with tinnitus. One data set was gathered by an outpatient tinnitus clinic, one from a self-help web platform and one from TYT. The study showed, that crowdsensing applications like TYT have the potential to recruit groups of users that would be harder to recruit by an outpatient tinnitus clinic.

Recruiting as many people as possible is important. This way, more data can be collected, and researchers empowered. TYT data has already been used for various studies.

In [12] the authors researched, whether the patients' emotional state influences their tinnitus distress. Among others, they examined the association of tinnitus loudness and tinnitus distress and how they relate to stress. They found, that stress "significantly mediates the association" between loudness and distress [12].



The study [13] used TYT data to find out, when the tinnitus is loudest and most distressing. Their results show, that this is the case between 12 a.m. and 8 a.m. The authors propose to take circadian rhythm into account to improve the therapy.

#### 3.1.2 KINDEX-Application

This application is presented in [28]. The objective is to help pregnant women, gynecologists, and midwives to assess psychological risks for the unborn child. For that, it uses the "Konstanzer Index" (KINDEX), which is an interview developed for this purpose.

It has been empirically proven, that the development of children can be negatively influenced because of psychosocial factors concerning the mother, before her child is born [28]. With the KINDEX, such factors can be found. However, in daily practice the medical experts often do not have enough time to conduct and evaluate the interview. That is why, an application for tablets was developed. Pregnant women can now fill in this questionnaire by themselves, using a tablet. Later it is evaluated automatically, and the gynecologist or midwife can access the results directly.

#### 3.1.3 Insulin Dosage Prediction

In their article [29], Preuveneers and Berbers present a model to support type 1 diabetes patients by choosing the right medication dosage. The idea was to collect data about the user by the smartphone and then, based on this data, estimate the dosage of insulin he may need. Afterwards, the user would be informed about the estimated dosage.

The authors aimed to investigate how useful the smartphone can be for diabetes patients. The application was designed to not need any further hardware other than smartphones and items, diabetes patients already have, e.g., a blood glucose meter. They wanted to give diabetes patients the chance, to replace paper on the one hand, and supply them with "similarity measurements with previous situations" [29] on the other. Furthermore, they wanted to examine the overall suitability of smartphones for personalized health

### 3 Related Work

care assistance.

Some of the users' data collected for the experiment was the following:

- **Blood glucose level:** The user saves his blood glucose level into the application, which he needs to measure anyway.
- **Nutrition:** The user saves information about his eating habits into the application.
- **Location:** The application constantly tracks the location of the user via GSM.
- **Activity:** The user names his daily activities, which can influence the blood glucose level, such as nutrition or sports.

By evaluating this data, a trend can be created for each individual user. The users' location can be assigned to activities. By using information from the past, the application is able to predict which location the user enters next. Also, depending on his location, the user's activity can be determined. Based on these predictions, the application forewarns the user if his blood sugar level might change considerably. This kind of predictions can only be made, if the living habits of the user are consistent to some degree.

This is a highly valuable feature, since both hyper- and hypoglycemia, are threats to the patients' health and are likely to happen with diabetes (see 2.1).

The authors found, that a short training phase for the application can suffice, to find relevant habit patterns, that let the users blood glucose level vary too much. Since it is of central importance to constantly have access to the users' location, a survey was conducted to find out when smartphone users they take their smartphones along with them. Indeed, judging by their habits, it is feasible to have access to their location often enough. GPS would be a much more precise indicator for the location. However, the authors decided to determine the location using GSM since it is more resource saving than GPS. Besides, the application does not need the exact location. It is more important to determine the users' actual activity. However, after the application's training phase, the location could be determined with sufficient accuracy. The application was able to calculate the current activity of the user and the next place he will visit with a "certain

accuracy" [29] of 55% to 85% on average. However, the prediction of the users' next activity was even more accurate.

The authors demonstrated the feasibility and practicability to support decisions of type 1 diabetes patients concerning the insulin dosage. This was achieved by gathering information with the users' smartphones. Surveyed volunteers stated, that they would be ready to use their smartphone more frequently, if they had the prospect of gaining an improved live quality.

#### 3.1.4 Comparison of Healthcare-Apps

In [30], existing healthcare apps were explored and evaluated. Moreover, the authors show possibilities of smartphones to become a medical toolkit for the private individual. They state, that smartphones could in the future be used as a medical equipment, due to their sensors. The device can become a "multipurpose mobile medical equipment" [30], which can be utilized by both, private users and medically trained staff. For example, with the proper application, the microphone could be used as a heart rate monitor. Therefore, costs of medical care could be reduced, since smartphones would be able to replace medical equipment. People could get professional advice without visiting the doctor. This can also be improved by the aid of artificial intelligence. By empowering patients, physicians would have more time for those patients, who need their attention more urgently.

While analyzing healthcare applications, the authors considered aspects like: efficiency, cost, simplicity and effectiveness. They found, that most of the applications are "glossaries of clinical information", containing profound or basic knowledge. Consequently, apps for real "clinical data management and monitoring health" are a minority, and "evidence-based or professional-informed apps" are very rare [30].

The authors addressed the need of peripheral devices in many diagnostic procedures as a cause for the shortage of the above-mentioned health care applications. Also, medical

### 3 Related Work

applications need certification. Moreover, the process for application certification can vary from country to country. Another problem is, that many different smartphones are in use. Their sensors vary in quality, which makes the certification process even more difficult. The authors think, that for future cases, where smartphones may really be used as medical equipment, applications can only gain certification for being used by trained staff. Still, these applications could be useful in societies with little medical regulation.

#### 3.1.5 Compliance Self-Monitoring

Paper [31] introduces a system for supporting decisions of diabetes patients and measuring their compliance. When the compliance index of patients is known, their care-needs can be prioritized. The system tracks data such as the blood sugar level, a patient's lifestyle, his mood, and complication prevention attributes. The system evaluates this input and helps the patient, if required. For example, depression can be a serious problem for diabetes patients. If, after data evaluation, the application finds that the patient might be suffering from depression, the system can notify his medical expert. As a result, the medical expert can take care of the depression.

The system is divided into two parts:

- **Patient's hub:** The patient's hub is accessed via smartphone. Medical sensors, for collecting the required data, are linked with Bluetooth. Also, the patient can interact with the management hub by using the patient's hub.
- **Web-based disease management hub:** This is the core of the system. The data collected is stored, evaluated, and monitored here. In addition, the management hub contacts the devices as well as human actors such as patients, physicians, and others. Decisions the management hub makes, consider the patient's individual treatment.

### 3.1.6 Diabetes Digital Coach

In [32], a so-called *test bed* is described, where the authors combined several healthcare technologies for diabetes. Before, they identified the problem, that innovations on this sector are often made in isolation to others. Also, new applications do not offer anything new in many cases. Furthermore, they state, that the utility of many applications is not really proven since most data is collected experimentally. This lack of evidence prevents investors of supporting the development of such applications.

The authors aim to help diabetes patients to make the right decisions by implementing an application that combines other healthcare technologies. In addition, they want to solve the problem of missing evidence, mentioned above, by enrolling 12000 people in this *test bed*, and thereby close the gap.

### 3.1.7 Big data

In the sector of psychology, Monteith, Glenn, Geddes, Whybrow, and Bauer examined, whether and how big-data can help to treat people suffering from bipolar disorder. In short, the aim is to collect data from patients and thereby achieve better understanding of the disease. The collected data comes from electronic medical records, smartphone applications, sensors and internet activities. In summary, they wanted to identify the challenges and opportunities of using big data in the medical field. [33]

## 3.2 Other Diabetes Apps

In this section applications will be shortly introduced, which can be found in the stores of Android and Apple. There is a vast amount of diabetes related applications. Most of them are diary apps for tracking diabetes related data or for educating patients.

The following is an overview to some applications found:

### 3 Related Work

- **mySugr:** The users can track their blood glucose level, medication and nutrition. This data is depicted clearly for the user. They can also get medical advice from the application. It can be installed on Android and iOS. For iOS mySugr offers two additional applications, called *mySugr Scanner* to connect one's glucose meter, and *mySugr Academy*, that offers information for type 2 diabetes users. [34]
- **Social Diabetes:** This application is a diary for diabetes patients of both types. Other features are customized notifications and connect a personal glucose monitor. Interestingly, patients can connect themselves to their medical expert. Thereby, the expert can view his patient's progress and, if necessary, adjust his treatment. [35]
- **mapmydiabetes:** This software is used as an education platform. Target groups are not only patients, but also clinicians and commissioners. Patients can use it to learn about their health condition and what to eat. They can also share their information with their doctor. Doctors, on the other hand, can use their patient's data to motivate them for better self-management and improve their quality of life. By that, doctors can save time. However, a mobile application is not yet available. [36]
- **OVIVA:** OVIVA is an application for Android and Apple. This application is mostly specialized on "diet-related health conditions". The application generally targets people who should maintain a diet for some reason. This makes it useful for diabetes patients, too. Physicians and dieticians can participate, also. With this application, dieticians can coach their clients, wherever they are and save time. Patients can track their food by taking photos. This way, they do not have to write down every single ingredient. They can also track their daily activities and weight. [37]
- **Diabetes Forum:** This mobile application is a diabetes forum in the UK [38]. It is listed here, because it can be used by diabetics and their relatives for reading about this topic and helping each other. They can access the forum anytime, if they have a smartphone and an internet connection.
- **Gadge Diabetes Care:** Gadge Diabetes Care Center is a clinic for treating diabetes in India. With this application users can be consulted virtually or by video,

from the doctors of the clinic. They can also use the application to set appointments. [39]

- **Diabetes Self-Management:** This app is a kind of magazine. Here, diabetes related articles and recipes, which are safe for diabetics, are posted. Hence, it is an educational platform. The application is financed by a membership fee, however one month of free trial is offered. [40]
- **Diary apps:** There is a whole section of applications, designed to store the user's diabetes related information, such as the blood glucose level and body-weight. Some of them also track activities and the users' mood. Most of them present the data, using diagrams to give a clear overview and they also make the user's data available on other hardware. Examples for such applications are:
  - **Sugar Diary** [41]
  - **Diabetes:M** [42]
  - **Diabetes PA** (Diabetes Manager) [43]
  - **Diaguard** [44]
  - **Diabetes Connect** [45]
  - **Sugar Sense** [46]

Many more applications could be listed here. Particularly the Play Store offers a big selection.





# 4

## Requirement Analysis

In this chapter the requirements for the application will be introduced.

### 4.1 Functional

Functional requirements for the application are:

1. **Registration within the application:** The application cannot be used without user account. Therefore, it should be possible to create such an account directly on the device.
2. **The application can be used without internet connection:** Users do not need to have internet connectivity to use the application, as they might have poor connectivity or none at all. Therefore, relevant data needs to be saved, until internet is available to send it to the backend.
3. **Participation at multiple studies is possible:** Users should be able to subscribe to multiple studies and fill in the respective questionnaires. Furthermore, users should have the possibility of unsubscribing at any time.
4. **Studies can have different states:** Studies can be public or private. Subscription to public studies has no restrictions. For private studies authorization is required to subscribe. Persons entitled can change the status of studies, making them public or private.
5. **Invitations to studies are possible:** The head of the study should be able to invite users to his private study. The user receives a notification via e-mail.

#### 4 Requirement Analysis

6. **Invitations can be accepted:** Users can view their invitations in the application and accept them.
7. **Participation in private studies:** If users know the respective password, they can subscribe to a private study.
8. **Statistical questionnaires can be filled in within a study:** The head of the study can publish arbitrary questionnaires to fulfill the purpose of his study. Subscribers can view these questionnaires in the application.
9. **Statistical questionnaires can be extended or changed:** The head of the study can change existing questionnaires and add new ones.
10. **Questionnaires can be deactivated:** A questionnaire that has been deactivated cannot be filled in.
11. **The questionnaire state can be changed:** The configuration of questionnaires determines, whether they must be filled in once or multiple times. If a one-time questionnaire has already been filled in, it must not be shown to the user again.
12. **Statistical questionnaires can be filled in within the application:** Questionnaires can be filled in on the website and in the application.
13. **Synchronizing of the results:** Data gathered by the application should be sent to the backend server for research.
14. **No initial values:** Initial values should be avoided, since users could be influenced by them.
15. **Notifications for questionnaires:** Questionnaires can have schedules. Users should be notified by the application, if a questionnaire must be filled in, according to its schedule.
16. **Notification schedules can be changed:** Heads of studies can set questionnaire schedules to be fixed or variable. If the schedule is variable, the user should be able to change it.
17. **Show results in the application:** Users should be able to view their blood glucose level and weight in the application.

18. **Export data:** The users should be able to export their data, in order to share it with their physicians.
19. **In-app language settings:** The application should be designed to be multilingual. Users should be able to change the language within the application.
20. **Standard studies:** The application should have a standard study to track blood glucose level, weight and physical activity.

## 4.2 Non-Functional

- **Splash Screen:** When users open the application, a splash screen should be shown. This prevents a white screen, while the application is loading.
- **RESTful Backend API:** The application needs to cooperate with the given RESTful Backend API ([47]).
- **Questionnaires:** The application should be delivered with questionnaires. These questionnaires should be extracted from [48], [49] and [50].



# 5

## Architecture

This chapter introduces the architecture of the *Track your Diabetes* application. First, some design objectives will be presented, then architectural aspects such as UML excerpts. Afterwards, the application process, the database model used, and finally the backend server, will be introduced.

### 5.1 Design Objectives

While the concept of the application was designed, some design objectives were considered. These objectives were:

- **Intuitiveness and Simplicity:** The head of a study might be interested in having as many participants as possible. For that, it is important to recruit many users. Thus, the design of the application should be kept simple. It should be possible to understand the application without reading a manual, and people with little smartphone experience should be able to use it.
- **Extensibility:** The application should be extensible. If new features are needed, it should be possible to extend the user interface (UI) rather than reinvent it.
- **Modularity:** The application design should be modular. For example, if the head of a study needs new question-types, the developer should only need to change the part of the application which handles questionnaires. He should not need to change the database or the backend communication.

- **Model-View-Controller Pattern:** The application architecture should be designed following the Model-View-Controller pattern ([51]) to separate data, logic, and user interface.

### 5.2 Architectural Aspects

This section will introduce the activities and fragments of the application, and how they work together.

One objective for the application's design was to cover the whole functionality using as little activities as possible. The application has nine activities. One of these activities is used for the splash screen, which appears when the application is opened. This splash screen bridges the time between starting the application and showing the actual user interface, e.g., the *MainActivity* or *LoginActivity*. Thereby, it is avoided that the user stares at a blank display when he opens the application. Hence, the splash screen is not part of the actual functionality but rather a design issue. Therefore, the functionality of the application is covered by eight activities in total, two of which were already mentioned before. Here is a short introduction of these activities:

- **MainActivity:** In Android, one activity must be referenced as the entry point of the application [52]. This is done in the Manifest file. In Xamarin.Android this class becomes the `MainLauncher` [53]. Originally, it was planned for the *MainActivity* to be the entry point of the application. However, the implementation of the splash screen required the *SplashActivity* to be defined as entry point. Otherwise, it would not have been possible to show the splash screen after the application was opened.

However, the *MainActivity* is the central class of the application. It contains the home screen and navigation menu. From here the user can access the whole functionality of the application. If the user is logged in, this is the first activity he sees after the splash screen appears.

