



Markus Bernhard Schütz, BSc

# Development of a learning diary for a MOOC platform

## Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

**Graz University of Technology**

Supervisor

Priv.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner

Institute of Interactive Systems and Data Science  
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, November 2019



## **Affidavit**

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

---

Date

---

Signature



# Abstract

The term "learning diary" refers to a pedagogical tool, that is used by people who are in a learning progress to improve and keep track of their learning progress and learning process. This master thesis provides a detailed explanation of the mobile implementation of a learning diary for a MOOC platform. MOOC describes "Massive Open Online Course", which is a course for people who want to learn online. The applications for Android and iOS devices, which are the outcome of this master thesis, aim for the goal to keep an overview of all courses and events of students and help them increase their learning progress with the aid of gamification. Gamification describes a process which implements gaming-elements in non-gaming-contexts. The hypothesis given is "A mobile learning diary application for a MOOC platform can improve the learning progress of students of the MOOC platform". For the development of the applications, detailed research about learning diaries, gamification and similar applications is necessary. Furthermore, a prototype is created, which defines ideas, features and design, to ensure correct usability and user experience for the users of the mobile applications. The result of this master thesis is two applications, which are evaluated with the aid of a survey. This survey is answered by users of the application. A detailed evaluation proves the hypothesis as correct.



# Kurzfassung

Der Begriff "Lerntagebuch" beschreibt ein pädagogisches Werkzeug welches von Personen genutzt wird, die sich in einem Lernprozess befinden. Es soll dabei helfen, den Lernprozess und den Lernfortschritt zu verbessern. Diese Masterthesis bietet eine detaillierte Beschreibung über die Umsetzung eines Lerntagebuchs für eine MOOC Plattform. MOOC bedeutet "Massive Open Online Course" und ist ein Kurs für Personen, welche sich online weiterbilden möchten. Die Applikationen für Android und iOS Geräte, welche aus dieser Masterthesis hervorgehen, haben das Ziel einen Überblick über Kurse und Termine für die StudentInnen der Plattform zu behalten und ihnen dabei zu helfen, den Lernfortschritt mit Gamification zu verbessern. Gamification beschreibt einen Prozess der Spielelemente in einem nicht-spielerischem Kontext implementiert. Die Hypothese dieser Masterarbeit lautet "Eine mobile Lerntagebuch-Applikation für eine MOOC Plattform verbessert den Lernfortschritt der Studenten der MOOC Plattform". Diese Masterthesis evaluiert mit Hilfe der beiden Applikationen, ob diese Hypothese sich als wahr beweist. Für die Umsetzung der Applikationen ist eine detaillierte Recherche über Lerntagebücher, Gamification und ähnlichen Applikation notwendig. Des Weiteren wird ein Prototyp erstellt, welcher über Ideen, Features und Design entscheidet und dafür sorgt, dass eine korrekte Usability und User Experience für die Nutzer der Applikationen sichergestellt wird. Das Ergebnis dieser Masterthesis sind zwei Applikationen die mit Hilfe einer Umfrage evaluiert werden. Die Umfrage wird von Nutzern der Applikationen beantwortet. Eine detaillierte Evaluierung beweist, dass die Hypothese korrekt ist.





# Contents

<b>Abstract</b>	<b>v</b>
<b>Kurzfassung</b>	<b>vi</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Goals & Objectives . . . . .	2
1.2. Methodology & Structure . . . . .	3
<b>2. State of the art</b>	<b>5</b>
2.1. Learning diaries . . . . .	5
2.2. Gamification . . . . .	7
2.3. Similar applications . . . . .	12
2.3.1. Milk . . . . .	12
2.3.2. Seesaw . . . . .	13
<b>3. The prototype</b>	<b>19</b>
3.1. Idea . . . . .	19
3.1.1. Colouring of courses and events . . . . .	21
3.1.2. Notifications . . . . .	21
3.1.3. Creating events and learn goals . . . . .	22
3.1.4. Creating notes . . . . .	24
3.1.5. Filtering events and automatic scrolling . . . . .	25
3.2. Gamification . . . . .	26
3.3. Mockup . . . . .	28
<b>4. Developing the applications</b>	<b>43</b>
4.1. Strategy . . . . .	44
4.2. Android . . . . .	46
4.2.1. Project setup and libraries . . . . .	46

## Contents

4.2.2. Implementing the Android version . . . . .	50
4.3. iOS . . . . .	75
4.3.1. Project setup and libraries . . . . .	75
4.3.2. Implementing the iOS version . . . . .	76
<b>5. Proof of concept</b>	<b>99</b>
5.1. Prototype vs. application . . . . .	99
5.2. Survey & Evaluation . . . . .	108
5.2.1. The survey . . . . .	109
5.2.2. The evaluation . . . . .	114
<b>6. Conclusion</b>	<b>125</b>
6.1. Lessons learned . . . . .	125
6.2. Limitations & future work . . . . .	128
6.3. Summary & Outlook . . . . .	131
<b>Appendix</b>	<b>133</b>
<b>A. Survey of applications</b>	<b>135</b>
<b>Bibliography</b>	<b>141</b>

# List of Figures

2.1. Screenshot of Milk showing targets and assessments . . . . .	14
2.2. Screenshot of Seesaw . . . . .	17
3.1. Design of listing events and learn goals to the user . . . . .	29
3.2. Design of creating an event (a), a learn goal (b) or a note (c) .	31
3.3. Design of listing subscribed courses to the user . . . . .	34
3.4. Design of listing events, learn goals and notes of a specific course to the user . . . . .	35
3.5. Design of listing received notifications to the user . . . . .	37
3.6. Design of the Profile screen including gamification . . . . .	39
3.7. Design of listing progress of subscribed courses to the user .	41
4.1. Login screen of Android application . . . . .	51
4.2. List of events and learn goals in Android application . . . . .	58
4.3. Detail view event - Android application . . . . .	60
4.4. Creating event, learn goal and note - Android application . .	64
4.5. List of subscribed courses in Android application . . . . .	65
4.6. List of events of specific subscribed course - Android application	68
4.7. List of received notifications in Android application . . . . .	70
4.8. Profile including gamification - Android application . . . . .	72
4.9. Colouring - screen of Android application . . . . .	74
4.10. Login screen of iOS application . . . . .	78
4.11. List of events and learn goals in iOS application . . . . .	82
4.12. Detail view event - iOS application . . . . .	85
4.13. Creating event, learn goal and note - iOS application . . . . .	87
4.14. List of subscribed courses in iOS application . . . . .	89
4.15. List of events of specific subscribed course - iOS application .	92
4.16. List of received notifications in iOS application . . . . .	94
4.17. Profile including gamification - iOS application . . . . .	96

## List of Figures

4.18. Colouring - screen of iOS application . . . . .	98
5.1. Pie chart of answers for "Do you like the design/appearance of the app?" . . . . .	116
5.2. Pie chart of answers for "Do or did you have problems controlling the app?" . . . . .	117
5.3. Pie chart of answers for "Do you think the navigation of the app is confusing?" . . . . .	117
5.4. Pie chart of answers for "Do you use the app at least once a week in addition to an iMooX-Course?" . . . . .	118
5.5. Pie chart of answers for "After playing the tutorial, did you have the feeling you understand the app?" . . . . .	119
5.6. Pie chart of answers for "After playing the tutorial, did you have the feeling that you can control the app without any problems?" . . . . .	120
5.7. Pie chart of answers for "Does the app help you to keep an overview of your courses and events?" . . . . .	121
5.8. Pie chart of answers for "Does the app increase your learning progress?" . . . . .	122

# Listings

4.1. Gradle for module app . . . . .	48
4.2. Object Realm_Level in Android . . . . .	53
4.3. Object Realm_Level in iOS . . . . .	79



# 1. Introduction

This master thesis confronts itself with the task of improving the learning progress of users in an established MOOC platform. The MOOC platform is already implemented as a web application and already has a community. For increasing the learning progress, this master thesis targets a mobile application, available for the two most used operating systems Android and iOS on mobile devices<sup>1</sup>. This mobile application should represent a learning diary that is combined with gamification. There exists a broad field of researches, studies and books that dedicate to the proof of concept of learning diaries. Cazan (2012, p. 413) describes learning journals as a *"self-guided way of writing that allows for elaboration and reflection on learning content"*. It is also explained, that the task of self-regulated learning is related to improving metacognition, strategic action and the motivation to learn. Clipa, Ignat, and Stanciu (2012) explain that learning diaries have a great influence on the development of metacognitive strategies. Learning how to learn is a main part of the academic background because it trains the learning process of students. Both mention, that a learning diary improves metacognition, which is the engagement with thoughts, opinions, mindset, attention and creativity. (Metakognition 2019)

As mentioned, this master thesis combines a learning diary with gamification. Nicholson (2015, p. 1) clarifies that *"meaningful gamification is the use of gameful and playful layers to help a user find personal connection that motivate engagement with a specific context for long-term change"*. Seaborn and Fels (2015) mentions, that gamification is an approach for motivating users and rising engagement and enjoyment to non-gaming, computer-oriented environments. Nicholson (2015) and Seaborn and Fels (2015) mention, that

---

<sup>1</sup>Market share of mobile operating systems worldwide 2012-2019 2019.

## 1. Introduction

gamification is used to raise the motivation to succeed and proof in a non-game context area.

Using the researches regarding learning diaries and gamification executed during this master thesis and respecting them in an application for Android and iOS, it should be clarified that the hypothesis "A learning diary for a MOOC platform increases the learning progress of students" proves correct and to which extent it proves correct.

### 1.1. Goals & Objectives

According to chapter 1, this section describes the goals and objectives of this master thesis. Furthermore, it briefly describes why the goals and objectives are defined.

- **Gain knowledge of learning diaries, gamification and similar applications.** This objective is set to respect researched information that already had been done when creating the prototype and the applications. It is necessary to avoid mistakes and to provide a meaningful application to the user.
- **Design a prototype.** Designing a prototype is necessary before developing the applications to define the ideas, design and features of the applications.
- **Develop an application for Android devices.** Android devices are the most used mobile devices<sup>2</sup>. Therefore it is reasonable to provide an application to target students that are used to Android operating systems.
- **Develop an application for iOS devices.** Devices that make use of the iOS operating systems are the second most used ones. To target the other main community, besides the users of Android devices, it is necessary to provide an application for iOS devices.

---

<sup>2</sup>Market share of mobile operating systems worldwide 2012-2019 2019.



- **Improve overall learning progress of users.** The hypothesis of this master thesis claims, that the learning progress of users of the MOOC platform improves when implementing a learning diary application in addition to the MOOC platform. Therefore, it is necessary for the applications that they target this main goal.

## 1.2. Methodology & Structure

The main structure of this master thesis consists of five chapters. The five chapters are about state of the art, the prototype, development of the application, proof of concept and a conclusion including an outlook and facts about future work. Each of these chapters builds on the previous one. This means the outcome and research done in state of the art is the input for the part of the prototype. The output of the prototype is needed for the development of the application. The chapter of proof of concept is based on the application. The conclusion itself is based upon all other chapters because it sums up the whole master thesis.

The first chapter is about the background of this master thesis. The topics discussed in this chapter are about learning diaries, gamification and similar application. In the section for learning diaries, a basic understanding of learning diaries is provided. It describes how a learning diary works and which goal it aims for. The section for gamification also provides a basic understanding of the topic. It describes gamification in a way that is useful for this thesis and what to watch out for. The section for similar applications lists applications that provide a type of learning diary. This is necessary to get an overview of what already exists in the context of mobile learning diary applications.

The second chapter is about the prototype. It is built based on the information gained in the first chapter. The process of the prototype contains the idea of the application as well as a design and mockup. In the section for the idea, all necessary information about the application is provided. It

## 1. Introduction

defines what the goal is and what the application should aim for. Furthermore, it defines the features that the applications should be capable of. The design and mockup was created during the prototype process. It provides a detailed design of how the application should look and how the features should be implemented. The outcome of the prototype is important for the third chapter.

The third chapter is about the implementation of applications for Android and iOS. The versions for Android and iOS are separated. Each version is explained in detail. Every necessary feature is described. For every feature, all necessary Views, ViewModels, Models and further resources are explained. Each implementation is provided in a way, that a person who has experience in developing applications for Android and iOS can rebuild the applications. The applications are the outcome of this chapter. Based on these applications, the survey and evaluation follows, which is described in the fourth chapter.

The fourth chapter is about the proof of concept. It defines whether the features described during the prototype process have been implemented in the applications and to which extent they have been implemented. Furthermore, the chapter contains the survey that was provided to the users of the applications and its evaluation. The survey consisted of ten different questions which lead to the possibility to prove the hypothesis of this master thesis. The questions were about the design, navigation, controls and overall usability of the applications. Furthermore, it clarifies the question, whether the learning progress of the users has increased using the applications. Overall, this chapter reviews whether the concept is converted successfully.

The fifth and last chapter summarises the master thesis. Besides summarising the master thesis, it also describes what should be implemented in the applications in future work. A list of lessons learned is provided to prevent future work from blocking because of the same problems that came up during this master thesis. Finally, an outlook about the potential of the applications is provided.

## 2. State of the art

This chapter explains the research which is necessary for the further understanding of this master thesis and the basic idea for the applications which are the outcome of this master thesis. First, an analysis of learning diaries is given. After that, gamification is described in general and for the non-gaming-context education. After that, a list of applications that include a learning diary is provided.

### 2.1. Learning diaries

A learning diary is a pedagogical tool. It is used by people, who are in a learning process, to improve and keep track of their learning process and progress. The learning behaviour is documented in the learning diary for a certain time. The goal of a learning diary is to deal with the subject matter and improve the understanding of it<sup>1</sup>.

A learning diary forces students to deal with the subject matter and review the learning process. For that, the students write down which important things they have learned and what was clear and what they did not understand. Therefore, the learning diary is a tool for the post-processing of some lessons. The strong-text view is based on the assumption, that the major part of our memory can only be accessed through verbalisation and textualisation. Therefore, an open writing in the learning diary should be possible to give the student the possibility to write down their memory and consolidate the things they have learned. The writing-as-problem-solving-view stands in contrast to this approach. In this approach, the writing is considered as a problem-solving process, where students create a dialectical

---

<sup>1</sup>*Lerntagebuch 2017.*

## 2. State of the art

movement between a rhetorical and a semantic problem. This means, that the students who align with this approach write down a high-quality text to retain their memory. An example of this is the consideration of a specific topic or a scientific article. (Spiel et al., 2010)

Even though it is not clearly specified what a learning diary deals with and that it does not matter whether it is written by hand or digital, there are still some basic conditions that have to be fulfilled to make it a learning diary. The following points describe these basic conditions:

- **Learning diaries are written.** With the writing of the own learning process, underlying thinking processes become externalised and explicated. (Petko, 2013)
- **Learning diaries are continuous.** As described earlier in this section, the writing of a learning diary targets progressive documentation about results, misunderstandings and knowledge of learned subjects. Students improve during writing the learning diary. So the goal is not to finish a learning diary, but to learn during writing the learning diary. (Petko, 2013)
- **Learning diaries are personal.** As part of the personal learning environment, learning diaries are the responsibility of the student. Entries in learning diaries are personal and a special permission for others is required to view the learning diary. Entries and the learning diary itself have to be protected from public access. In formal learning settings, this principle can be handled differently. A complete orifice of the learning diary involves the danger of not documenting unfinished processes, but only presentable intermediate steps. This means, if others get involved in the learning diary and can see the content, the writer of the learning diary inclines to write down finished learning processes and progress instead of writing down the current mood about the learned subject in their own words. (Petko, 2013)
- **Learning diaries are connected to learning contexts.** Learning diaries can be deployed in different learning contexts and depending on which learning context the learning diary is connected to, the character of the learning diary changes. In informal settings, the writing of a learning

## 2.2. Gamification

diary is usually a voluntary and self-responsible part of the own learning strategy. In formal learning contexts, the writing of a learning diary can be recommended or obligatory, for example as a part of a learning exercise or as a part of a learning proof. Learning diaries can exclusively be written for own reflection or for other receivers like other students or teaching staff. With this approach, teaching staff can get information about the learning progress, learning difficulties, plans and the thoughts of the student. (Petko, 2013)

- **Writing a learning diary does not automatically lead to reflection or metacognition.** The quality of learning diaries can differ. Deciding aids to guarantee the conducive quality can be hints, examples from other learning diaries or recursive feedback about the learning diary. Usually, these types of aids only are available in formal contexts of education. To avoid a contradiction to the personal character of learning diaries, agreements about the handling of these aids have to be made. Ideally, the aids must not replace the self-controlling of the students but can be a supplement. (Petko, 2013)

## 2.2. Gamification

This chapter describes what the term gamification means, how it developed and how it applies to a non-game context like learning. Gamification is an approach for motivating users and increasing engagement and enjoyment of non-gaming, computer-oriented environments. The typical people playing games are 30 years old, 45% of them are female and are playing games like puzzle solving, board or casual games. (Seaborn and Fels, 2015) Furthermore, Burke (2014, p. 6) describes the term gamification as *"the use of game mechanics and experience design to digitally engage and motivate people to achieve their goals"*. Burke (2014, p. 6) breaks down this definition as follows:

- *"Game mechanics describes the key elements that are common to many games, such as points, badges, and leaderboards."*
- *"Experience design describes the journey players take with elements such as gameplay, play space, and storyline."*

## 2. State of the art

- *“Gamification is a method to digitally engage rather than personally engage, meaning that players interact with computers, smartphones, wearable monitors, or other digital devices.”*
- *“The goal of gamification is to motivate people to change behaviours or develop skills, or to drive innovation.”*
- *“Gamification focuses on enabling players to achieve their goals—and as a consequence, the organisation achieves its goals.”*

Söbke and Zander (2018) describe the implementation of a quiz-app in a course at university. The study shows, that the implementation of a quiz-app in a didactic context was successful. Based on the high values for general motivation, the deployment consisted of learning exercises, which were approached with a high learning motivation. The scale of the motivation was confirmed with the actual learning effort of the students. Therefore, the combination of a didactic context with the gamification in form of a quiz-app can be seen as successful, because the game mechanics of the quiz-app lead to a high amusement. Nevertheless, in the study, Söbke and Zander mention, that there is a need for optimisation which targets the questions in the areas of knowledge as well as the regularisation of the competition of single applicants. Furthermore, there is a need for an explanation whether the didactic context can be applied to different specialised fields and how students react if the didactic context keeps recurring in different areas. The study of the quiz-app is described as a showcase of successful conjunction between a didactic context and gamification to improve motivation. (Söbke and Zander, 2018)

Hsin-Yuan and Dilip (2013) described in a survey, that while the concept of gamification is rather simple, gamifying a concept is not. To overcome the difficulty of applying gamification to an educational context and raise the motivation of students to gain knowledge, they mention five steps. These five steps help to create a concept of gamification and to understand how to apply it to the educational context.