- **LoginActivity:** If the user is not logged in, this activity is the first being called after the splash screen. With this, users can register, log in, and reset their passwords.
- **MyAccountActivity:** With this activity, the user can view his personal data and change it. Also, he can change his password.
- **StudiesActivity:** Here all studies are listed whether they are public, private, subscribed, or unsubscribed. Invitations can be viewed as well. Normally, studies have descriptions and other information. This information is shown when the user selects one of the listed studies. Then, he can subscribe, unsubscribe, or request an invitation.
- **BloodSugarActivity:** This activity provides standard functionality of diabetes applications. Hereby, the blood glucose level can be tracked. Also, users can set their preferred unit and view a diagram of their past blood glucose levels.
- **WeightCaptureActivity:** This is the counterpart to *BloodSugarActivity* for tracking body-weight.
- **QuestionnaireActivity:** Hereby, users can list their questionnaires, view its information, and answer the questions.
- **SettingsActivity:** This activity lists the settings of the application. Among other, users can modify their notifications and export their data.

The user interface of the activities listed above will be introduced in chapter 7. Most activity-layouts are only designed to display fragments. In these cases, the whole user interface is implemented within those fragments. This approach is used when the user interface (UI) needs to be changed frequently [54]. Only *MyAccountActivity* and *SplashActivity* are not designed this way. Figures 5.2 and 5.3 are two parts of one simplified UML. They show all activities and the fragments they use.

Fragments are also used to provide modularity and extensibility. For example, figure 5.2 shows that *BloodSugarActivity* and *WeightCaptureActivity* are attached to one fragment only. This could have been done without using any fragments which would save the expenses of extra variables used for fragment management. However, the two

## 5 Architecture

activities are more extensible this way. If future developing requires a second UI in one of those activities, it can easily be extended by providing a new fragment.

Figure 5.1 shows how fragments are used. Most activities use the approach in figure 5.1a. They have a slot for fragments, called *FrameLayout*. Fragments are switched within the *FrameLayout* based on which UI is required. The answer sheets follow another design, which is depicted in figure 5.1b. Here, the *AnswersheetFragment* is always used to display the question. The third fragment is chosen depending on the question type, e.g., multiple-choice or single-choice. If the head of the study needs a new question type, the developer can design a new fragment with the proper UI. Then, he can embed it into the *AnswersheetFragment*. This makes the questionnaire function more extensible. In figure 5.3 the relation between answer sheet and question type is shown. *AnswersheetFragment* points to several other fragments. These fragments provide one question type each.

Depending on how big the questionnaire is, the *QuestionnaireActivity* must manage more or less fragments. For each question another fragment is added. This design allows to use one activity for the entire questionnaire.



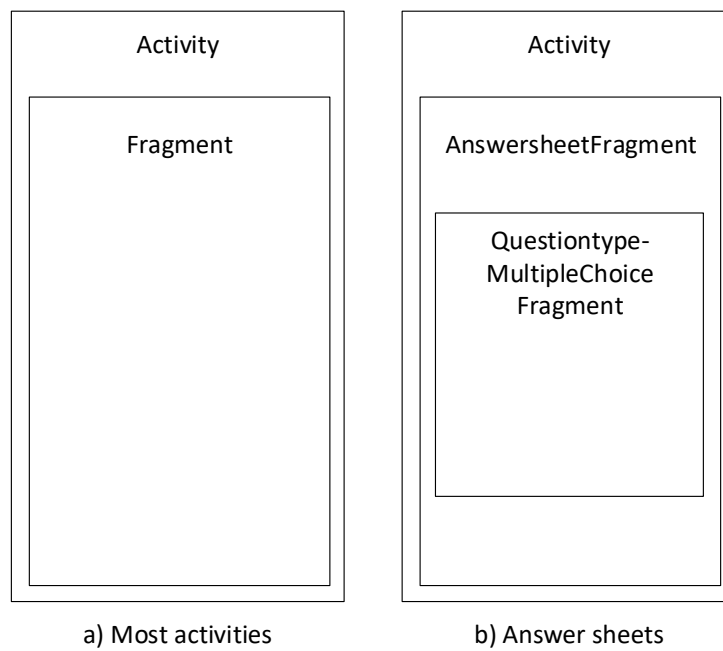


Figure 5.1: **Fragments within Activities:** This figure introduces how fragments are used within activities. In most activities one fragment is shown at any point in time (a). However, when answering questionnaires, an additional fragment is used to display the answers (b).

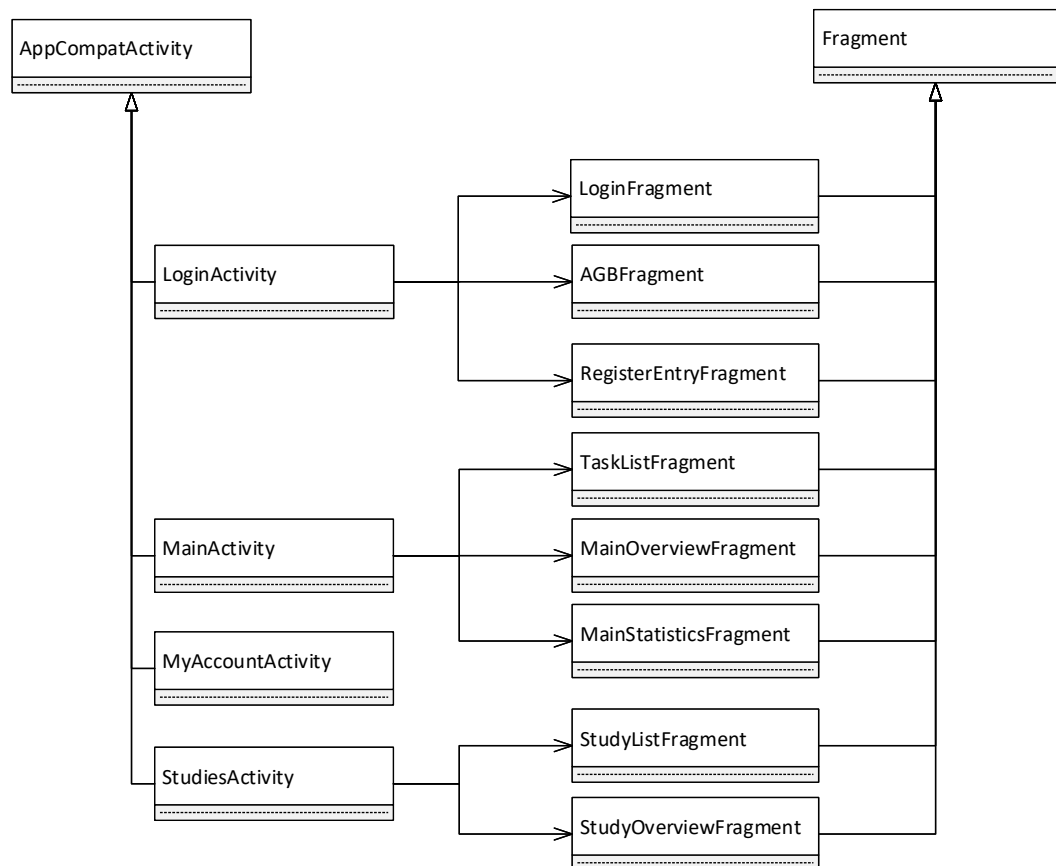


Figure 5.2: **Activities and Fragments:** This picture shows the application's activities and the associated fragments. This is the first part of a simplified UML. For the second part see 5.3

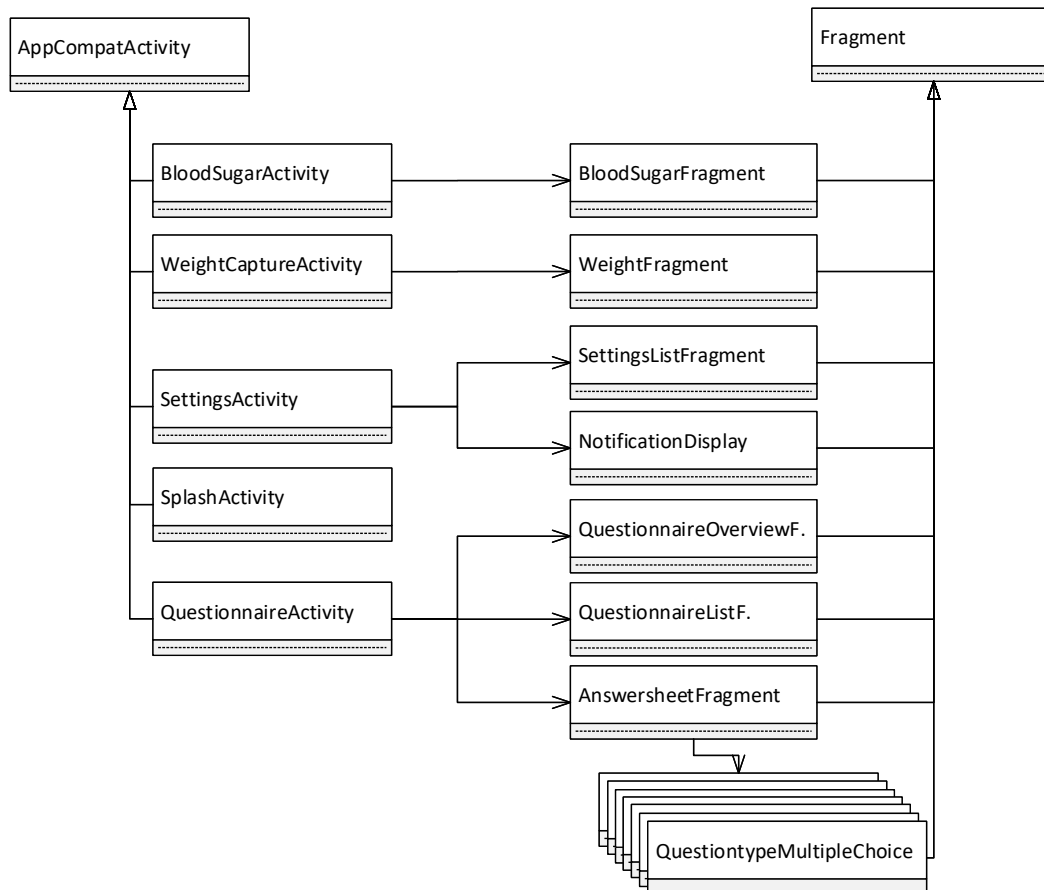


Figure 5.3: **Activities and Fragments:** This picture shows the activities and their fragments. This is the second part of a simplified UML. For the first part see 5.2

### 5.3 Application Process

One of the requirements presented in chapter 4 is that users need an account to use the application. Therefore, the user will be prompted to log in when he opens the application for the first time. If he does not have an account yet, he needs to create one. Even though it is a requirement for the application to work without an active internet connection, the log in screen will inevitably need one. Be it logging in, resetting the password, or creating a new account - all of these actions require contacting the backend server. If the device is not connected, the user will be informed. Then, he can retry at any time.

The user can also choose a language before logging in. This function was integrated into the login screen because, at this point of the process, no data has yet been downloaded. If the language could be changed after logging in, two scenarios would be possible:

- The device saves only one translation at any time. In this case, all data needs to be downloaded again. This requires internet connectivity. Each time the user switches the language, all data is downloaded anew. Hereby, resources of smartphone and backend server are not used efficiently.
- The language can easily be switched without needing internet connectivity. This is the case because all translations are already saved on the local device. For this to be possible, the application needs to make each request several times for each language. Moreover, each translation means additional data to be saved on the device.

Since both scenarios are not resource efficient, users can change the language only before logging in. This does not require additional memory or backend requests. After users click to sign in, only one language package will be downloaded. This might be best solution since, in daily use, it is unlikely for users to require switching the language. However, choosing a language at all should not be required in most cases. The default language will be the one set by the operating system. If the application does not have the respective language, the standard language is English. Hence, for most people the right language will be already preset. Therefore, users will not be prompted to choose

a language before logging in, but they can still change it, for example, if their mother language is not provided and they prefer rather German then English.

The general procedure, while using the application, is shown in figure 5.4. When the application is opened it first checks whether the user is logged in or not. If he is, he will be forwarded to the main screen where he can access the whole functionality of the application. In this case an internet connection is not required anymore. After login, the application remembers the user. This only changes when he explicitly signs off, or his data will be deleted for other reasons, e.g., when the user uninstalls and re-installs the application.

The user needs a working e-mail address to register. After his registration, he will be prompted to verify his address by following a link. He also needs this address if he forgets his password and needs to reset it.

The main screen is provided by the *MainActivity* class. As shown in figure 5.5 the main screen provides a navigation, for the user to access most of the functionality. Some of the functions presented in figure 5.5 are introduced shortly in the following:

- **View Statistics and Feedback:** The main screen has three tabs. On the first tab users can view their feedback. This functionality is important. Therefore, it is located on the main UI where it is difficult to overlook. On the third tab, he can view statistics about his tracked data on one page.
- **View Tasks:** The second tab of the main screen contains the user's tasks. He may need to track his blood sugar level or answer a questionnaire. These and all other tasks, he should complete over the day, are listed here. If he selects a task, he will be forwarded to the activity where he can complete it. After completion the task disappears from the list. However, if the user, for example, needs to track his blood glucose level three times a day, the task is not shown three times. A counter indicates how often a task needs to be completed. This counter is then decreased until it reaches zero. Then, the task disappears from the list.

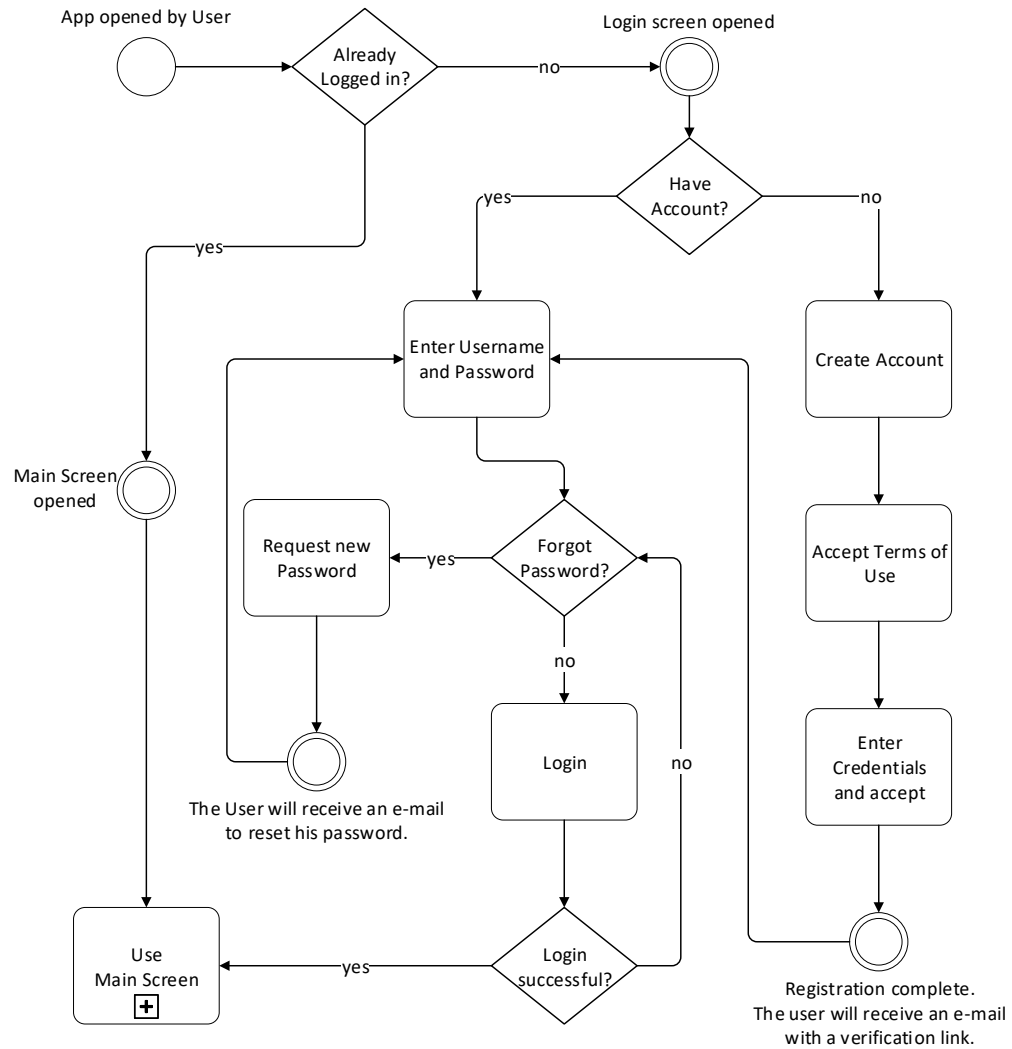


Figure 5.4: **Application Process:** This process shows, how the application is used. First the user needs to authenticate. Afterwards, he will be forwarded to the main screen where the whole functionality of the application can be accessed.

- **Manage Studies:** This includes viewing information of a study, subscribing, un-subscribing, viewing one's invitations, and accepting them. To subscribe to a new study, the device requires a connection to the internet. This is the case, because additional data needs to be downloaded from the backend server. Also, the user sometimes needs to authenticate by providing the subscription password. This also requires a connection to the backend.
- **Answer Questionnaires:** To answer a questionnaire the user can navigate to the questionnaire view, where all his questionnaires are shown. When selecting one, a description will be shown as well as its schedule. By clicking a button, the user can start answering the questions. He can also answer a questionnaire by selecting the respective task, as mentioned above.
- **Enter Weight and Blood Sugar Level:** For these functions two separate screens are provided. One for tracking the user's weight and one for his blood sugar level. On both screens a diagram of the past days will be shown to the user where he can see his progress.
- **Track Activities:** For tracking activities, a standard questionnaire is offered. It is not listed in the questionnaire section. The user can access this function via navigation menu.

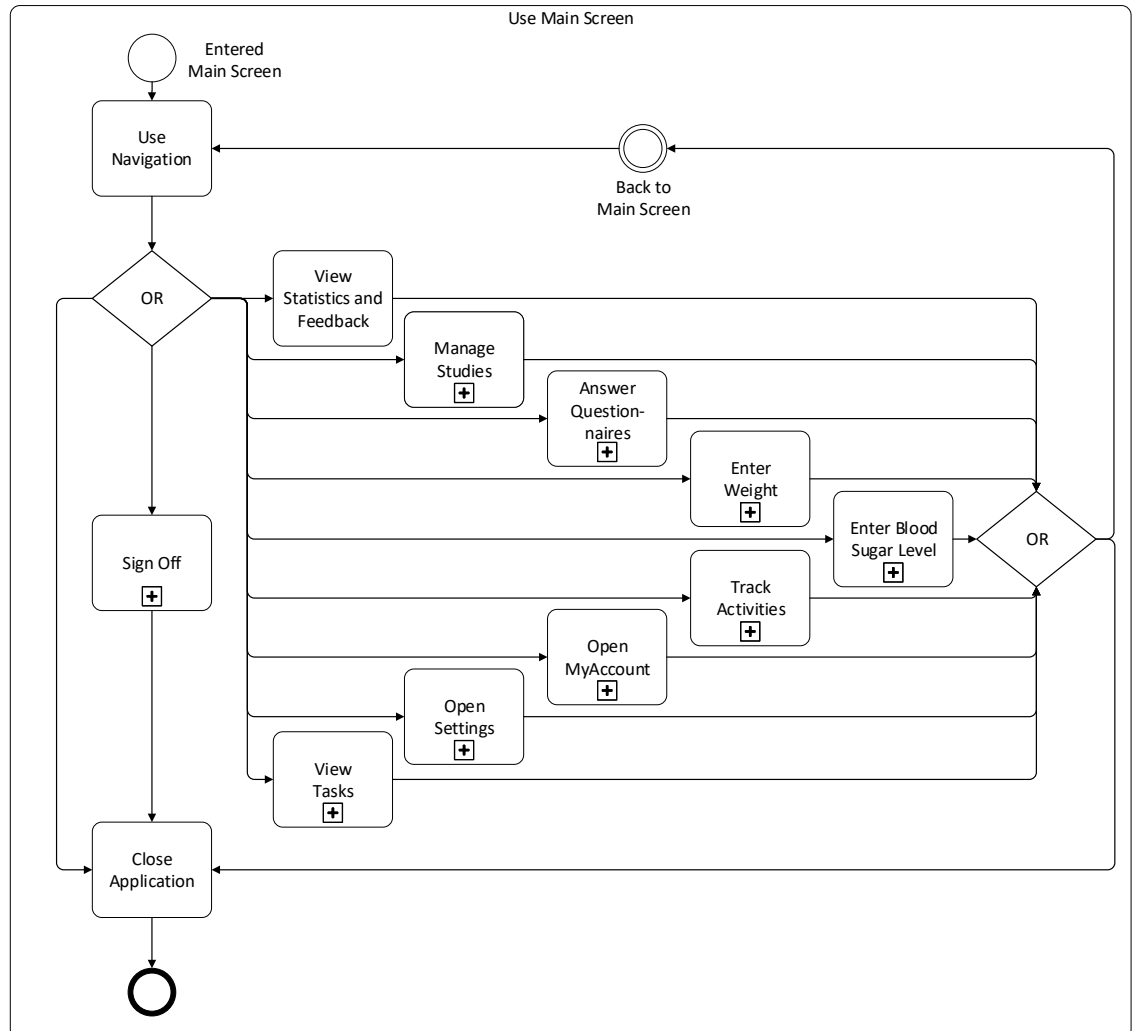


Figure 5.5: **Main Screen Process:** This is a sub process of figure 5.4. It shows how the user can access the functionality of the application. It is a simplified diagram since most of the functionality is presented as a sub process.



## 5.4 Local Data Model

Data downloaded from the server must be stored on the device. This is necessary, for the application must work without internet connection. Also, a local database enhances the speed of the application immensely and saves resources such as bandwidth and computation. Hence, a local storage model has been designed for the application.

Data is stored on the device using SQLite [55]. It is downloaded from the backend server after the user successfully logs in. This can take some time, but it is important for the application to download the data all at once. Otherwise, it cannot be guaranteed that the application works without internet. Hence, once the user logs in and is forwarded to the main screen, all necessary data is already stored on the device.

Figure 5.6 shows a part of the data model used. The tables were all realized using C# objects. These objects can be stored in the local database by SQLite. Most object attributes are named after their counterparts at the backend server (see section 5.6).

The table *Task* is not a backend entity. For each task the user has, a new task object is created. Afterwards, these objects are instantly saved in the database. As a result, they only need to be computed once. The questionnaire's schedule is decisive for the computation of tasks. Sometimes, this schedule can also be set by the user himself. When the user closes and opens the application, the tasks are fetched from the database, instead of being computed again.

Figure 5.7 shows another part of the data model. The two tables *BloodSugarMeasurement* and *WeightMeasurement* are data structures, in which the application saves the blood sugar level and the weight, submitted by the user. Data of the table *Invitations*, on the other hand, is retrieved from the backend server. For each invitation the user has, one entry is added. These entries are fetched when the user wants to view his invitations. A separate class provides all required methods to fetch the data and return it in the form of C# objects.

## 5 Architecture

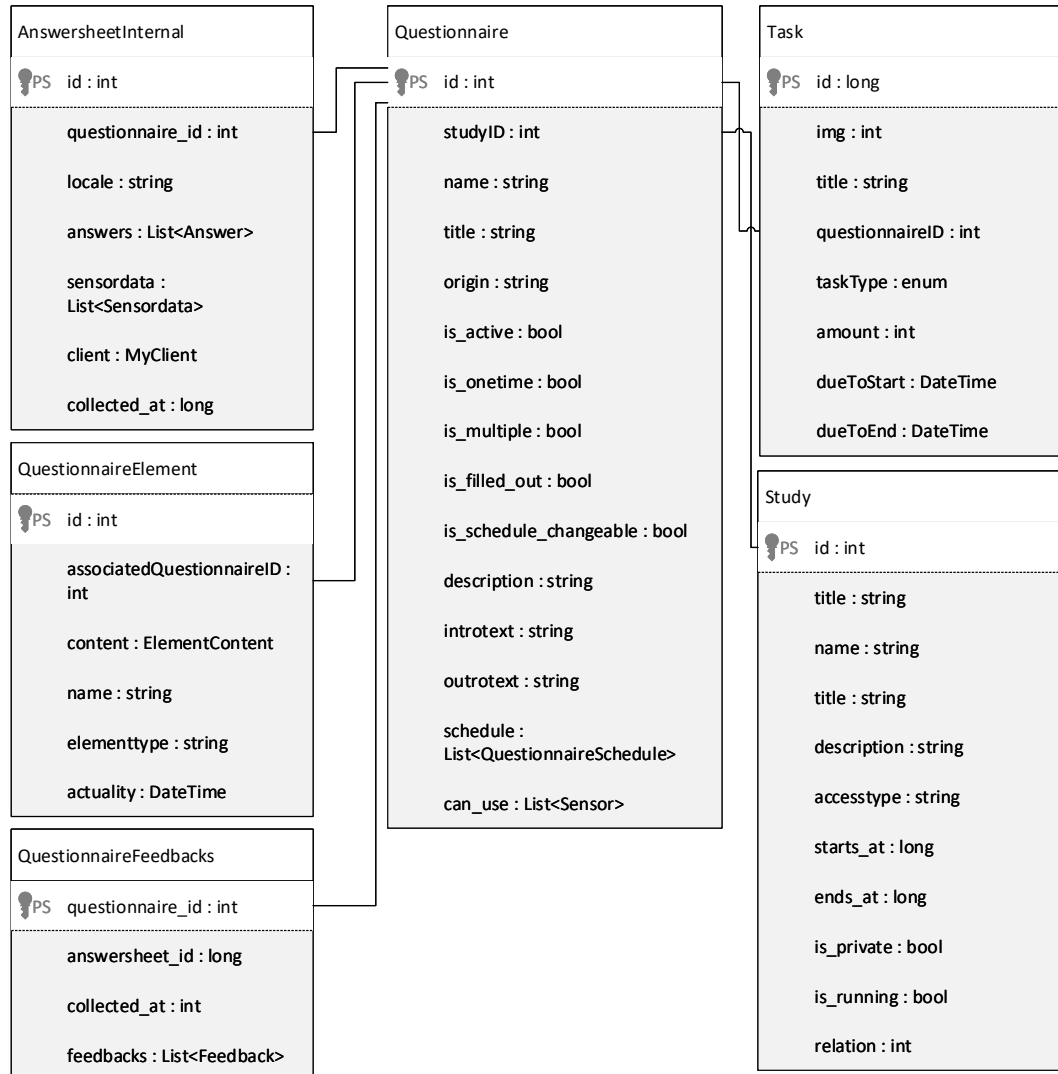


Figure 5.6: **Data model** (part 1): This is one section of the tables used to store the relevant data on the device. *Task* is a local object. Except for *Task*, all other tables in this figure are modeled after the objects the backend server sends.

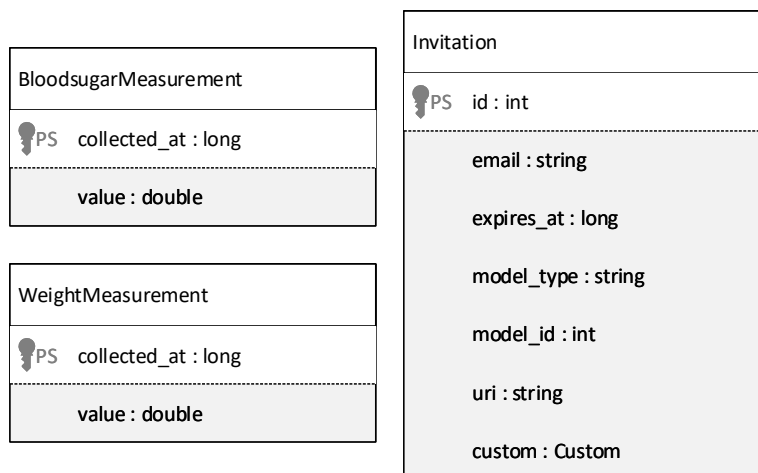


Figure 5.7: **Data model** (part 2): This is another part of the tables used to store the relevant data. Here *Invitation* is modeled after the data received from the backend. The two remaining tables are only available on the local device.

## 5.5 Background Threads

While the application is used, a thread runs in the background. After login, this thread is responsible for most communication between the device and the backend server.

It positively affects the usability of the application if the backend communication is done in the background. This way, the user does not have to wait for data exchange nor be bothered by messages concerning the internet connection either. If, at the present time, no internet connectivity is available, the background thread queues the request and retries later. The thread has two queues:

- **Retrieve queue:** This queue contains tasks for updating data stored on the device, e.g., studies, invitations, and questionnaires. These tasks keep the database up to date and are completed once a day.
- **Deliver queue:** Here, requests are queued for uploading the user's data. For example, if the user fills out a questionnaire, the answers need to be submitted to the backend. Since the application must work without internet, a task is produced

and queued here. As soon as the device has internet connectivity, the queued data will be sent to the server.

All of these tasks are also stored in the database. After a task is completed, the database-entry is deleted. These database-entries are important since the user can forcefully close the application at any time. By that, the application context is deleted together with outstanding tasks. For these cases, the outstanding tasks need to be backed up. Hence, when the application is opened again the outstanding tasks can be fetched from the database and enqueued again. This way, data does not get lost.

The background thread is designed to prioritize deliver tasks higher than retrieve tasks. This is the case since retrieve tasks request data that changes rarely and cannot get lost. Those tasks are mainly used to keep the database up to date. Delivery tasks, on the other hand, contain user data that could get lost. Hence, it should be sent to the backend as soon as possible. An example is if the user forcefully signs off while he does not have a working internet connection. Then, his delivery data, queued in the background thread, will be deleted. The answers of a questionnaire could be a part of this data. Afterwards, when the user logs in again, he would have to fill out this questionnaire once more. The aim of prioritizing the delivery queue is to make such cases as unlikely as possible.

### 5.6 Restful-API Back End

The backend server delivers all important data and stores the users' input. It is a RESTful-API programmed and maintained by Johannes Schobel. It is written in PHP and uses a MySQL database. In the following, the interface of the backend server, used by the application, will be introduced.

To contact the backend server, an HTTPS request is sent by the application to the respective URL. Depending on which function is requested the proper HTTP request method needs to be used, e.g., GET or POST. Additional data must be sent to the server within the request body formatted in JSON. The server sends its data formatted in JSON,

also.