- *“Understanding the Target Audience and the Context”*. The first step to gain knowledge about how gamification is applied to an educational context optimally is to understand the target group. This is a key factor

## 2.2. Gamification

in realising an education program that is about to use gamification. A good understanding of who the student is, is necessary. Factors like age, learning abilities and skills are necessary to know. Furthermore, the context of where the gamification should be applied can lead to more details about the students. For example, if a course at a university starts right after lunch, students lack focus. Also, the type of educational context is important to know when applying gamification. In this step, it should be clarified whether the learning program takes place in groups, what the size of the group is or whether every student solves problems on their own. Besides, it is important to know whether the course takes place in a classroom, in an office, at home, etc. Hsin-Yuan and Dilip (2013) also mention the "Common Pain Points in Education". These points include "Focus", "Motivation", "Skills", "Pride", "Physical, mental and emotional factors" and "Learning environment & nature of the course". These five points should be respected in the first step of understanding the target group and the context. (Hsin-Yuan and Dilip, 2013, p. 7-8)

- "*Defining learning Objectives*". This is the second step that needs attention to accomplish the goal of gamification in education. This step describes, that the instructor needs to be clear about what she/he wants to accomplish by completing an educational program. The gamification in education has to come with an objective that the student should achieve at the end of the learning program. This leads to general instruction goals, which for example can be an assignment, a test, a quiz, an exam or a project. Specific learning goals lead to the fact that the student understands a concept that allows her/him to perform a task after training or completing a learning program. The last possible objective includes behavioural goals which lead to the state that the student has to concentrate in class, complete assignments faster or minimise distractions. In sum, the instructor of an educational learning program has to be clear about the objectives she/he wants the student to achieve during a learning program that includes gamification. (Hsin-Yuan and Dilip, 2013, p. 9)
- "*Structuring the Experience*". This step includes the approach of structuring the learning program and breaking it down to smaller tasks.

## 2. State of the art

This can lead to the implementation of stages and milestones. Milestones are a handy tool that make it possible to define a learning task. Each milestone can include a part of the learning program. The milestones can build upon each other and this leads to the fact, that a student has to accomplish one milestone to be able to get to the next one. Furthermore, implementing milestones has the advantage that the students can accomplish one task after another, instead of seeing only the end of the project. This makes the goal of the learning program seem more achievable. Furthermore, the advantage of the instructor is, that she/he can measure the progress of the students. If students lack motivation or are not able to accomplish a task or a milestone,

the milestone or break it down to smaller milestones. Furthermore, the instructors can interact with the students to understand where the problems are and how she/he can assist the students in accomplishing the milestones and reaching the overall goal. (Hsin-Yuan and Dilip, 2013, p. 9-11)

- *"Identifying Resources"*. This step defines the resources needed to implement gamification in education. After the learning objectives and milestones have been defined, an instructor can identify which tasks and milestones can be gamified. Furthermore, this step defines how the milestones can be gamified. There are five steps that an instructor should focus on. The first is about a tracking mechanism. This defines whether it is necessary to track the progress of students and which tool can be used to measure the student's progress. In combination with this step comes the next step, the currency. This defines the unit of measurement. Some units of measurement can be points, credits, a leaderboard, etc. For example, when implementing milestones, a good measure unit would be time. This means students have to accomplish a milestone before a certain date. The next step is level. This defines, after which amount of currency the student is allowed to move forward within the learning program. With the example of time: If a user has accomplished a milestone, she/he is allowed to move forward to the next milestone. This is a type of "leveling up". The next step defines rules for the gamification. The rules describe what a student can or cannot do during the learning task and to achieve milestones.



## 2.2. Gamification

This introduces fairness and equality for all students. For example: When defining a quiz as a milestone, the student has to reach at least 80% to accomplish the milestone. The fifth and last step includes feedback. This defines how the instructor and students can get feedback about the learning progress. This also includes the quality of the tasks completed. (Hsin-Yuan and Dilip, 2013, p. 11-13)

- *"Applying Gamification Elements"*. This is the fifth and final step that an instructor needs to focus on to successfully implement gamification in education. This step defines what elements of gamification should be applied to the milestones and the learning task. In general, two types of game mechanics can be applied: Self-elements and Social-elements. Self-elements can be points, levels, time restrictions and everything where students are competing with themselves and can achieve progress "against themselves". Social-elements define game mechanics that include interaction with other students. This can include group games or a leaderboard where every student can achieve points and compete against other students. That way, the students can check their progress compared to other students. The game element chosen can lead to different reactions from the students. The instructor has to be aware of all the four steps mentioned earlier to make the right choice for a game element. If the steps are not applied correctly, this can, for example, lead to the fact that some students lack the skill to accomplish a game element and therefore are not able to proceed in the learning task. This can lead to frustration and leads to a lack of motivation, which is exactly the opposite of what gamification should lead to. Furthermore, the game elements should be applied in a way that they are equal for all participants. An example of that is a leaderboard. A leaderboard is used during a milestone. This milestone, for example, includes programming software. When uploading software to a verification system that tests the software, the student gets points for the number of tests the software passes. The leaderboard is refreshed in a certain time interval and the student can see where she/he stands compared to other students. This leaderboard and verification system has to be accessible at any time. For example, if the verification system is only accessible in the evening it can be a drawback for some students because maybe some of them are working in the evening.

## 2. State of the art

This has to be respected in terms of equality. (Hsin-Yuan and Dilip, 2013, p. 13-14)

### 2.3. Similar applications

This chapter provides a list of applications that represent or include learning diaries. Some aspects of the applications can be applied to the context of this master thesis. The applications show how a learning diary in a digital environment can be implemented and how it will lead to success.

#### 2.3.1. Milk

Milk is described as an interactive learning kit and a student work diary app. The producers promise an increase of student engagement, parental support, reducing administration and workloads for teachers, improving school communications and learner-centered interaction. The application is available on Android and iOS and is also web-based, which means that students do not need a smartphone or a tablet to join the school community. Milk reduces the administration required by teachers. Furthermore, students receive a push notification whenever a teacher sends out homework or messages. In combination, a teacher can verify whether a student has read the message or the homework task. Teachers can also receive feedback via the application. Students can rate homework and tasks, so the teacher can adapt to the feedback. Students may think that the given homework was too difficult or not appropriate. The teacher can adapt to the feedback and change the homework according to the feedback given by the students. Milk enables teachers, parents and students to see the students' progress. Schools and teachers can provide accessible feedback on the progress of the students to the parents. This leads to the advantage, that parents can adapt to the students' progress in school more often instead of going to a parents' evening once in a while. This improves the indirect communication between school and parents. The messaging system of Milk is based on groups and individuals. Teachers can create groups. An example of a group is a class. If a teacher wants to send a message to this certain class, the teacher writes

## 2.3. Similar applications

a message to the group which represents this class. Furthermore, imagine there is a class with boys and girls in it. The sports teachers may just want to access the boys for instruction since the gym class may be separated by gender. This way, the sports teacher can send a message only to the males in the class and this avoids female students from receiving unnecessary information. As mentioned earlier in this section, Milk is also web-based. This means, that teachers cannot only access the application with their smartphone, but also with their laptops. This makes it possible for the teachers to work at home and send out homework at any time. This leads to the fact, that students can receive push notifications on their smartphones at an inappropriate time. For this issue, Milk makes it possible for schools to define restrictions for push notifications and the application itself. For example, the school can define that push notifications can only be received within certain time slot, like from 08:00 until 16:00. According to a study, the homework submissions went up by 44% and punctuality increased by 87%. To sum it up, Milk provides the possibility to track the progress of students. This progress can be viewed by the teachers and parents if allowed by the school. This makes it possible for the teachers and parents to react and adapt to the students' progress. Since students can give feedback about homework, the teacher cannot only react to the progress of the students but also the direct feedback. With push notifications, students can automatically receive new messages in an instant. The features lead to an overall improvement of homework, presence and punctuality. (Dowling-Feet, 2015) Figure 2.1 shows a screenshot of a part of the application Milk.