The API documentation provides information about the various methods and how to invoke them. It lists all methods and delivers the following information:

- **URI path:** Each method has a unique combination of a URI path and an HTTP request method. For example, the URI path for registering is: `/api/v1/auth/register` [47].
- **HTTP method:** Various HTTP methods are used, e.g., *GET*, *POST*, *DELETE* and *PATCH*. *GET* is used for requests without request body such as "Get a Questionnaire" [47]. For others, a request body is required since they either want to send data to the backend (*POST*) or change existing data (*PATCH*).
- **Authentication:** Mostly, backend methods can only be invoked when authentication is provided. An authentication token (xyz) can be requested by using the login method. Then, this token must be added to the request URI by adding `?token=xyz`. When the backend receives this request, first, the token is checked. If the check succeeds, the client receives the requested data. If not, he receives an exception.
- **Request Body:** As mentioned before, some requests contain a request body formatted in JSON. The documentation informs how the respective JSON object is structured.
- **HTTP Status:** If something goes wrong, e.g., when the URI is wrong, the authentication fails, or the request body is wrongly formatted, a negative HTTP status message will be sent to the client. If everything went well, the client receives a positive HTTP status message.



# 6

## Selected Implementation Aspects

This chapter presents the implementation of some parts of the application. First, it is shown how notifications are implemented. Afterwards, it is introduced how threads are used in this application. Next, the backend communication is presented with an example. Finally, the implementation for both, questionnaire management and basic functionality, is introduced.

### 6.1 Notifications

Questionnaires can have schedules, and notifications are used to enforce these schedules. This way, the user does not have to worry about forgetting to fill out a questionnaire in time. If the user is logged in, he does not even have to open the application to check whether he needs to fill out questionnaires. He can rely on the notification function to remind him. That is why it is advised to the heads of studies to determine a schedule for their questionnaires. When a notification appears, and the user clicks it, he will be forwarded to the respective questionnaire or activity.

Notifications are local and are computed directly after login. For that, the schedule of each questionnaire is evaluated, and a proper notification set up.

To schedule a notification, first, a *NotificationTask* object is created for each. Figure 6.1 shows the attributes of the said object. These attributes stand for the following:

- **schedule:** This variable determines the exact date and time.

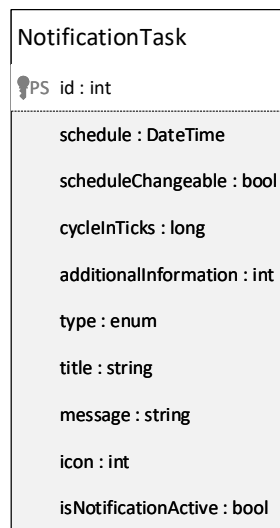


Figure 6.1: **NotificationTask**: This picture shows the data structure for a notification task. It is a database table and used to handle all notifications.

- **scheduleChangeable**: If the study director allows for the schedule to be changed by the user, this variable is set to true.
- **cycleInTicks**: Some questionnaires need to be answered regularly. For example, if a questionnaire needs to be filled out every week, a seven-day cycle is determined by this variable.
- **type, additionalInformation**: There are different types of notifications. For example, a notification can remember the user to track his blood sugar level or to fill out a questionnaire. With the attribute *Type* the program can differentiate between all different notification types, and the *NotificationTask* object can therefore be used for all notifications. Thereby, the notification model is more extensible. If another kind of notification is needed in the future, a new type can be declared. If the notification references a questionnaire, this information is saved in *additionalInformation*.
- **title, message, icon**: These variables contain the text and the reference to the image shown by the notification in the UI.



- **isNotificationActive:** If the notification is already active, this variable is set to true.

This information will be saved in the database for further handling of all notifications, e.g., if the user wants to view the notifications of a questionnaire. In most cases, if a NotificationTask needs to be fetched from the database, it is searched by providing the attributes *type* and *additionalInformation*.

Finally, the NotificationTask objects can be used to schedule the respective notifications. Listing 6.1 shows a code excerpt of the scheduling process. First, an intent is created (myIntent). This intent receives all information needed, such as the notification title, message, and icon. The type of the intent is *NotificationScheduler*, which inherits from Android's BroadcastReceiver. This is the class that will ultimately build the notification.

Then a PendingIntent [56] is initialized that is later used to schedule the notification. Considering, that more than one notification might be scheduled, the ability it is required to distinguish between several PendingIntents. This is important for various reasons. For example, if the user wants to deactivate a notification. In this case, the PendingIntent needs to be found first. For this reason, the request code of the PendingIntent is set to be the id of the NotificationTask (see figure 6.1). This id is assigned by the database, is auto incremental, and unique.

The AlarmManager [57] is used to schedule notifications at a specific point in time. To determine the correct time, the method *SystemClock.ElapsedRealtime()* is used, which returns "the time since the system was booted" [58]. Then, a previously calculated number of milliseconds is added, and thereby the correct time for the notification is set.

---

```

1 int requestcode = task.id;
2 Intent myIntent = new Intent(context, typeof(NotificationScheduler));
3 myIntent.PutExtra("TITLE", task.title); (...)
4 PendingIntent pendingIntent = PendingIntent.GetBroadcast(context,
    requestcode, myIntent, 0);

```

## 6 Selected Implementation Aspects

```
5 DateTime time = task.schedule;
6 long time_difference = (time.Ticks - DateTime.Now.Ticks) /
    TimeSpan.TicksPerMillisecond;
7 AlarmManager manager =
    (AlarmManager) context.GetService(Context.AlarmService);
8 if (task.cycleInTicks == -1)
9 {
10     manager.Set(AlarmType.ElapsedRealtimeWakeup,
        SystemClock.ElapsedRealtime() + time_difference,
        pendingIntent);
11 }
12 else
13 {
14     manager.SetRepeating(AlarmType.ElapsedRealtimeWakeup,
        SystemClock.ElapsedRealtime() + time_difference,
        task.cycleInTicks, pendingIntent);
15 }
```

---

Listing 6.1: This is a code excerpt of the method that schedules notifications. Notifications are represented by a *NotificationTask* object (here called *task*).

## 6.2 Threads

In chapter 5.5 one use of background threads in this application was introduced. In the following an insight into the implementation will be given.

Background threads are used all over the application. This is done every time something should be processed in the background, so that the user interface can still be used. An example for that is the login process. While logging in the user interface should not freeze since it would seem like the application could have crashed. In this case, it is more appropriate to show a progress bar. The user sees that it is moving, and a message tells him that the login process is still being executed. Once the login process is done, the thread downloading data in the background is merged with the main thread again.

Listing 6.2 shows the method called after the login button is clicked.

---

```

1 private void bLoginClick(object sender, System.EventArgs e)
2 {
3     progress = new ProgressDialog(thisActivity);
4     progress.Indeterminate = true;
5     progress.SetProgressStyle(ProgressDialogStyle.Spinner);
6     progress.SetMessage("Contacting server. Please wait...");
7     progress.SetCancelable(false);
8     progress.Show();
9
10    ThreadPool.QueueUserWorkItem(o => doLogin());
11 }

```

---

Listing 6.2: An example for outsourcing code into a background thread.

First, the progress bar is set up and started. Then the method *doLogin()* is called as a background thread. Listing 6.3 shows how the method *doLogin()* lets the thread return to the main thread. For that, it calls *startMainActivity()* by using Android's method *RunOnUiThread()*. Then, *startMainActivity()* deactivates the progress bar and fulfills its purpose. For this to work, the progress bar object needs to either be global or passed on from method to method. In this case, it is a global object within the class.

---

```

1 public void doLogin() {
2     ...
3     this.Activity.RunOnUiThread(() => startMainActivity());
4     ...
5 }
6 public void startMainActivity()
7 {
8     if (progress.IsShowing) progress.Dismiss();
9     thisActivity.startMainActivity();
10 }

```

---

Listing 6.3: A code excerpt that merges the background thread with the main thread.

## 6 Selected Implementation Aspects

Mostly, this kind of threads are used for a short period of time. The thread introduced in chapter 5.5, which handles the backend communication, on the other hand, runs as long as the user is logged in. However, it only runs while the application also is running. This thread sends backend requests or saves them until the device is connected to the internet. It is initialized every time the *MainActivity* is, e.g., when the application is started, and the user is logged in. Listing 6.4 shows the code excerpt to start this thread. First, a cancellation token is initialized. With this token the thread can be stopped by the *MainActivity*. This needs to be done in the sign-off process, for example. Then, the respective thread class is initialized, and finally the thread is queued. Starting this thread is done in a similar way to the threads in the example before. Except that the method *Work()* will only run if the cancellation token allows it.

---

```
1  cts = new CancellationTokenSource();
2  Threaded.InitializeThreaded(this);
3  ThreadPool.QueueUserWorkItem(tok =>
4  {
5      CancellationToken cancelToken = (CancellationToken)tok;
6      while (!cancelToken.IsCancellationRequested)
7      {
8          Threaded.Work(cts.Token, this);
9      }
10 }, cts.Token);
```

---

Listing 6.4: This code excerpt starts the thread for the backend communication.

The thread checks whether a new delivery or retrieve task is waiting in the queues to be executed. When accessing these queues, mutual exclusion is used to prevent data inconsistency. This is necessary since the user interface and the background thread run at the same time. When the user completes a questionnaire the creation of a delivery task is triggered. Therefore, the delivery queue must be accessed. If, at the same time, this queue is accessed by the background thread, data inconsistency can happen, due to concurrent access. In listing 6.5 it is shown how mutual exclusion works in Xamarin. The object *deliverQueueLocker*, used for the mutual exclusion, needs to be global so it

can be accessed from everywhere.

---

```

1 lock (deliverQueueLocker)
2 {
3     if (deliverQueue.Count > 0)
4         task = deliverQueue.Dequeue();
5 }

```

---

Listing 6.5: Code excerpt for accessing a queue with mutual exclusion.

As already mentioned before, delivery queue tasks aim to send data to the backend server, e.g., the answers of questionnaires. These tasks are stored in a uniform data structure called *DeliveryTask*. Listing 6.6 shows this data structure.

---

```

1 public class DeliveryTask
2 {
3     [PrimaryKey, AutoIncrement]
4     public long id { get; set; }
5     public string path { get; set; }
6     public string input { get; set; }
7     public BackendMethods method { get; set; }
8     public bool needAuthentication { get; set; }
9     public DeliveryItem type { get; set; }
10    public int additionalInformation { get; set; }
11 }

```

---

Listing 6.6: This is an excerpt of the *DeliveryTask* class which is used to uniformly store delivery tasks.

The attributes of *DeliveryTask* are used for the following:

- **id:** Delivery tasks are stored in the database until they are processed. For that a unique identifier is required. The two keywords *PrimaryKey* and *AutoIncrement* are SQLite references.
- **path:** This is the path for the backend request.

## 6 Selected Implementation Aspects

- **input:** Here the JSON data body is saved.
- **method:** This attribute defines which HTTP method needs to be used.
- **type:** With this the delivery task type is set, e.g., *answer sheet*.
- **additionalInformation:** If needed, this variable can be used for additional information, e.g., the questionnaire id. This can vary for different delivery task types.

The data structure for retrieve tasks is implemented similarly. This design aims to provide a clearer structure and to make the implementation extensible so that new task types can be added in the future.

### 6.3 Backend Communication

The backend communication is done on two layers. On the first layer, the request path is chosen, and the data is formatted to meet backend requirements. Mainly, this layer is represented by the *DataRetreive* class. The second layer handles HTTP requests. It is represented by the class *BackendCommunication*. This class initiates HTTP requests and returns the responses.

In the following the said backend communication is introduced. It will be shown, how the studies are retrieved from the server. First, the backend path for downloading all available studies needs to be determined. The API documentation [47] provides this information: `/api/v1/studies`.

The documentation also specifies, that this request requires authentication. In this application, the token is saved using *SharedPreferences* [59]. In Xamarin this class is called *ISharedPreferences* [60], and has - with few changes - basically the same functionality. First, the token (`xyz`) is retrieved. Then, the URL path is completed by adding `?token=xyz`.

After the complete request path is available, the data body needs to be prepared.

It needs to be sent to the backend server as an object formatted in JSON.

In this implementation Json.NET, a Newtonsoft library, is used [61]. With this library it is possible to serialize a C# object into the JSON notation and de-serialize it back into a C# object. Listing 6.7 shows how the serialization is done. In this case, settings are used for the serialization. This is an optional step. Here, they are used because the object contains attributes, which are unknown to the backend server. They should not be sent, and to suppress them they are set to null. These serialization settings make sure those attributes are ignored. If this is not done, the backend server returns an exception, saying that the request body is not processable.

---

```

1 JsonSerializerSettings settings = new JsonSerializerSettings();
2 settings.NullValueHandling = NullValueHandling.Ignore;
3 string jsonData = JsonConvert.SerializeObject(bObj, settings);

```

---

Listing 6.7: **Json.NET serialization:** This is an example for serializing a C# object (bObj) into a JSON formatted string.

To request all studies the HTTP method GET is used. Hence, a request body is not needed and is therefore an empty string. When the request path and body are available the request can be executed by the second layer. For this purpose, the proper method in the *BackendCommunication* class is used. Listing 6.8 shows the required code for sending an HTTP-GET request.

---

```

1 client.DefaultRequestHeaders.AcceptLanguage.Add(new
    StringWithQualityHeaderValue(LANGUAGE_SET));
2 Uri requestUrl = new Uri(mUrl + path);
3 HttpResponseMessage response = client.GetAsync(requestUrl).Result;

```

---

Listing 6.8: **HTTP request:** This is a code snippet taken from the `GET()` method of the *BackendCommunication* class. First, the language is set. Then, the URI is created and finally the request is sent by using the method *GetAsync(URI)* from the HttpClient library.

## 6 Selected Implementation Aspects

The server reply is retrieved by `client.GetAsync(requestUrl).Result` (see listing 6.8). Now, the *response*-object contains the required data, from which the status message and the response body are extracted. The *BackendCommunication* class only handles server-side exceptions and exceptions that happen while attempting to connect. All client errors are handled by the *DataRetreive* class. This is done for two reasons. The first is to strictly separate the HTTP requests and the application logic. Secondly, because error codes can have different causes in different cases. For example, the status code 400 can say that the authentication token is expired in one case, while in the other that no user can be found [47].

A de-serialization method converts the JSON objects, received from the server, into C# objects. For that, a class must be provided, which is structured the same way the JSON object is. Listing 6.9 shows a code example for de-serializing a JSON string into an object. The object *bObj* is consists of several generic objects nested in one another. This structure is somewhat complicated, but it has to fit the object structure of the JSON string received by the backend.

---

```
1 public class DeliveryTask
2 List<DataBody<Study>> dataBody = new List<DataBody<Study>>();
3 BackendObject<List<DataBody<Study>>> bObj = new
    BackendObject<List<DataBody<Study>>>(dataBody);
4 bObj = JsonConvert.
5   DeserializeObject<BackendObject<List<DataBody<Study>>>>(response.data);
```

---

Listing 6.9: **Json.NET de-serialization:** This is an example for de-serializing a JSON formatted string into a C# object.

Now, the studies can be extracted from the data structure (*bObj*) and saved into the database.



## 6.4 Questionnaire Management

The class *QuestionnaireActivity* is used to organize questionnaires. The layout file for this activity is composed of a Toolbar and a *FrameLayout* within a *LinearLayout*. In this activity the user interface needs to be changed constantly. Therefore, it is managed by fragments occupying the whole screen. Three fragment types are used:

- ***QuestionnaireListFragment***: This fragment lists all questionnaires the user has.
- ***QuestionnaireOverviewFragment***: Here the information of one questionnaire is shown, e.g., the description, the schedule, and whether it is a one-time questionnaire or not.
- ***QuestionnaireAnswersheetFragment***: This fragment shows questions and gives users the possibility to answer them.

The layout of *QuestionnaireListFragment* is composed of a *ListView* within a *LinearLayout*. This means, the whole fragment is a single listing of items. To fill such a list with content, an *Adapter* is needed. Here, the adapter class is called *QuestionnaireListAdapter* and it inherits from Android's *BaseAdapter* class. It expects a list of questionnaires as argument. For each entry of this list a new row is created. These rows also need a layout file to specify the user interface. Figure 6.2 shows how these rows are structured.

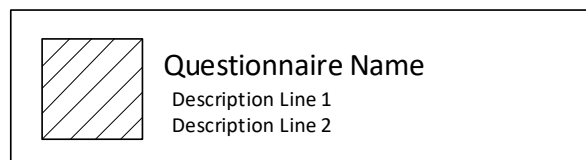


Figure 6.2: **List row**: This image shows the layout of list rows in the *QuestionnaireListFragment*. On the left, a picture is shown. To the right of the picture, the name and description of the questionnaire are presented by *TextViews*.

If users click on one of these items, the *QuestionnaireListFragment* gets covered by the *QuestionnaireOverviewFragment*. However, the old fragment still exists since the user can access it again with the *back* button. The fragment could also be destroyed and

## 6 Selected Implementation Aspects

recomputed if it is needed again, but it would be less efficient. Instead all fragments are saved and ordered by sequence. In addition, the fragment currently displayed is saved in a global variable within the activity.

The activity acts as a communication channel between these fragments. For example, if a questionnaire is clicked in *QuestionnaireListFragment*, a global variable, within the activity, points to this questionnaire. If the user decides to answer the questionnaire, *QuestionnaireAnswersheetFragments* need to be created to show the questions. The *QuestionnaireOverviewFragment* retrieves the questionnaire id from the global variable where the clicked questionnaire was saved. Hence, the activity provides shared memory for the fragments so, they can communicate with each other.

To answer the questionnaire, the questions are retrieved from the database, first. For each question, a new fragment is created. The questionnaire activity has a counter to point to the question currently displayed. After the user has answered the question, he clicks on a button to show the next one. For that, the fragment uses a method called *nextQuestion()*, provided by the *QuestionnaireActivity*. Listing 6.10 shows how this method can change the user interface.

---

```
1 internal void nextQuestion(V4Fragment caller)
2 {
3     Fragment nextFragment = questions[questionNo++];
4     var transaction = FragmentManager.BeginTransaction();
5     transaction.Add(Resource.Id.fragment_container,
6         nextFragment, "Question" + questionNo);
7     transaction.Hide(caller);
8     transaction.Commit();
9 }
```

---

Listing 6.10: **Show another fragment:** This method shows how to hide the current fragment and show another one. For this, a *FragmentManager* is initialized which provides all methods required.

When the user is done answering the questions an answer sheet object is created. Also, a `DeliveryTask` is created, so the background thread can send the answers to the backend server (see chapter 6.2).

## 6.5 Basic Functionality

Tracking blood glucose levels and weight is basic for diabetes therapy. For this reason, these functions were implemented independently from studies and questionnaires. The objective was, to be able to track this data even if a questionnaire is not provided. Furthermore, the user should be able to access this functions without searching for the respective questionnaire first. For those reasons, separate activities were built for tracking the blood glucose level and weight.

Therefore, if a questionnaire does not exist, the patients may still track this information and view it in a diagram. In this case, however, the data will not be saved by the backend server. Nonetheless, a standard study was created on the server for this purpose. It contains a questionnaire for the blood sugar level, the weight and physical activity. If the user tracks his blood sugar level, the application tests whether a proper questionnaire exists. If it does, the data is sent to the backend. If not, it is only saved locally.

## 6.6 Used Technologies and Frameworks

The application was implemented using Microsoft Visual Studio Enterprise 2017. The .NET Framework's version was 4.7.02046. The version of Xamarin that was used was 4.5.0.486 with the Xamarin.Android SDK 7.3.1.2.



# 7

## Introducing the Application

This chapter will introduce the application by showing screenshots and explaining how the user interface works.

### 7.1 Login

Users can only access the application with a valid user account. After opening the application for the first time, the screen in 7.1a will be shown to the user. There, he can decide whether he wants to log in, create a new account or change his password.

By clicking the menu icon in the toolbar, a language can be chosen. Figure 7.1b shows the languages available. The default language is the one set by the Android OS. If the respective language is not provided, English is the default.

To register, the user needs a valid e-mail address. After he clicks on *Create Account*, he receives an e-mail with a verification link. Only when his e-mail address is verified, he can use it to log in. Also, a username and password need to be specified for registration. Figure 7.1c shows a screenshot of the registration process.

## 7 Introducing the Application

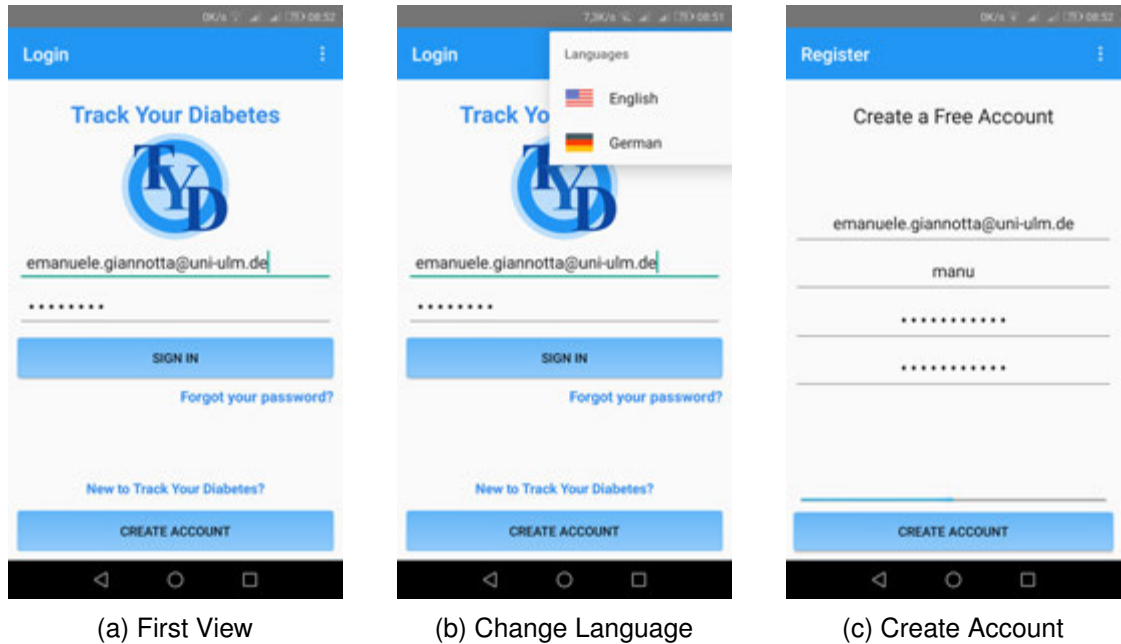


Figure 7.1: **Login and Register:** These screenshots show the user interfaces for logging in, changing the language and registering.

## 7.2 Main Page

After logging in, the main screen is shown. This screen is depicted in figure 7.2a. The main screen is composed of three tabs. The overview tab lists information that can be interesting to the user. The first in the list is a notification for an incoming invitation. If no invitations are available, this item does not show up. The second one is feedback for the user. In this case, it informs him that his blood glucose level is too high. This feedback is based on his last measurement, which can also be seen below in the list.

The listing also shows how many of tasks the user has. These tasks can be viewed in detail in the second tab (see 7.2b). There, all tasks are listed, and their schedule is shown. The first task, for example, is for tracking the physical activity. If the user clicks it, he will be lead to the respective questionnaire. The number in each task is a counter. It varies depending on how often a task needs to be completed. In this case, for example, the blood sugar level needs to be tracked three times. When it is tracked, the counter is reduced by one until it reaches zero. Then, the task disappears.

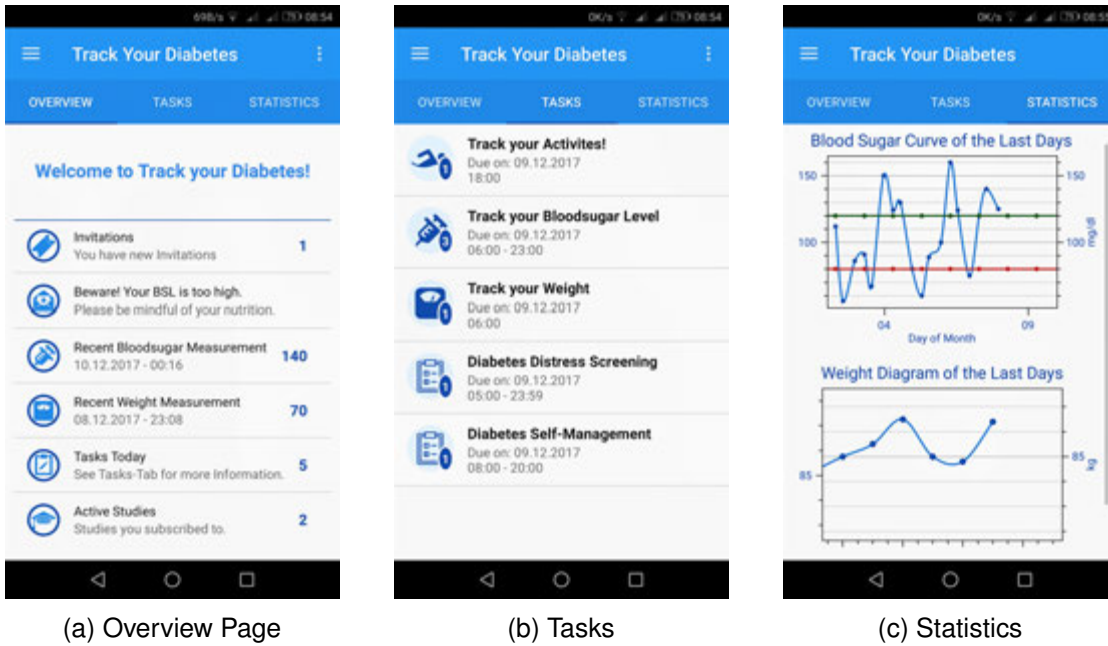
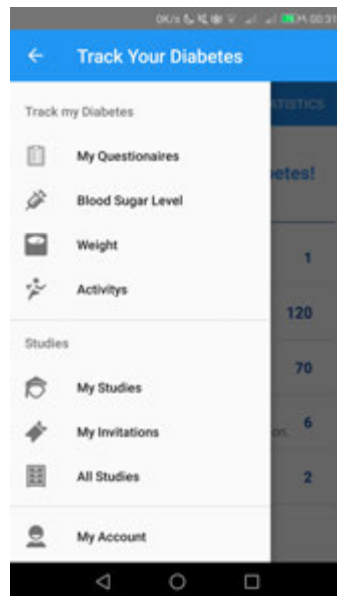


Figure 7.2: **Main Screen:** These screenshots show the three tabs of the main screen.

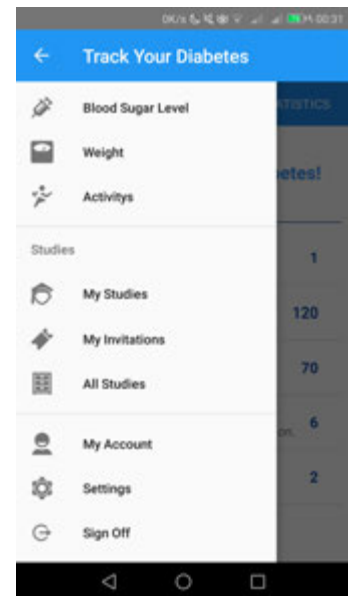
Finally, in the third tab, the user can see statistics of his treatment process. The screenshot 7.2c shows two diagrams. The first one shows the course of the blood glucose level. The other two lines indicate the range that should not be overstepped. In the future, additional information can be listed here. This way, the user has a summary about his treatment.

On the main screen the whole functionality can be accessed. This is achieved by a navigation menu. This menu is shown in figure 7.3.

## 7 Introducing the Application



(a) Navigation Menu (part 1)



(b) Navigation Menu (part 2)

Figure 7.3: **Navigation Menu:** These figures present the navigation menu by which nearly all features of the app can be accessed.

### 7.3 Studies

Figure 7.4a shows a screenshot of the frame "All Studies". There, all running studies are shown to the user. This includes those the user already subscribed to. Studies marked with a green checkmark are subscribed. When marked with a red lock, they are unsubscribed and require either a password or an invite to subscribe. All unmarked studies are free to subscribed to without restrictions. In 7.4b a view is shown where only subscribed studies are listed. Figure 7.4c lists all invitations.

After clicking a study in figure 7.4 an overview screen is presented (see figure 7.5). This screen shows information about the study, such as the description, the status, or the accessibility. When a study is password protected, the user will be prompted to insert it if he clicks "Subscribe" (figure 7.5c).



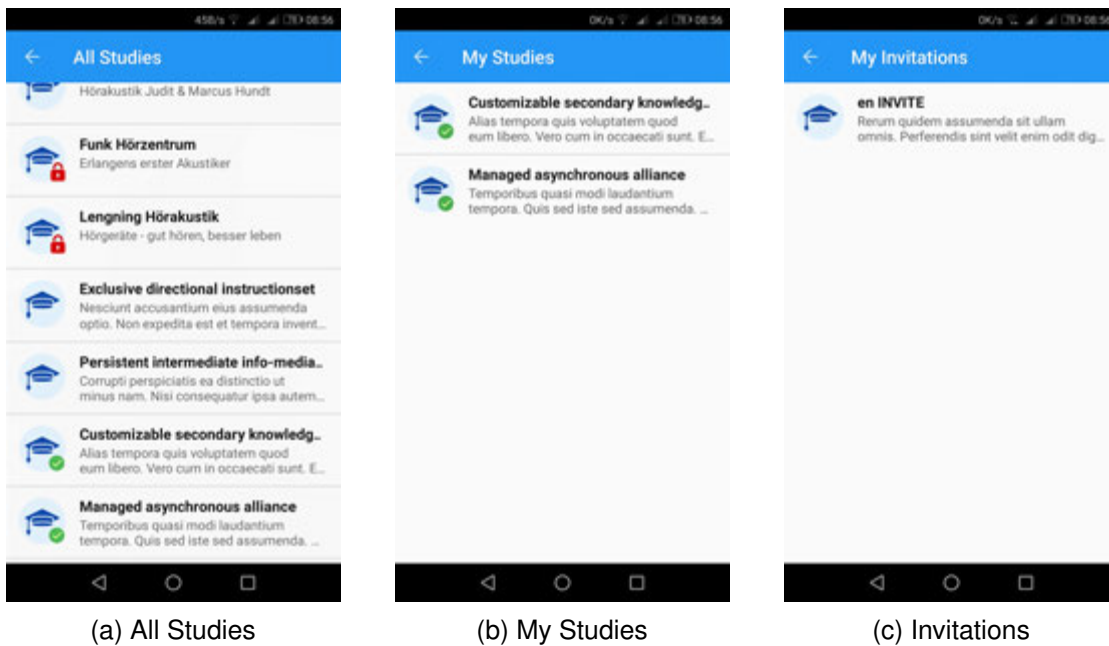


Figure 7.4: **List of Studies:** These images show how studies are listed. These studies are fictional and are only used for testing.