### 2.3.2. Seesaw

Seesaw is a program which is available for smartphones as mobile applications or as a web application. It is a tool that is used by schools and teachers to make it possible for students to create and reflect on learning progress. This means, that Seesaw is able to track down the learning progress of students and that teachers are able to define at which education level the students are. Students can use photos, videos, drawings, text, PDFs and links to provide further details about their educational level. Furthermore, Seesaw enables students to share and collaborate. Due to this feature, teachers can

## 2. State of the art

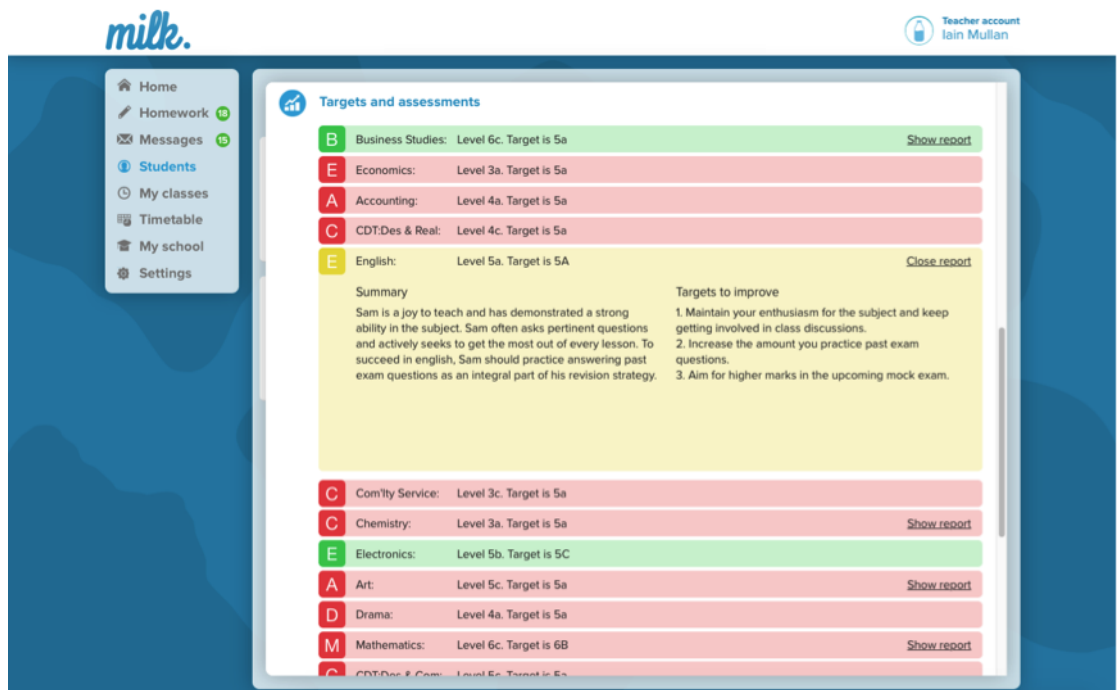


Figure 2.1.: Screenshot of Milk showing targets and assessments (Dowling-Fleet, 2015)

### 2.3. Similar applications

enable whether students have access to the learning progress, drawings, texts, PDFs, links, photos and videos of other students. This application also includes the parents of students. This makes it possible for parents to be up-to-date about the progress of their children. Parents have the possibility to stay in contact with teachers, so parents and teachers can provide a customised learning program for the students. Apart from the possibility to communicate directly, Seesaw also provides communication within the application. Teachers are able to provide encouragement, criticism and suggestions for improvement to motivate the students. It is also possible for parents to comment on the learning progress and digitised documents within the application. The application can be used for assignments and assessments as well. Teachers have the possibility to push assignments to students via Seesaw. The students then complete the assignments and push them back via the application. This way the teacher can give feedback directly after evaluating the assessment. Homework, assignments and assessments can be completely processed via the application. (Seesaw 2019) Seesaw provides a study about the efficacy of Seesaw for schools, the digital portfolio and the parent communication platform on their website. The key areas examined by the study are "student academic performance and development of 21st Century skills", "parent engagement and involvement", "the extent to which Seesaw for schools contributed to overall school objectives" and "administrator satisfaction". The study is supported by a survey carried out during the evaluation process of the study. 400 administrators using Seesaw for their schools were surveyed. Possible answers to the survey were "Agree" and "Disagree". The following data was collected by the study and includes the survey and the given feedback. (Seesaw, 2016)

- *"I have a better understanding of where students are academically because of Seesaw."* Administrators agreed: 92%. Administrators disagreed or did not give feedback: 8%. (Seesaw, 2016)
- *"Seesaw for Schools effectively demonstrates our students' progress over time."* Administrators agreed: 89%. Administrators disagreed or did not give feedback: 11%. (Seesaw, 2016)
- *"Seesaw supports developing 21st Century skills, like collaboration, creative and critical thinking, and digital literacy at my school."* Administrators

## 2. State of the art

agreed: 95%. Administrators disagreed or did not give feedback: 5%. (Seesaw, 2016)

- *"Seesaw helps us use technology resources effectively at our school."* Administrators agreed: 97%. Administrators disagreed or did not give feedback: 3%. (Seesaw, 2016)
- *"Seesaw for Schools helps us develop a better relationship between our school and parent community."* Administrators agreed: 97%. Administrators disagreed or did not give feedback: 3%. (Seesaw, 2016)
- *"I have seen an increase in parent involvement since using Seesaw for Schools and can use Seesaw data to track parent engagement."* Administrators agreed: 91%. Administrators disagreed or did not give feedback: 9%. (Seesaw, 2016)
- *"It's easier for my teachers to communicate with parents using Seesaw than other methods we've tried."* Administrators agreed: 88%. Administrators disagreed or did not give feedback: 12%. (Seesaw, 2016)

Figure 2.2 shows a screenshot of the Seesaw application. It provides a general overview of the application and its features.

Seesaw and Milk are applications that include a learning diary for students. Seesaw and Milk demonstrate, how a learning diary should be implemented. Furthermore, the surveys based upon the applications give proof of the success. This shows, that a learning diary as a mobile application can perform and that the applications of this master thesis can lead to success and the improvement of the learning progress of the students. Milk and Seesaw are showcases to refer to when creating the prototype which is described in chapter 3.

## 2.3. Similar applications



Figure 2.2.: Screenshot of Seesaw (*What is Seesaw?* 2019)





## 3. The prototype

This chapter explains the ideas which came up during the design and prototype process and how the prototype developed to the final result over time. In this context, a prototype means the creation of a detailed conception of how the application will look and work. The prototype is a mockup<sup>1</sup> which provides all screens of the application and links the screens. This is useful to provide a real showcase.

### 3.1. Idea

The idea was to provide an application in Android and iOS to assist the students of a MOOC platform. The application should be able to keep track of all courses the student is assigned to. Furthermore, it should provide the possibility to check upcoming events of the courses and remind whether one event is about to take place. Courses can be of different kinds and in different languages. There are courses which can be done at any time. On the other hand, there are courses on the MOOC platform which can only be done in a certain time span. Therefore some events can be done at any time and other events have a specific date where they have to be finished. This is a fact which had to be considered during the prototype process. Another part which had to be considered during the prototype process was the navigation of the app. The goal in terms of navigation is to provide the user an easy-to-use application that is based on the platform-specific operating system. This means, that the navigation behaviour of the application does

---

<sup>1</sup>"In manufacturing and design, a mockup, or mock-up, is a scale or full-size model of a design or device, used for teaching, demonstration, design evaluation, promotion, and other purposes. A mockup is a prototype if it provides at least part of the functionality of a system and enables testing of a design." *Mockup* (2019)

### 3. The prototype

not clash with the standard navigation of the operating systems Android and iOS. The first idea was to provide a so-called "hamburger menu". A hamburger menu is a navigation type, where three lines, aligned vertically, are based on the top of the screen. If a user clicks on these three lines, a navigation menu pops up, where the user can choose where she/he wants to navigate to. The problem is, that iOS does not provide a native hamburger menu. This clashed with the fact, that the application should align with the behaviour of the operating systems. This means, if the iOS application would have had a custom-build hamburger menu, the navigation would not align with the navigation the user is used to on an iOS device. The next and final idea of navigation was to build a tab-based app. This means, the navigation menu is visible all the time and not hidden behind an icon like with the hamburger menu, which is saving the user one click to reach the navigation. A tab-based application uses a navigation menu, which is positioned at the bottom of the screen. There are some icons, in case of this application four icons, which are placed next to each other in a horizontal alignment. The icons are described with a short description. In the case of this app, the descriptions are "Events", "Courses", "Notification" and "Profile". The idea of why these four parts were chosen for the navigation is explained in the section 3.3.

After the main idea and the navigation was settled, there were some further features to ensure a satisfying user experience:

- Colouring of courses and events
- Notifications
- Creating events
- Creating learn goals
- Creating notes
- Filtering events

The following sections will describe these features in detail and why they are helpful for the user and can support her/him increasing her/his learning process.