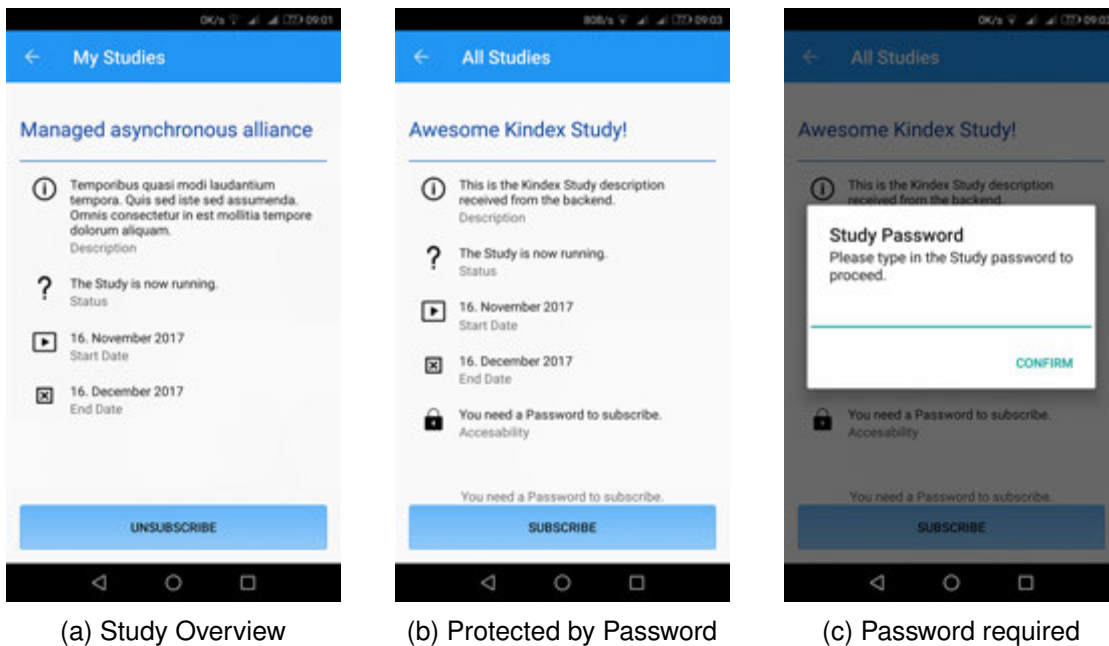
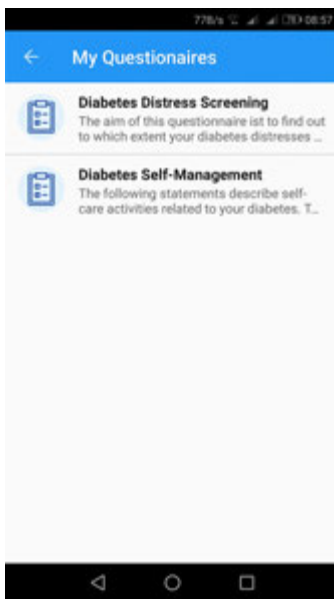


Figure 7.5: **Study overview:** These screenshots show the overview to studies. To reach these screens the respective study must be clicked in the user interfaces shown in figure 7.4

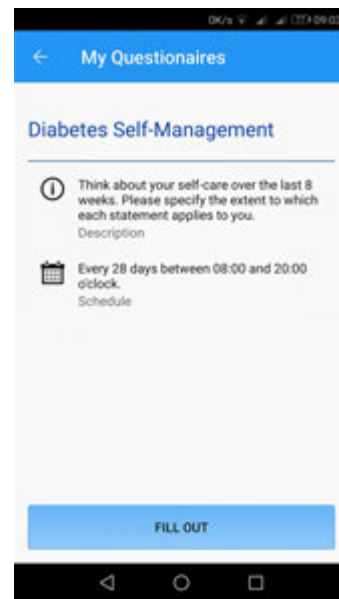
## 7.4 Questionnaires

Questionnaires are listed similarly to studies (figure 7.6). *My Questionnaires* (figure 7.6a) lists only those the user can fill out. It is a requirement not to show one-time questionnaires when they are already filled out. In figure 7.6b the overview of a questionnaire is depicted. It presents information, such as the description or the schedule. The user can start answering the questionnaire by clicking the button "Fill Out".

If he does, the questions will be displayed individually. Figure 7.7 shows some questions.

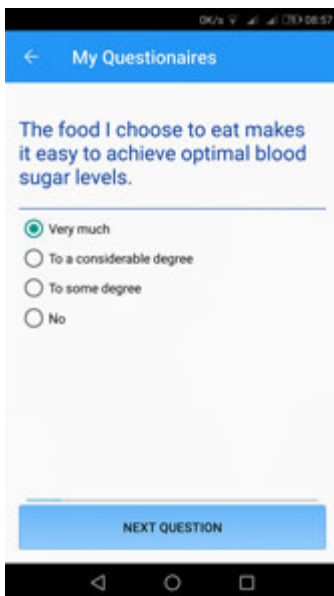


(a) My Questionnaires

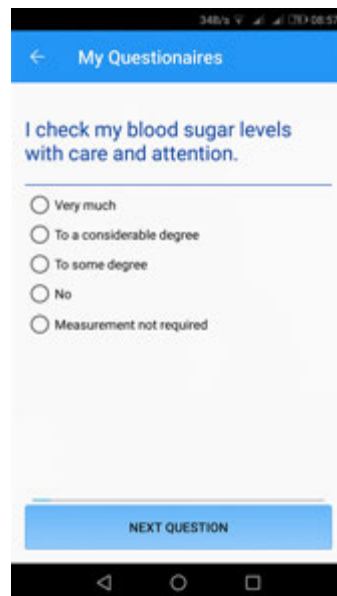


(b) Overview

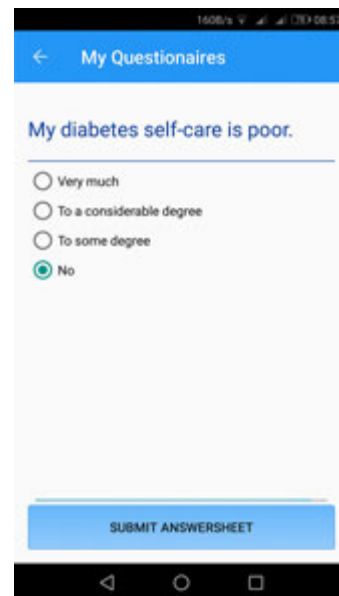
Figure 7.6: **Questionnaire listing:** These images show how the user's questionnaires are listed and how information is displayed.



(a) Question 1



(b) Question 2



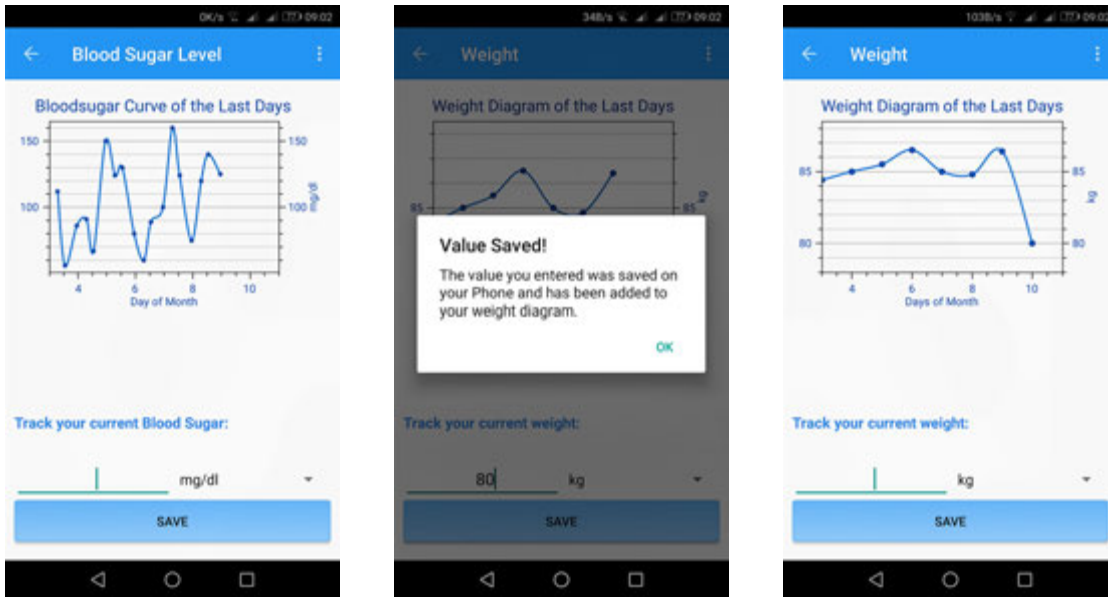
(c) Last Question

Figure 7.7: **Answer sheets:** These screens illustrate single questions and their possible answers. If the last question is reached, the "Next Question" button becomes a "Submit" button.

## **7.5 Basic Diabetes Tracking Functions**

Figure 7.8 shows how the user can track his blood sugar level and weight. The past entries can be viewed within a diagram. Next to the field, in which the user can enter the current value, he can also choose the unit. This will set the unit globally.

Apart of the blood sugar level and weight, the user can also track his physical activities. However, this is done by a questionnaire and is similar to figure 7.7.



(a) Blood Sugar tracking

(b) Track Weight

(c) New Weight Diagram

Figure 7.8: **Blood Sugar Level and Weight:** The screenshots (a), (b) and (c) show how the user can track his blood sugar level and weight.

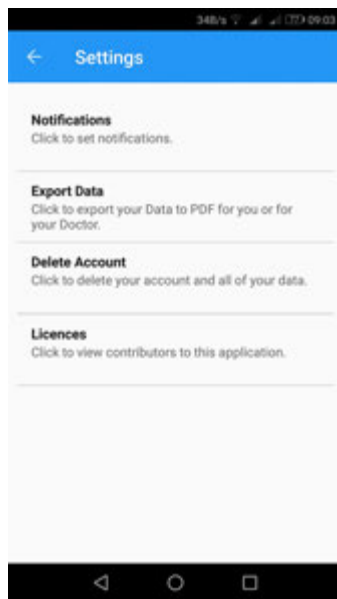
## 7.6 Settings

Figure 7.9a shows the settings screen. Users can set their notifications, export their data and delete their account. In *Licenses* some links are listed, such as the website where some of the icons, used in the application, originate [62].

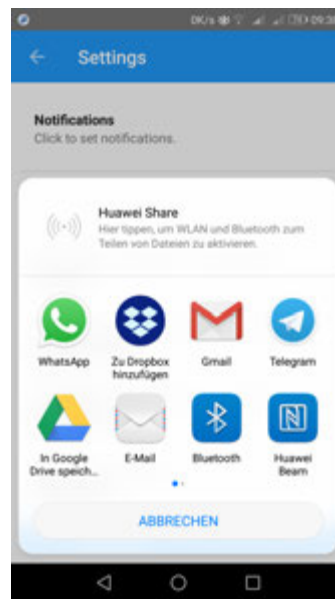
When *Export Data* is clicked, and the application has the proper permissions, users can export their data. Figure 7.9b shows that multiple channels can be used to share this data. A patient can send it via e-mail to his doctor. Figure 7.9c shows how this is done with Gmail. The user only needs to type in the respective e-mail address. Figure 7.10b shows an example of how a questionnaire is exported to PDF.

If the application does not have the permission to access the device memory, the user will be prompted to grant it (see figure 7.10a). When clicking on *OK* the user is forwarded to the Android settings.

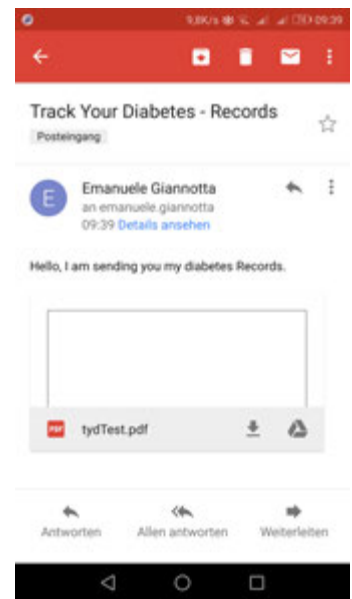
## 7 Introducing the Application



(a) All Settings

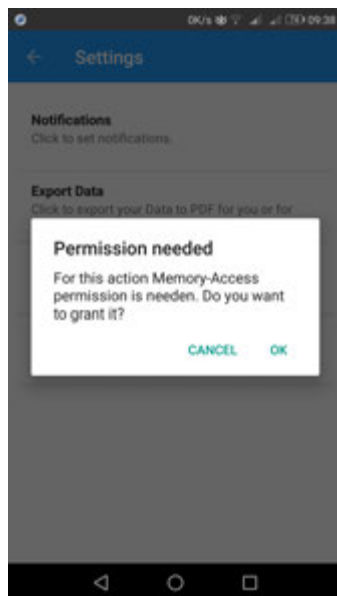


(b) Export Data



(c) Send via E-Mail

Figure 7.9: **Settings:** Screenshot (a) shows all settings. The other two show how users can export their data.

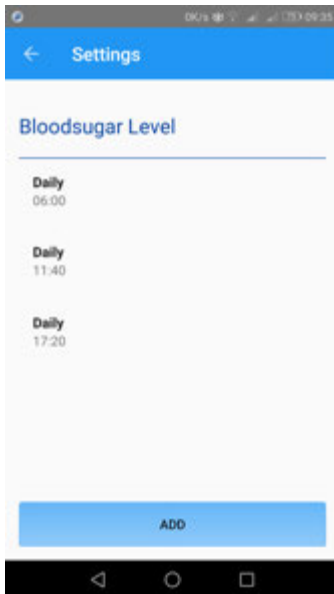


(a) Permission for Export



(b) Exported Data

Figure 7.10: **Data Export:** When the user attempts to export his data, the application needs memory access (a). Screenshot (b) shows a questionnaire exported to PDF.



(a) Notification Schedule



(b) Choose a Day

Figure 7.11: **Notifications:** These screenshots show the notifications for tracking the blood sugar level. In (b) a new notification is being added. First, a date for the notification needs to be chosen. For the continuation see 7.12

Figure 7.11 shows the notifications of a questionnaire. Also, the process of adding a new notification is shown in the following figures (7.11b, 7.12a, 7.12b). Notifications can also be altered with a long click. However, some questionnaires have fixed schedules. The notifications of such cannot be changed.

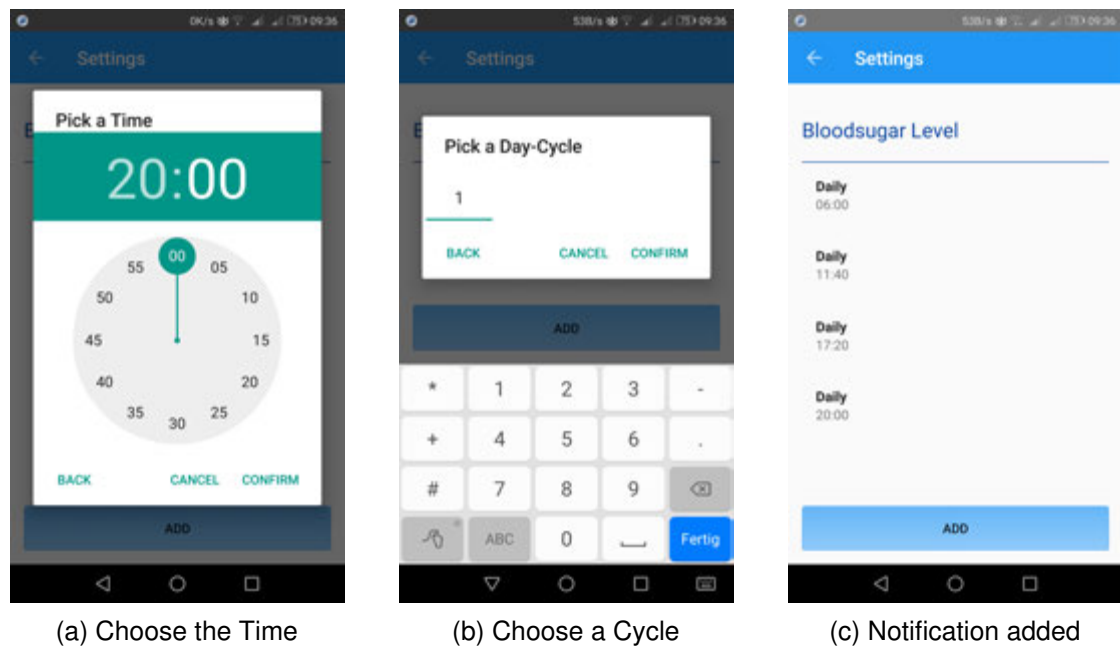


Figure 7.12: **Notifications:** Here the process of adding a new notification, started in figure 7.11b, is continued. After having chosen a date for the notification, a time (a) and a day cycle (b) needs to be chosen. A day cycle of 1 means the notification will be scheduled daily. Screenshot (c) shows that the new notification is now added.



# 8

## Discussion

In this chapter, it is shown whether the requirements were met. Afterward, it is discussed, what part of the implementation code can be reused for an iOS application.

### 8.1 Requirements Check

1. **Registration within the application:** This requirement has been fulfilled. First, users need to register. Then, they confirm their e-mail address by following the link they receive after registration. Now, a login is possible, and the application can be used.
2. **The application can be used without internet connection:** The application can be used without internet connectivity with some limitations. First, and most obvious, the user cannot log in without contacting the backend. Authenticating and downloading the relevant data can only be done over the internet. After the login is done the application can mainly be used without connectivity. However, to subscribe to a new study, the application also needs a working connection. Moreover, the user cannot change his personal data, like his password, without internet. In these few cases, the user will be notified that internet is required.
3. **Participation at multiple studies is possible:** The user can view all active studies. He can subscribe and unsubscribe if he wants to. After he successfully subscribed, all questionnaires of the respective study will be downloaded.
4. **Studies can have different states:** Studies, downloaded from the server, have a flag for this requirement. The application considers that a study can be private

or public. If it is private, the user is prompted to insert a password if he wants to subscribe (see chapter 7.3).

5. **Invitations to studies are possible:** This requirement needs to be met by the backend. As for the application, the invitations are realized similarly to an e-mail inbox. The user will be notified on the main screen if he has been invited (see chapter 7.3).
6. **Invitations can be accepted:** Invitations can be viewed on an extra screen. By clicking an invitation, the information of the respective study will be shown, and the user can accept it. This process is similar to subscribing.
7. **Participation in private studies:** When attempting to subscribe to a private study, the user is prompted to insert the respective password. If he does, the subscription succeeds and the questionnaires are downloaded from the server.
8. **Statistical questionnaires can be filled out within a study:** With an editor account, the head of the study can publish new questionnaires. The application will show these questionnaires to the user, once they are activated.
9. **Statistical questionnaires can be extended or changed:** This requirement needs to be met by the backend server. The application can show all questionnaires provided by the backend.
10. **Questionnaires can be deactivated:** The application only shows active questionnaires. Hence, deactivated questionnaires cannot be filled out.
11. **The questionnaire state can be changed:** This requirement has been met. A one-time questionnaire cannot be filled out more than once.
12. **Statistical questionnaires can be filled out in the app:** This requirement has been met (see section 6.4).
13. **Synchronization of the results:** Data provided by the user is sent to the backend, as soon as a working internet connection is available (see section 6.2).
14. **No initial values:** No initial values are shown to the user.

15. **Notifications for questionnaires:** This requirement has been fulfilled. See chapter 6.1.
16. **Notification schedules can be changed:** The user can view all schedules and change them if they are not fixed (see chapter 7.6).
17. **Show results in the app:** A diagram is plotted to show the history of the blood glucose level and weight of the user (see chapter 7.2).
18. **Export data:** The export function creates a PDF containing the relevant data. This PDF can be shared via e-mail, WhatsApp, and other programs suitable for sharing PDFs (see chapter 7.6).
19. **In-app language settings:** Before the user logs in, he can change the language of the application (see chapter 7.1).
20. **Standard studies:** This requirement has been fulfilled. A standard study for tracking the blood glucose level, weight and the activities, is provided (see chapter 6.5).

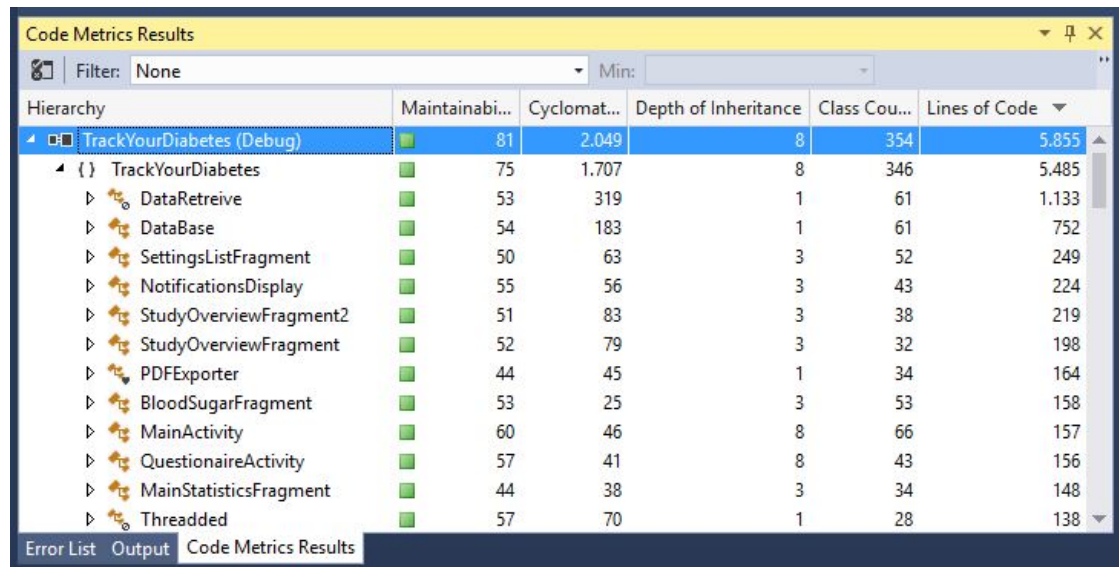
## 8.2 Shared Code

Code metrics of the *Track Your Diabetes* project can be calculated by Visual Studio ([63]). Figure 8.1 shows the output. At this point, the project has around 6000 lines of code.

As already mentioned in chapter 2.3, Xamarin is a cross platform for application development. When implementing the same application for the various operating systems, there is a section of code that can be shared. How big it is, depends on the requirements of the application and how they are implemented.

By reviewing the code metrics in detail, the total amount of shared code can be estimated:

- **Already shareable:** Around 28% (1644 lines) of the code can be shared right away. This code mainly consists of database objects and database methods.



Hierarchy	Maintainabi...	Cyclomat...	Depth of Inheritance	Class Cou...	Lines of Code
TrackYourDiabetes (Debug)	81	2.049	8	354	5,855
TrackYourDiabetes	75	1.707	8	346	5,485
DataRetreive	53	319	1	61	1,133
DataBase	54	183	1	61	752
SettingsListFragment	50	63	3	52	249
NotificationsDisplay	55	56	3	43	224
StudyOverviewFragment2	51	83	3	38	219
StudyOverviewFragment	52	79	3	32	198
PDFExporter	44	45	1	34	164
BloodSugarFragment	53	25	3	53	158
MainActivity	60	46	8	66	157
QuestionnaireActivity	57	41	8	43	156
MainStatisticsFragment	44	38	3	34	148
Threadded	57	70	1	28	138

Figure 8.1: **Code Metrics:** This shows a small part of the code metrics which are calculated by Visual Studio.

- Some changes needed:** Additional 19.35% (1133 lines) can be shared with some changes. Some of these methods use the Android context to access Shared-Preferences. If these few methods are changed to use the iOS or Windows Phone equivalent, this part of the code can largely be reused, also.

# 9

## Summary and Outlook

### 9.1 Summary

The objective of this master's thesis was to enhance the quality of life of diabetes patients by designing an appropriate application, able to improve and support self-health-care and treatment methods of the diabetes patients.

In short, the application *Track Your Diabetes* is designed on the one hand to act as a daily companion for the patients. On the other hand, it can be a valuable help for future treatments of diabetes patients, as the application is able to collect valuable data by using mobile crowdsensing.

By collecting data from the patients, who are already working with the application, researchers can gather large amounts of information about the disease. The so gathered information can be integrated in their research. There are many benefits in this for the patients. It could for example improve treatments or give the physicians a better understanding, how secondary diseases emerge and how they can be prevented.

As an introduction, background information to diabetes was collected and analyzed, stating the different types of diabetes in medical terms and explaining, what it means for the patient to suffer diabetes, and what potential difficulties and secondary diseases might emerge for the patient. Then a summary of mobile crowdsensing was presented, explaining its advantages and how it allows the patients and the medical researchers to take advantage of it. Finally, Xamarin, the framework used for implementation, was

introduced.

The next step was to introduce the related work in this field. The focus was set on projects, able to support diabetes patients, and such designed for mobile crowdsensing in healthcare. Many interesting projects were found and introduced in this thesis. One was *Track Your Tinnitus* ([10], [11]), a project of the Institute of Databases and Information Systems. *Track Your Tinnitus* uses questionnaires for crowdsensing, collecting information about tinnitus.

The requirements for this application were elaborated in cooperation with Dr. Rüdiger Pryss, the supervisor of this work. This requirements specification is presented in this thesis. One requirement was, that the application needed to work in cooperation with an existing backend server, therefore had to be accordingly designed and adjusted. The actual usability of the application had to be considered, also how the data needs to be stored on the device and last, how to integrate all requirements, while coping with the Android SDK. Although it was not a formal requirement, it was important, to make the application extensible for future features or new issues, arising during the process. To achieve this, the application had to be built in a modular way. The questionnaire sheets can be viewed as an example. The answer format within the question-view can easily be replaced by using other fragments.

Finally, the design was implemented. The application was then introduced by showing screen shots. The last step was to review the requirements specification of the application and to check, if everything fits. As the review showed that in fact all requirements were fulfilled, a prove of the concept was given. Therefore, the feasibility of the idea behind *Track your Diabetes* was shown. The Xamarin code was then analyzed, to find out, how much of it could be re-used for the iOS application.

So far, *Track your Diabetes* can serve many purposes by publishing the appropriate questionnaire, then in the future, physicians may be able to deploy their own questionnaires and send automated competent medical advice to the patient by using the feedback

function. The patient could for example be alerted, when his blood sugar level reaches a critical level, in order to make an appointment with his doctor. This way secondary diseases could be recognized quite a lot earlier, and therapies for treating these could be more promising. Patients can use the application on a regular basis to track their blood glucose level, weight and physical activity. Those who forget to check regularly, will be notified and urged to do so without delay. At the same time, data about the disease can be collected in a large-scale manner, if enough people make use of the application. This is also of considerable value for the doctor as he then can rely on an extensive documentation of his patients. For this, various studies can be deployed. Also, there is always the possibility to add new questionnaires, if desired. All data can be used to assist the work of researchers. Overall, it can be expected that *Track your Diabetes* has the potential to enhance the chances for a better life quality of diabetes patients. However, further research will be necessary to prove this.

## 9.2 Future Work

In order to deploy *Track your Diabetes*, a website should be designed and implemented. The website is important for the research managers, enabling them to deploy questionnaires and feedbacks quickly and easily. At present, it is only possible to deploy such a questionnaire by manually sending the respective requests to the server. For that the questionnaires must be formatted in JSON, which can become very confusing. Hence, a web interface with higher usability would be desirable. Another step would be to implement the application for iOS also. By that, many more users could be reached.

As mentioned in the last section, further research and tests must be done, to check, whether this kind of diabetes healthcare application can improve the lifestyle and the health of diabetes patients. Also, a field experiment will have to be done, to see, if the crowdsensing approach of the application works by testing the quality of the collected data.

More features can be implemented into the application to make it more complete. If

## 9 Summary and Outlook

diabetes patients have all features they need in one application, it is more likely for them to use it. And the more users the application has, the more data can be collected for research. Additional features could be:

- **Calorie meter:** Proper nutrition is an important part of the diabetes treatment. If the application would be able to track, what their users eat, the doctors and dieticians could more easily point out, which habits the patients can improve. Constantly improving the diet of a diabetes patient can lead to less complications. This function could already be implemented by providing an appropriate questionnaire.
- **Patient education:** In the future, the application could have a section, where the user can educate himself about his disorder, or look something up, if he has further questions. For this FAQs (frequently asked questions) could be provided.
- **Blood glucose meter:** A feature could be implemented to connect a blood glucose meter to the application. Preferably the application would then be able to track the measured value without further interaction of the user. This would enhance the usability of the application and is also a good argument for the patient to use it.
- **Motivation:** A system could be designed for motivating the users to enhance their diabetes self-care. Users could receive positive feedback, when their self-care behaviors have improved. If their self-care behaviors have not improved, these could be motivated by enlighten and advise them of the benefits of good self-care behavior.

All in all, this is an interesting topic, and it will be exciting, how *Track Your Diabetes* will evolve in the future.