### 3.1.1. Colouring of courses and events

The idea of the feature is that students of the MOOC platform should be able to differentiate between different courses and events at first sight. This means that the border, the icon and the progress view in the items in a list view, seen in section 3.3, are drawn in a colour the user chooses. The user can define a colour for a course in the "Profile" section of the app. If the user clicks on "Color" in the "Profile" section, all her/his courses are listed. The user can choose a course from this list and can define a colour for this specific course. All the events, learn goals, notes and the course itself are then painted in this colour in the app. The idea for this feature was the following: If one user is only assigned to a few courses, this feature may not be necessary. If a user is assigned to some courses, this may come in handy. The reason for that is that for example in the section "Events", where all events for all courses are listed, a big number of events may pile up. When the user is defining colours for her/his courses, the user then can differ which event belongs to which course at first sight. To make sure the user does not forget which colour is assigned to which course, the courses in the section "Courses" are also painted in this colour. Besides this type of colouring, there is one more feature that uses colouring. When a user opens the application and checks the section for "Events", all events are listed, no matter whether future, present or past ones. The events are listed with their due date. To provide the user the possibility to check whether events are past the current date without checking the due date, past events are coloured in light grey. The user can check at first sight, whether the due date of the event is already past and whether she/he has completed the task for this event.

### 3.1.2. Notifications

The idea of this feature is, that students of the MOOC platform receive notifications, if they have set a "reminder date" to an event or a learn goal or if a new course is upcoming. The user has the possibility to set "reminder dates" for events and learn goals. If a user chooses an event, the application navigates to a detailed view of the event. The user then can set

### 3. The prototype

a "reminder date" for this specific event on this detailed view. When the date is about to come, the user gets a push notification to her/his device, so the user can make sure that she/he does not forget to complete the task of this event. This feature is possible with events that are received from the backend of the MOOC platform or with events and learn goals created by the user her-/himself. If a user decides not to complete the task on the day she/he is reminded, the user can set a new "reminder date" for this event. It is not possible to set a "reminder date" for notes. Notes only provide the possibility to store text, ideas, hyperlinks, etc. and should only provide information. Furthermore, the application also notifies a user if a new course of the MOOC platform is about to start. This makes sure, that the user does not miss any chance to assign to a new course which she/he is interested in. The user does not have to check the website of the MOOC platform for new courses. The user can rely on the application to remind her/him that a new course is about to start. If a user gets notified and opens the app, the section for "Notification" in the bottom navigation bar is badged. This means, that there is a small red circle with an exclamation mark at the top of the "Notification" icon which indicates that there are new unseen notifications. If the user checks the "Notification" section, she/he can see all notifications she/he has received, including reminder notifications and notifications for new courses. Each of these notifications has its own specific text and icon, which can be seen in section 3.3. If the user clicks on one notification in the list, she/he either navigates to the detail view of the event, the detail view of the learn goal or the URL of the new course. If a user navigates away from the "Notification" section, the badge which indicates new notifications is deleted, because the user has seen all new notifications. This makes sure that the badge only appears when there are unseen notifications.

#### **3.1.3. Creating events and learn goals**

The idea of this feature is, that students of the MOOC platform can create their own events and learn goals for their courses. For a self-created event or a learn goal, the user has to provide some information about it. The minimum information about the event/learn goal includes a title for the

### 3.1. Idea

event/learn goal, a due date where the event/learn goal should be finished and the course the event/learn goal belongs to. In terms of the MOOC platform, it makes sense to link an event to a course because the MOOC platform works in the way that a course has at least one event and an event belongs exactly to one course. The MOOC platform itself does not include learn goals. Learn goals are only available inside the application. The idea behind the learn goal is, that the user can set milestones. An event should be used for tasks.

To demonstrate the difference between an event and a learn goal: The event should be used, when a user is planning to do something. For example, the user is programming an app. Someday, she/he wants to design the login screen. On another day, she/he wants to program the logic of the login screen. On another day, she/he wants to program the communication of the application with a database. Before doing these three tasks, the user can set a learn goal. In this example, it may be called "Login Screen finished". After the user has finished all three tasks, she/he may set the learn goal to "done", so the user knows she/he has completed the tasks, the learn goal and can receive experience points for it. Experience points are explained in section section 3.2.

In the application itself, the self-created event/learn goal acts like the events which are received from the backend of the MOOC platform. The only difference is, that the user can delete her/his self created events/learn goals. This is not possible for the events coming from the MOOC platform. The user can provide additional information to the self-created event/learn goal. This additional information includes a description for the event/learn goal, a time when it should be finished and a "reminder date", explained in subsection 3.1.2. Besides creating events/learn goals to increase and keep track of the learning progress, it has another side effect. This side effect includes gamification. When a user finishes an event, she/he gets experience points that reflect the progress of a user. A detailed explanation about the idea of gamification in this application is given in section 3.2

### 3. The prototype

#### 3.1.4. Creating notes

The idea of this feature is, that users, besides creating events and learn goals, can create notes within the application. Notes are intended for writing down and saving thoughts, links, ideas and suchlike. Notes are not available on the MOOC platform. This means, that the notes are not received from a backend but can only be created within the application. To create a note, the user, as with events and learn goals, has to provide some information. Besides events and learn goals, where the user has to provide information such as title, due date and course (for more see subsection 3.1.3, she/he only has to provide a title and a course it is assigned to. The first thought was, that a user can assign notes only to events, but this leaves the user inflexible with information. For example, the user wants to provide a note which is helpful for every event in a course. So the user has to add this note to every event or learn goal in the course. This is inefficient. The solution was, that the user can provide notes, not for specific events, but the overall course. If a user wants to remember himself that a note is only important for one event, she/he can define that in the description of the note. For notes, the user can only provide the information for a title, a description and a course it is assigned to. Title and course are obligatory. In this application and the way it is built, it does not make sense to set a reminder date, a time or a due date for notes. As explained before in this section, the feature of creating a note should only be used for saving thoughts and information. Notes should assist the events and learn goals within a course. Therefore it makes no sense to set a due date for a note. There is no time limit for a note. Furthermore, it does not make sense to provide a reminder date. A user should not get reminded for a note, only for events and learn goals within a course and new courses that are available on the MOOC platform. If a user has the opinion that a note is not current, no longer relevant or creating it was a mistake, the user can delete the note from the course. Additionally, because a note is only assigned to a course, it should not be visible in the detailed view of an event or a learn goal. The note is only visible if a user chooses a course in the related section "Courses" within the application. In the detailed view of the course, the user can check all the events, learn goals and notes related to the selected course. For detailed visualisation check the section 3.3

### 3.1.5. Filtering events and automatic scrolling

The idea of this feature is, that users can filter their events according to three different types. If a user starts the application, she/he lands on the start page of the navigation menu which is the section for "Events". This section contains all self-created events and learn goals and those which are received from the MOOC platform. If a user is assigned to a number of courses, this view can be confusing. To assist the user in keeping an overview of the (current) events, the idea of the following feature came up during the prototype process. The user should have the possibility to choose between three different types of events. The first type is "uncompleted" events, which include all events and learn goals, no matter whether self-created or not, that have not been fulfilled or completed by the user yet. The second type is "completed" events, which includes all events and learn goals, no matter whether self created or not, that have already been fulfilled or completed by the user. The third type is "all" events, which include all events and learn goals, no matter whether self-created or not, that exist for all courses. Additional to this feature, the first event that comes into sight if the user starts the application and lands on the section for "Events" is the event or learn goal, which is nearest to the current date, ignoring past events.

For example, the user starts the application on the 1st of March 2019. If an event or learn goal exists with a due date of the 1st of March 2019, the application should automatically scroll to this event. If no event or learn goal exists where the due date is the current date, the application should automatically scroll to the event which is the nearest in the future. This means, if the user starts the application on the 1st of March 2019 and two events exist, which have the due dates of 4th of March 2019 and 28th of February 2019, the application automatically scrolls to the event with the due date of 4th of March 2019.

This feature should help the user to quickly check her/his upcoming events and learn goals and not have to scroll down to find the most current one. In general, these two features combined should help the user to keep an overview of the events and not have to spend time on finding the events she/he is searching for. With the ability of the application to scroll down to

### 3. The prototype

the most current events and the ability to filter the events by "uncompleted", "completed" and "all", the application should provide the experience of keeping an overview of events and learn goals for the user and speed up the finding process.

## 3.2. Gamification

Besides the fact that the application should be able to keep an overview of all events and courses and provide the features of creating events, learn goals (see subsection 3.1.3) and notes (see subsection 3.1.4), colouring of events, learn goals and courses (see subsection 3.1.1), notifications (see subsection 3.1.2) and filter events (see subsection 3.1.5), there was the task during the prototype process to provide a concept of gamification<sup>2</sup>. The user should be rewarded for completing events and learn goals.

The first idea to fulfill this requirement was to bring in avatars. The user should receive some type of currency, like coins or experience points, for every event or learn goal completed. With this currency, the user should be able to purchase clothes, appearance styles and accessories for her/his avatar. The idea was, that the user can set the primary basic points of his avatar, like skin colour, gender, clothes, hair and face at the first start of the application. Over time, the user can improve her/his avatar by purchasing new stuff for her/his avatar with the app-specific currency. This type of gamification should motivate the user to keep on learning and complete learn goals and tasks in events, so the user can improve her/his avatar. Furthermore, the application should provide the possibility to export the avatar as a picture, so the user can share it within the forum of the MOOC platform or share it with whoever she/he wants to. After thinking of how this type of gamification can be implemented and discussion with heuristic experts, it came up that it would not suit the application and the MOOC platform. Therefore, there was the requirement to come up with an alternative idea of how gamification can be solved within this application.