# Bibliography

- [1] World Health Organization: Data and statistics. <http://www.euro.who.int/en/health-topics/noncommunicable-diseases/diabetes/data-and-statistics>. (Online, visited 2017-12-08)
- [2] Fézer, Z., Kovács, L.: The economic impact of diabetes. In: 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY). (2017) 000077–000082
- [3] Seuring, T., Archangelidi, O., Suhrcke, M.: "The Economic Costs of Type 2 Diabetes: A Global Systematic Review". *PharmacoEconomics* **33** (2015) 811–831
- [4] Kelley, H., Chiasson, M., Downey, A., Pacaud, D.: "The clinical impact of ehealth on the self-management of diabetes: the double adoption of IT and health change". *Journal of the Association for Information Systems* **12** (2011) 208–234
- [5] Ekroos, N., Jalonen, K.: E-health and diabetes care. *Journal of Telemedicine and Telecare* **13** (2007) 22–23
- [6] Harno, K., Kauppinen-Mäkelin, R., Syrjäläinen, J.: Managing diabetes care using an integrated regional e-health approach. *12 Suppl 1* (2006) 13–5
- [7] Pryss, R., Schlee, W., Langguth, B., Reichert, M.: Mobile Crowdsensing Services for Tinnitus Assessment and Patient Feedback. In: 6th IEEE International Conference on AI & Mobile Services (IEEE AIMS 2017), IEEE Computer Society Press (2017)
- [8] Pryss, R., Probst, T., Schlee, W., Schobel, J., Langguth, B., Neff, P., Spiliopoulou, M., Reichert, M.: Mobile Crowdsensing for the Juxtaposition of Realtime Assessments and Retrospective Reporting for Neuropsychiatric Symptoms. In: 30th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2017), IEEE Computer Society Press (2017)

## *Bibliography*

- [9] Pryss, R., Reichert, M., Herrmann, J., Langguth, B., Schlee, W.: Mobile Crowd Sensing in Clinical and Psychological Trials - A Case Study. In: 28th IEEE Int'l Symposium on Computer-Based Medical Systems, IEEE Computer Society Press (2015) 23–24
- [10] Pryss, R., Reichert, M., Langguth, B., Schlee, W.: Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy and Research. In: IEEE 4th International Conference on Mobile Services (MS 2015), IEEE Computer Society Press (2015) 352–359
- [11] Jochen Herrmann: Track your Tinnitus. <https://www.trackyourtinnitus.org/home>. (Online, visited 2017-12-10)
- [12] Probst, T., Pryss, R., Langguth, B., Schlee, W.: Emotional states as mediators between tinnitus loudness and tinnitus distress in daily life: Results from the "TrackYourTinnitus" application. *Scientific Reports* **6** (2016)
- [13] Probst, T., Pryss, R., Langguth, B., Rauschecker, J., Schobel, J., Reichert, M., Spiliopoulou, M., Schlee, W., Zimmermann, J.: Does tinnitus depend on time-of-day? An ecological momentary assessment study with the "TrackYourTinnitus" application. *Frontiers in Aging Neuroscience* **9** (2017) 253–253
- [14] Probst, T., Pryss, R., Langguth, B., Spiliopoulou, M., Landgrebe, M., Vesala, M., Harrison, S., Schobel, J., Reichert, M., Stach, M., Schlee, W.: Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples? *Frontiers in Aging Neuroscience* **9** (2017) 113–113
- [15] Schlee, W., Pryss, R., Probst, T., Schobel, J., Bachmeier, A., Reichert, M., Langguth, B.: Measuring the Moment-to-Moment Variability of Tinnitus: The TrackYourTinnitus Smart Phone App. *Frontiers in Aging Neuroscience* **8** (2016) 294–294
- [16] Arastéh, K.e.a.: *Duale Reihe: Innere Medizin*. Thieme (2013)
- [17] Guyton, A.C., Hall, J.E.: *Textbook of Medical Physiology*. Saunders (2006)
- [18] Howorka, K.e.a.: *Funktionelle Insulintherapie*. Springer (1996)

- [19] Van den Berghe, M.G.: Acute Endocrinology: From Cause to Consequence. Humana Press (2008)
- [20] Jian, A., Xiaolin, G., Jianwei, Y., Yu, S., Xin, H.: Mobile Crowd Sensing for Internet of Things: A Credible Crowdsourcing Model in Mobile-Sense Service. In: 2015 IEEE International Conference on Multimedia Big Data. (2015) 92–99
- [21] Ganti, R.K., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Communications Magazine **49** (2011) 32–39
- [22] Alsheikh, M.A., Jiao, Y., Niyato, D., Wang, P., Leong, D., Han, Z.: The Accuracy-Privacy Trade-off of Mobile Crowdsensing. IEEE Communications Magazine **55** (2017) 132–139
- [23] Vergara-Laurens, I.J., Jaimes, L.G., Labrador, M.A.: Privacy-Preserving Mechanisms for Crowdsensing: Survey and Research Challenges. IEEE Internet of Things Journal **4** (2017) 855–869
- [24] Fiandrino, C., Capponi, A., Cacciatore, G., Kliazovich, D., Sorger, U., Bouvry, P., Kantarci, B., Granelli, F., Giordano, S.: CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments. IEEE Access **5** (2017) 3490–3503
- [25] Hermes, D.: Xamarin mobile application development: cross-platform C# and Xamarin.Forms fundamentals. Apress (2015)
- [26] Panigrahy, N.: Xamarin mobile application development for Android: develop, test, and deliver fully featured Android applications using Xamarin. Packt Publishing (2015)
- [27] CHRODIS: ABOUT US. <http://chrodis.eu/about-us/>. (Online, visited 2017-12-10)
- [28] Ruf-Leuschner, M., Brunnemann, N., Schauer, M., Pryss, R., Barnewitz, E., Liebrecht, M., Reichert, M., Elbert, T.: Die KINDEX-App - ein Instrument zur Erfassung und unmittelbaren Auswertung von psychosozialen Belastungen bei Schwangeren in der täglichen Praxis bei Gynäkologinnen, Hebammen und in Frauenkliniken. Verhaltenstherapie (2016)

## *Bibliography*

- [29] Preuveneers, D., Berbers, Y.: Mobile Phones Assisting with Health Self-care: A Diabetes Case Study. In: Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services. MobileHCI '08, New York, NY, USA, ACM (2008) 177–186
- [30] Gan, S.K.E., Koshy, C., Nguyen, P.V., Haw, Y.X.: "An overview of clinically and healthcare related apps in Google and Apple app stores: connecting patients, drugs, and clinicians". *Scientific Phone Apps and Mobile Devices* **2** (2016) 8
- [31] Al-Tae, A., Al-Tae, A., Muhsin, Z., Al-Tae, M., Al-Nuaimy, W.: Towards Developing Online Compliance Index for Self-Monitoring of Blood Glucose in Diabetes Management. In: 2016 9th International Conference on Developments in eSystems Engineering (DeSE). (2016) 45–50
- [32] Winterlich, A., Stevenson, I., Waldren, A., Dawson, T.: Diabetes Digital Coach: Developing an Infrastructure for e-Health Self-Management Tools. In: 2016 9th International Conference on Developments in eSystems Engineering (DeSE). (2016) 68–73
- [33] Monteith, S., Glenn, T., Geddes, J., Whybrow, P.C., Bauer, M.: "Big data for bipolar disorder". *International Journal of Bipolar Disorders* **4** (2016) 10
- [34] mySugr GmbH: mySugr. <https://mysugr.com/>. (Online, visited 2017-11-27)
- [35] SocialDiabetes: SocialDiabetes. <https://www.socialdiabetes.com/>. (Online, visited 2017-11-27)
- [36] mapmydiabetes: mapmydiabetes. <http://www.mapmyhealth.co.uk>. (Online, visited 2017-11-28)
- [37] OVIVA: OVIVA. <https://oviva.com/de/>. (Online, visited 2017-11-28)
- [38] Diabetes Digital Media: Diabetes Forum. <http://www.diabetes.co.uk/>. (Online, visited 2017-11-28)
- [39] Gadge, P.: Gadge Diabetes Care. <https://play.google.com/store/apps/details?id=com.gadgesdiabetescare>. (Online, visited 2017-11-29)

- [40] MAZ Digital Inc.: Diabetes Self-Management. <https://play.google.com/store/apps/details?id=com.maz.dsmmag>. (Online, visited 2017-11-29)
- [41] mEL Studio: My Sugar Diary. <https://play.google.com/store/apps/details?id=com.aiims.mysugardiary&hl=de>. (Online, visited 2017-11-28)
- [42] Sirma Medical Systems: Diabetes:M. <https://www.diabetes-m.com/>. (Online, visited 2017-11-28)
- [43] Diabetes Digital Media: Diabetes PA (Diabetes Manager). <https://play.google.com/store/apps/details?id=com.diabetes.android>. (Online, visited 2017-11-28)
- [44] Fahlteich, P.: Diaguard: Diabetes Tagebuch. <https://play.google.com/store/apps/details?id=com.faltenreich.diaguard>. (Online, visited 2017-11-28)
- [45] SquareMed Software GmbH: Diabetes Connect. <http://www.diabetesconnect.de/>. (Online, visited 2017-11-28)
- [46] MedHelp: Sugar Sense. <https://itunes.apple.com/us/app/sugar-sense-diabetes-app-blood-glucose-mgmt/id880725347?mt=8>. (Online, visited 2017-11-29)
- [47] Johannes Schobel: API Documentation v1. <https://tyt.johannesschobel.com/dingodocs/v1.html>. (Online, visited 2017-12-02)
- [48] L Craig, C., Marshall, A., Sjostrom, M., Bauman, A., L Booth, M., Ainsworth, B., Pratt, M., Ekelund, U., Yngve, A., F Sallis, J., Oja, P.: International Physical Activity Questionnaire: 12-Country Reliability and Validity. **35** (2003) 1381–95
- [49] Fisher, L., Glasgow, R.E., Mullan, J.T., Skaff, M.M., Polonsky, W.H.: Development of a Brief Diabetes Distress Screening Instrument. **6.3** (2008) 246–252
- [50] Schmitt, A., Gahr, A., Hermanns, N., Kulzer, B., Huber, J., Haak, T.: The Diabetes Self-Management Questionnaire (DSMQ): development and evaluation of an instrument to assess diabetes self-care activities associated with glycaemic control. **11** (2013) 138

## *Bibliography*

- [51] : Pro ASP.NET MVC 5 platform. Online-ausg. edn. APress, New York (2014)
- [52] Google Inc.: App Manifest. <https://developer.android.com/guide/topics/manifest/manifest-intro.html/>. (Online, visited 2017-11-29)
- [53] Xamarin Inc.: Android.App.ActivityAttribute.MainLauncher Property. <https://developer.xamarin.com/api/property/Android.App.ActivityAttribute.MainLauncher/>. (Online, visited 2017-11-29)
- [54] Google Inc.: Fragments. <https://developer.android.com/guide/components/fragments.html>. (Online, visited 2017-11-30)
- [55] SQLite: SQLite. <https://www.sqlite.org/>. (Online, visited 2017-12-01)
- [56] Google Inc.: PendingIntent. <https://developer.android.com/reference/android/app/PendingIntent.html>. (Online, visited 2017-12-03)
- [57] Google Inc.: AlarmManager. <https://developer.android.com/reference/android/app/AlarmManager.html>. (Online, visited 2017-12-03)
- [58] Google Inc.: SystemClock. <https://developer.android.com/reference/android/os/SystemClock.html>. (Online, visited 2017-12-03)
- [59] Google Inc.: Saving Key-Value Sets. <https://developer.android.com/training/data-storage/shared-preferences.html>. (Online, visited 2017-12-07)
- [60] Xamarin Inc.: Android.Content.ISharedPreferences. <https://developer.xamarin.com/api/type/Android.Content.ISharedPreferences/>. (Online, visited 2017-12-07)
- [61] Newtonsoft: Json.NET. <https://www.newtonsoft.com/json>. (Online, visited 2017-12-05)
- [62] Icons8: All the Icons You Need. Guaranteed. <https://icons8.com/>. (Online, visited 2017-12-11)
- [63] Microsoft: Visual Studio. <https://www.visualstudio.com/de/?rr=https%3A%2F%2Fwww.google.de%2F>. (Online, visited 2017-12-08)

# List of Figures

2.1	"Xamarin libraries bind to native OS libraries" [25]. . . . .	20
2.2	"Xamarin.Forms architecture with custom renderers" [25]. . . . .	20
5.1	<b>Fragments within Activities:</b> This figure introduces how fragments are used within activities. In most activities one fragment is shown at any point in time (a). However, when answering questionnaires, an additional fragment is used to display the answers (b). . . . .	41
5.2	<b>Activities and Fragments:</b> This picture shows the application's activities and the associated fragments. This is the first part of a simplified UML. For the second part see 5.3 . . . . .	42
5.3	<b>Activities and Fragments:</b> This picture shows the activities and their fragments. This is the second part of a simplified UML. For the first part see 5.2 . . . . .	43
5.4	<b>Application Process:</b> This process shows, how the application is used. First the user needs to authenticate. Afterwards, he will be forwarded to the main screen where the whole functionality of the application can be accessed. . . . .	46
5.5	<b>Main Screen Process:</b> This is a sub process of figure 5.4. It shows how the user can access the functionality of the application. It is a simplified diagram since most of the functionality is presented as a sub process. . .	48
5.6	<b>Data model (part 1):</b> This is one section of the tables used to store the relevant data on the device. <i>Task</i> is a local object. Except for <i>Task</i> , all other tables in this figure are modeled after the objects the backend server sends. . . . .	50
5.7	<b>Data model (part 2):</b> This is another part of the tables used to store the relevant data. Here <i>Invitation</i> is modeled after the data received from the backend. The two remaining tables are only available on the local device.	51

## List of Figures

- 6.1 **NotificationTask:** This picture shows the data structure for a notification task. It is a database table and used to handle all notifications. . . . . 56
- 6.2 **List row:** This image shows the layout of list rows in the *QuestionnaireListFragment*. On the left, a picture is shown. To the right of the picture, the name and description of the questionnaire are presented by TextViews. . . . . 65
- 7.1 **Login and Register:** These screenshots show the user interfaces for logging in, changing the language and registering. . . . . 70
- 7.2 **Main Screen:** These screenshots show the three tabs of the main screen. 71
- 7.3 **Navigation Menu:** These figures present the navigation menu by which nearly all features of the app can be accessed. . . . . 72
- 7.4 **List of Studies:** These images show how studies are listed. These studies are fictional and are only used for testing. . . . . 73
- 7.5 **Study overview:** These screenshots show the overview to studies. To reach these screens the respective study must be clicked in the user interfaces shown in figure 7.4 . . . . . 73
- 7.6 **Questionnaire listing:** These images show how the user's questionnaires are listed and how information is displayed. . . . . 75
- 7.7 **Answer sheets:** These screens illustrate single questions and their possible answers. If the last question is reached, the "Next Question" button becomes a "Submit" button. . . . . 75
- 7.8 **Blood Sugar Level and Weight:** The screenshots (a), (b) and (c) show how the user can track his blood sugar level and weight. . . . . 77
- 7.9 **Settings:** Screenshot (a) shows all settings. The other two show how users can export their data. . . . . 78
- 7.10 **Data Export:** When the user attempts to export his data, the application needs memory access (a). Screenshot (b) shows a questionnaire exported to PDF. . . . . 78
- 7.11 **Notifications:** These screenshots shows the notifications for tracking the blood sugar level. In (b) a new notification is being added. First, a date for the notification needs to be chosen. For the continuation see 7.12 . . . . 79



7.12	<b>Notifications:</b> Here the process of adding a new notification, started in figure 7.11b, is continued. After having chosen a date for the notification, a time (a) and a day cycle (b) needs to be chosen. A day cycle of 1 means the notification will be scheduled daily. Screenshot (c) shows that the new notification is now added. . . . .	80
8.1	<b>Code Metrics:</b> This shows a small part of the code metrics which are calculated by Visual Studio. . . . .	84

Name: Emanuele Giannotta

Matrikelnummer: 750755

### **Erklärung**

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Emanuele Giannotta