---

<sup>2</sup>"Gamification is the application of game-design elements and game principles in non-game contexts." *Gamification* (2019)



### 3.2. Gamification

The next idea to fulfill this requirement was to bring in a reward system with experience points and levels. This means, for every task completed by the user, the user should receive some amount of experience points. If a user reaches a certain amount of experience points, the user increases her/his level. The number of levels should be infinite. Besides the fact that every user should have "learning levels", there somehow should be a visual reward for the user. Just showing the experience points and the level is not enough. Therefore, the idea arose to bring in a replacement for the avatar. The solution was to define images for every level. Of course, it is not possible to define images for an infinite amount of levels, but a certain amount of levels. If a user exceeds this certain amount of levels, the image for every level earned stays the same. The first idea was to bring in images like cats, dogs and so on. The second idea was to bring in some designed image with the level of a user on the picture and a different icon and background colour. Still, providing the visualisation of the experience points, the current level and the corresponding image for the current level of the user is not enough to fit the gamification requirements of the application. Therefore, an individual text for every level up to a certain amount of levels should be provided. An example text for a level to get a feeling of how this text should look: "Level 1. You're just getting started." or "Level 12: You're the master of learning."

To improve the user experience with gamification within the application, a start screen depending on the current level of the user should be included. This means, if the user starts the app, the application shows a screen with an image covering the whole screen. The image depends on the current level of the user. As with the showcase of the experience points, the level, the text for the level and the corresponding image in the section for "Profile", the image for the start screen shows the current level of the user, the icon and the text for the current level. This should motivate the user right from the start. If a level changes, the user automatically sees her/his progress at the start of the application and within the application itself in the section for "Profile".

To sum up the idea of gamification regarding this app: The application implements a system of gamification that includes experience points and

### 3. The prototype

levels. To reach a higher level, the user has to earn experience points by completing events and learn goals. To encourage the user to increase the level, the application provides individual visualisation for each level in the form of a image and a text that should motivate the user. The visualisation of the current level of the user is shown in two parts within the application. The first part is in the section "Profile". The second part is on the start page of the application.

## 3.3. Mockup

This section points out the mockup which was generated based on the ideas explained in section 3.1 and in section 3.2. A mockup in the area of software engineering/development, is defined as a prototype that shows the end-user or a client the user interface without building the software or features of the software<sup>3</sup>. In the case of this app, the mockup includes all screen designs based on the ideas generated earlier in the prototype process and some vestigial functionality. The functionality is the linking between the screens of the application. The goal was that the user of the mockup gets a feeling of how the application will work like by clicking at specific icons on the screen leading to different screens. One basic example of this functionality would be the basic navigation described in section 3.1. If someone uses the mockup, she/he can click on the icons/elements of the tabbed navigation, placed at the bottom of the screen, to get a feeling of how the main navigation in the application will work. The mockup just shows a basic concept of the app. The application was changed later on in the developing process, due to technical difficulties in implementation or the chance to improve usability and the user experience. The concept of the application described in this chapter does not completely reflect the final result gained in the developing process, described in chapter 4. The mockup created during the prototype process was discussed with heuristic experts. It was a process which developed a result by receiving feedback and implementing ideas gained by the heuristic experts.

---

<sup>3</sup>Mockup 2019.

### 3.3. Mockup

The figure 3.1 shows the main screen containing the main navigation. The main navigation is located at the bottom of the screen. The bottom navigation contains four sections. The first section from left to right is "Events", which is shown in the 3.1. The second section is "Courses", which contains a list of all courses the user is assigned to. The third section is "Notifications", which contains a list of all notifications the user has received. The fourth and last section is "Profile". This section contains the visualisation of gamification described in section 3.2, the feature to colour courses, described in subsection 3.1.1, and some links to the MOOC platform. Furthermore, figure 3.1 shows how events will be displayed to the user. This can be seen

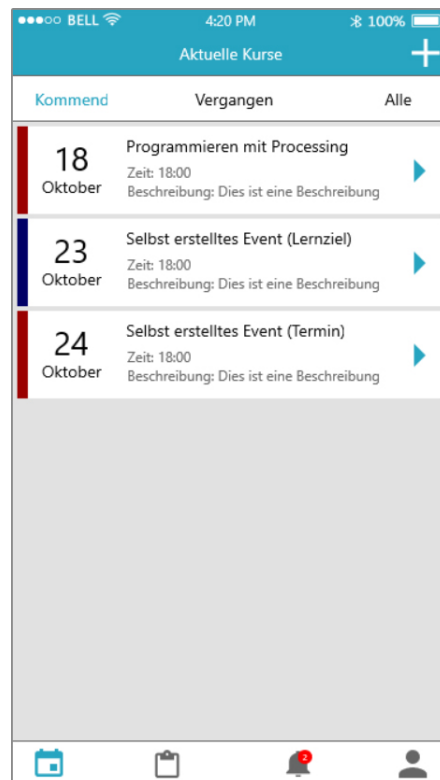


Figure 3.1.: Design of listing events and learn goals to the user

above the bottom navigation in the Figure 3.1. An event holds information for a title, a description, a time and a due date. The due date is displayed on the left side of the item. The day of the due date is shown as a number

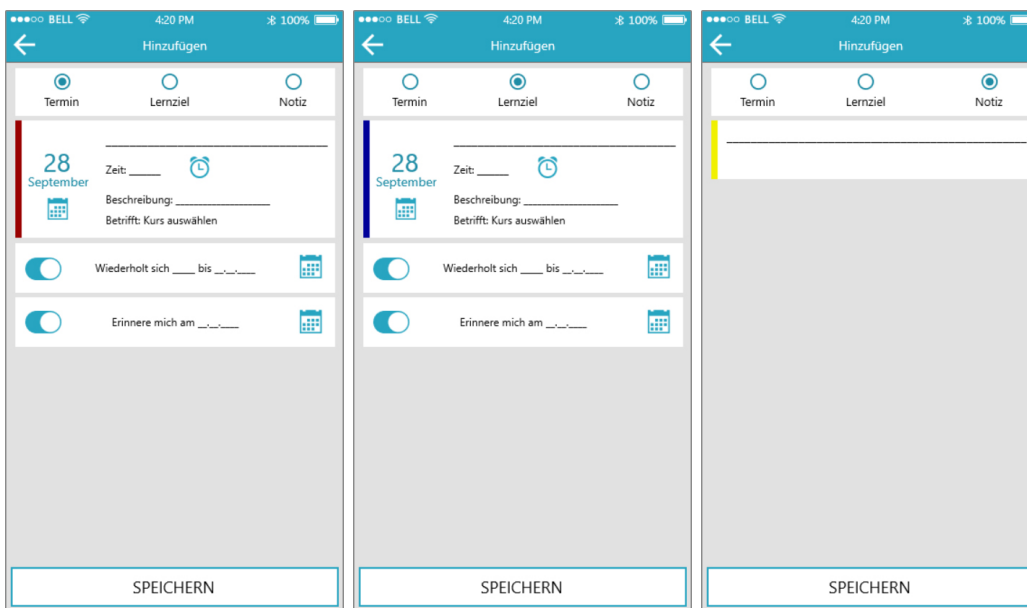
### 3. The prototype

and larger than the rest of the text of the item. This is due to the fact, that the user should be able to focus on the due dates of the events to keep an overview of upcoming events. The month of the due date is shown below the day, written in written form. To the right of the due date, at the top of the item, there is the title for the event. Below the title, there is the part for the time. The time section describes the time when the event has to be finished on the due date given. Below the time section, there is the section for the description. The description contains further information about the event, for example, what the event is about and what the focus is on. On the right side of the item, there is an icon. This icon, in 3.1 shown as an arrow, is a placeholder for different icons. This icon describes the type of item. There are three different types, as described in section 3.1. The event, the learn goal and the note. Each type has a different icon. This should help the user to identify which type of event an item is in the list view. To help the user identify which events belong to which courses at first sight, there is the feature to colour courses and events, described in subsection 3.1.1. The colouring is visualised with the items listed in figure 3.1. On the left side of the item, there is a rectangle. This rectangle will be painted in the colour the user chooses for the course of the event in the "Profile"-section of the navigation.

In figure 3.1, above the list view for the events, there is another type of navigation bar. This navigation bar includes the items, from left to right, "Kommend" (English: "Future"), "Vergangen" (English: "Past") and "Alle" (English: "All"). This navigation bar provides the feature of filtering events, explained in subsection 3.1.5. The left item "Kommend" in the bar, which is the standard item chosen, will list only events to the user, which are current or about to come. If the user clicks the item "Vergangen" in the middle of the bar, the application will list only events that have already happened. The right item "Alle" will list all events to the user.

In the right top corner of the screen, there is a plus-sign. This plus-sign indicates the possibility to create events, learn goals and notes, described in subsection 3.1.3 and subsection 3.1.4. If a user clicks on this plus-sign, she/he will be redirected to another screen. This screen will be the one to create a new event, learn goal or note. Figure 3.2 shows all three possible screens. Figure 3.2 shows, how the screen of creating an event, a learn goal or a note should look like. At the top of the screen, there is a navigation bar

### 3.3. Mockup



(a) Create event

(b) Create learn goal

(c) Create note

Figure 3.2.: Design of creating an event (a), a learn goal (b) or a note (c)

### 3. The prototype

that contains the title of the screen. In this case, the title is "Hinzufügen" (English: "Add"). The navigation bar also contains a "back-button". If the user clicks on it, she/he is redirected to the screen where he/she opened the "creation screen" from. Below the navigation bar, there are three radio-buttons<sup>4</sup>, which are aligned horizontally. The default radio button chosen is the one for creating an event. If a user chooses to create a learn goal or a note, the screen changes depending on the chosen type. In all three different types, there is one information that is obligatory for all of them. The obligatory information which has to be provided by the user is a title for the event, learn goal or note. Furthermore, they all share a coloured rectangle. The colour of this rectangle changes, depending on the type of event chosen. It is used for visualisation. events and learn goals share the same information which have to or can be provided. The only difference between these two is the way how it is handled within the app. A detailed explanation is given in subsection 3.1.3. The following explanation is given for the screens shown in figure 3.2a and 3.2b, because they share the same information. The screen shown in figure 3.2c is explained afterwards. In figure 3.2a and 3.2b, below the section for radio buttons, there is the section to provide information for an event or a learn goal. It looks similar to the items in the list view, shown in figure 3.1, to help the user understand how it is visualised after creating it. Going from left to right within this section, the first item within there is a rectangle, which has already been explained above. After that, there follows the due date. The due date is indicated with a day and a month. Below the month, there is an icon that represents a calendar. If a user clicks on this calendar, a date picker is shown and the user can choose a date for the event. To the right of the due date, there are five text areas aligned vertically. The one at the top is obligatory and the user has to provide a title for the event there. The second one, from top to bottom, is the part for the time. This part is optional. If a user clicks on the icon which represents a clock, she/he can choose a time when the event or learn goal has to be done on the given due date. The third one in this section is the part of the description. This part is also optional. The user can provide some information there, which may sum up the task(s) to do within this event. The fourth and last one in this section is the part of the course. This part

---

<sup>4</sup>"A radio button or option button is a graphical control element that allows the user to choose only one of a predefined set of mutually exclusive options." *Radio button* (2019)

### 3.3. Mockup

is obligatory. If the user clicks in this text area, she/he can choose from all courses she/he is assigned to. This assigns the event created to the chosen course. Further details are given in subsection 3.1.3. Below the section for the information of the event, two more sections allow the application to notify the user for this event. The first of these two sections was an idea for a feature that was abandoned due to complexity. The idea was, that the user can tell the application to create the same event for an interval. For example, the user could have told the application to create the event all seven days until the 31st of December 2019. After thinking through how this feature could be implemented, it was abandoned because there was a limit regarding the hours invested in this master thesis. The second of these two sections represents the "reminder date", explained in subsection 3.1.2. If a user turns on the switch<sup>5</sup>, which is one the left side of the section, she/he can set a "reminder date" for this event. This means, that the user gets notified about this event on the date she/he is setting there. If the user clicks on the icon on the right of the section, which represents a calendar, she/he can use the appearing date-picker to choose a date. The screen shown in figure 3.2c indicates the screen for creating a note. The user has to provide the information for title and a course, like with creating an event or a learn goal. The part for the course is missing in the figure. Furthermore, the user can describe the note, which should contain information and details about the note. The creation process of a note is explained in subsection 3.1.4. At the bottom of the screen, there is a button with the text "SPEICHERN" (English: "SAVE") in it. If a user clicks this button, the application should check whether all necessary information is provided, depending on the type of event chosen. If there is some information missing, the user should get informed about the missing parts. If the necessary information is provided, the event, learn goal or note is saved.

Looking at figure 3.1, the next section in the bottom navigation is the section for "Courses". Figure 3.3 shows this section. At the top of the screen, there is the title bar, which contains the title for the screen. The title says "Meine Kurse" (English: "My courses"). The screen contains a list view with items. These items represent the courses the user is assigned to. One item contains four elements. The first element, which is one the left of an item, is an

---

<sup>5</sup>"A Switch is a two-state toggle switch widget that can select between two options." *Switch* (2019)

### 3. The prototype

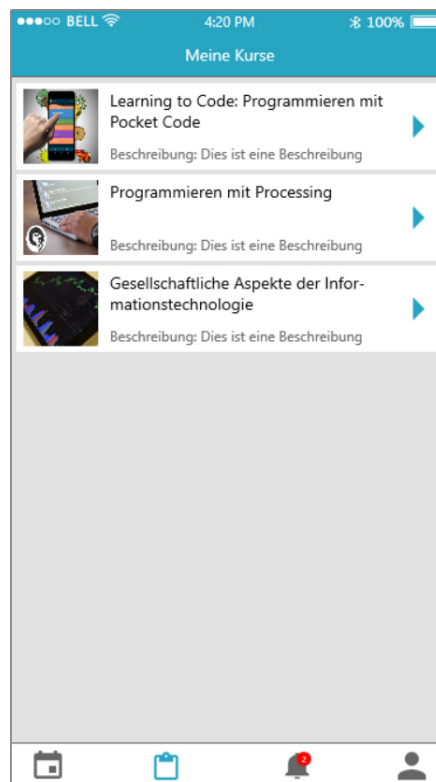


Figure 3.3.: Design of listing subscribed courses to the user



### 3.3. Mockup

image. This image is the picture for the course, which is also provided on the MOOC platform. To the right of the picture, in black font, there is the title of the course. In the case of 3.3, the title of the first item is "Learning to Code: Programmieren mit Pocket Code". Below the title, there is the part for the description. This description contains further information about what the course is about. At the right of the item, there is an icon. This icon indicates, if a user clicks on this icon, she/he will be redirected to a new screen.

The screen the user is directed to, is shown in figure 3.4. The three screens shown in this figure, provide all events, learn goals and notes which belong to the chosen course. So this is, next to filtering described in subsection 3.1.5, one more way to filter the events, learn goals and notes. The following

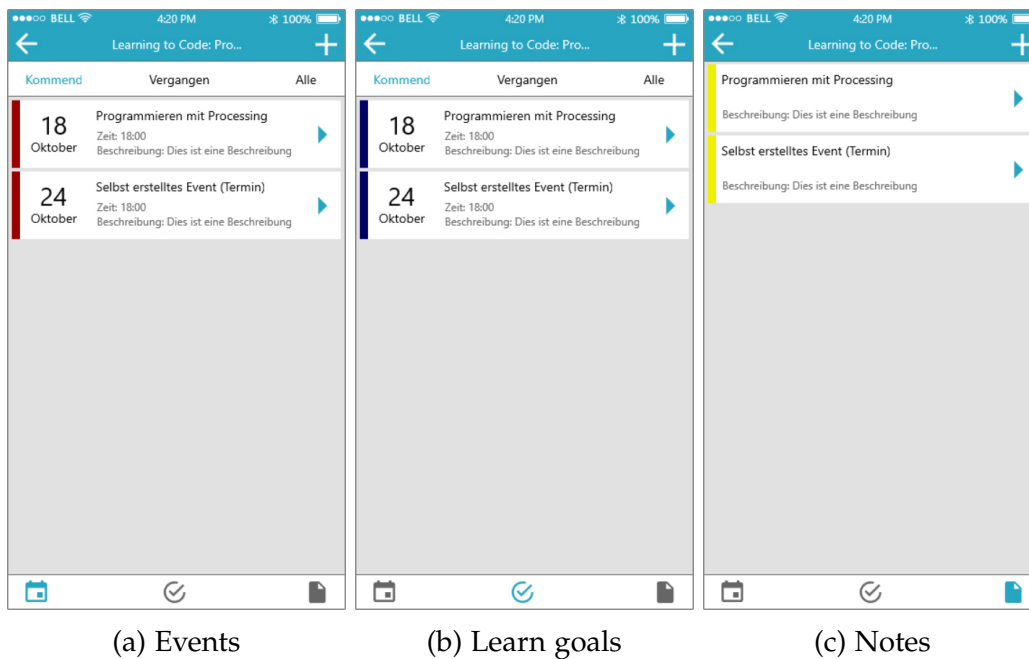


Figure 3.4.: Design of listing events, learn goals and notes of a specific course to the user

explanation describes the elements in the screens shown in figure 3.4 from bottom to top. The bottom navigation bar shown in the figure is different from the bottom navigation bar shown in figure 3.1. There are three elements in this navigation bar. From left to right, there are the icons for events, learn

### 3. The prototype

goals and notes. When directed to this screen, the default section chosen is the one for Events. This means, that only the Events for this course are listed to the user. If the user clicks on the section for "Learn Goals", only the learn goals for the selected course are listed to the user. If the user clicks on the section for "Notes", only the notes for the selected course are listed to the user. This behaviour should help the user to navigate through the events, learn goals and notes and differ from each type. Looking at figure 3.4a, above the bottom navigation bar there is a list view. This list view contains all events for the selected course. The items look the same as shown in figure 3.1 and they also provide the same information as given in the explanation for the figure. If a user clicks on an event item, the user is directed to the detail screen of the event. Above the list view, there is another tab bar, which works the same as described in the explanation for figure 3.1. The screen shown in 3.4b works the same, only that the events are replaced by learn goals. The difference in the screen for the notes, shown in figure 3.4c, is that the tab bar for "Kommend", "Vergangen" and "Alle" is missing. This is due to the fact, that there are no notes which have a due date. Further explanation to notes is given in subsection 3.1.4. At the top, in all three figures shown in figure 3.4, there is a top navigation bar. This top navigation bar contains two icons and a title. The title contains the title of the selected course. The icon on the right represents the "back-button". If a user clicks it, she/he is redirected to the screen where all courses are listed. The icon on the left represents a "plus-sign". If a user clicks it, she/he is lead to the screen where she/he can create an event, like shown in figure 3.2, described in subsection 3.1.3 and subsection 3.1.4. The only difference is, that if a user clicks the icon on this screen and not on the screen shown in 3.1, that some information in the creation screen will automatically be filled out for the user. This depends on the section the user has chosen in the bottom navigation bar shown in figure 3.4. If a user has chosen the section for "Events", the application will automatically choose the radio button for "Event". This will lead to the screen shown in figure 3.2a. If a user has chosen the section for "Learn Goals", the application will automatically choose the radio button for "Learn Goal". This will lead to the screen shown in figure 3.2b. If a user has chosen the section for "Notes", the application will automatically choose the radio button for "Note". This will lead to the screen shown in figure 3.2c. Furthermore, the application will automatically choose the course the event or learn goal is assigned to, depending on the

### 3.3. Mockup

course the user chose in the section for "Courses", shown in figure 3.3. The rest of the creation process will work as described in the explanation for figure 3.2.

Figure 3.5 shows the screen for the section "Notifications" within the main bottom navigation bar. The section for "Notifications" will list all notifications which have been pushed to the user. This section includes notifications for reminding the user about events, learn goals or new courses. The items

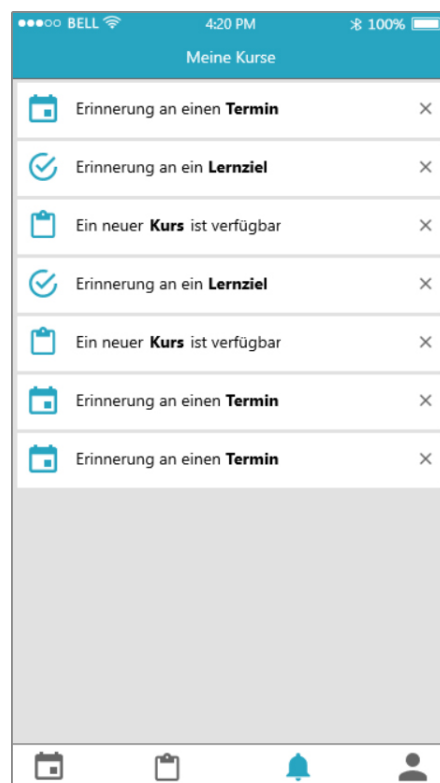


Figure 3.5.: Design of listing received notifications to the user

in the list view shown in figure 3.5 include three elements. At the left of an item is an icon. It describes the type of notification, so the user can differ between the notifications at first sight. There are three different possible icons. The first item from top to bottom represents a notification for an event. The second item represents the notification for a learn goal. The third item represents the notification for a new course on the MOOC platform. On the

### 3. The prototype

right side of the icon, there is a text. It describes the type of notification. On the right side of the item, there is an "x"-icon. This represents the possibility to delete the notifications if not needed anymore. The feature of deleting notifications has been abandoned in the developing process. Further explanation is given in the chapter for the developing process. If a user clicks on a notification, she/he is redirected to a different screen. To which screen the user is redirected to depends on the type of notification. If a user clicks on a notification which is a reminder for an event or a learn goal, the user is directed to the detail view screen of the event or the learn goal. If a user clicks a notification which reminds for a new course available on the MOOC platform, the user is redirected to the browser of the device which opens the URL for the course on the MOOC platform. Further explanation is given in subsection 3.1.2.

The figure 3.6 shows the last screen of the main navigation. It shows the section for "Profile". Some of the elements included in the section have changed during the developing process. This screen was primarily designed to show how the screen will look like. It shows the possibility, that multiple items can fit there if they will not fit somewhere else in the app. At the top of the screen in figure 3.6, there is the part for the gamification described in section 3.2. At the top of this part, the current level of the user is displayed. In this case, it is "Level 1". Below the level, there is a text depending on the current level of the user. In this case, it says "Du bist ein Anfänger" (English: "You are a rookie."). It is a text that should motivate the user and changes every time the user reaches a new level. Below the text for the level, there is a progress bar. This progress bar indicates the experience points needed to reach the next level. As explained in section 3.2, the user can achieve experience points by completing events and learn goals created by the user. In this case, the progress bar shows "350/1000 XP". "XP" stands for "experience points". So in this case, the user has to achieve 650 more experience points to reach level two. On the right side of this part, there is a placeholder for an image in the form of an "abstract human". Already explained in section 3.2, every level has, besides the text, its image.

Below the part for gamification, five items can be used for further information. The first item says "Fortschritte" (English: Progress). If a user clicks on this item, she/he will be redirected to the screen shown in 3.7. Below the item for progress, there are four more items. The items contain the texts

### 3.3. Mockup



Figure 3.6.: Design of the Profile screen including gamification

### 3. The prototype

"Alle Kurse" (English: "All courses"), "Für einen Kurs anmelden" (English: "Subscribe to a course"), "Von einem Kurs abmelden" (English: "Unsubscribe from a course") and "iMooX.at". These items hold information for a URL. If a user clicks on one of these items, she/he will be redirected to a website in the browser of the device, depending on which item she/he has clicked.

The figure 3.7 describes a screen within the application that has been abandoned during the developing process. It was designed to show the user the progress of all of his courses and the average progress of all others subscribed to this course. If a user wants to have further information about the course and its events, she/he can click the item for the course and will be redirected to a screen, which looks similar to the screen shown in figure 3.7. In the following screen, the application lists all events to the user which are related to this course. The application then shows the progress of the user of this event and the average progress of all others who are dealing with this event. The screen in figure 3.7 was abandoned due to the fact, that it was possible to provide this information without using an additional screen. How the visualisation of the progress of courses and events was implemented is described in the chapter for developing app.

The screens and figures described in section 3.3 are the basic concept which has been used during the developing process. Due to technical difficulties in developing or because there are ways to solve features that fit better in this type of app, some screens and features have changed during the developing process. The application described in this section is not the final result. It is a basic concept to start developing with. The mockup, features and ideas were approved by heuristic experts. As defined in a discussion with those experts, the mockup and the features gained during the prototype process are the basic fundamental of the application.

### 3.3. Mockup

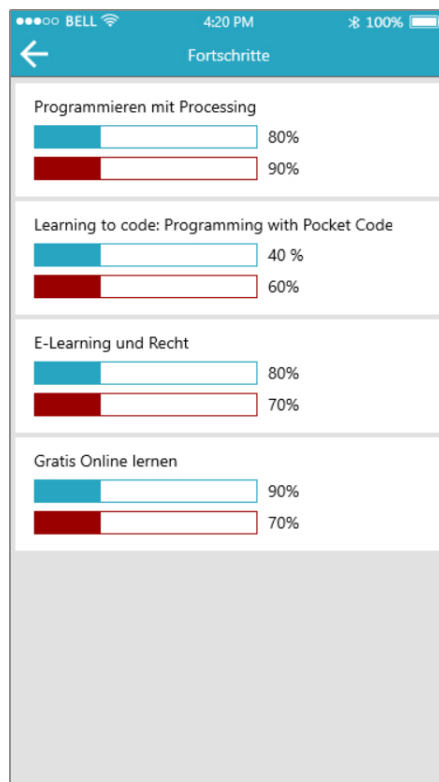


Figure 3.7.: Design of listing progress of subscribed courses to the user





## 4. Developing the applications

After finishing the mockup and the prototype process, the developing of the application followed. The application was developed based on the mockup, which was approved by heuristic experts. The process of developing the application was the most expensive one in terms of time. The task was to develop an application for two operating systems. One application for Android and one application for iOS. The features and the look-a-like of both apps, described in chapter 3, should stay the same, without losing the user experience of the operating systems. This means, if the operating systems provides build-in behaviour for a certain kind of feature, it should be used instead of forcing the application to behave differently. This guarantees the user the feeling that the application was developed only for this operating system. Furthermore, it helps the user navigating through the application and understand it. The apps were developed in succession, not simultaneously.

This was due to the following fact: If one feature changes during the developing process, the developer (in this case the writer of this master thesis), did not have to change the feature in both apps at the same time, but only in the first application. If the features of the first application were set, the developer built the succeeding application in the same way.

The operating system to build the first application for was Android. This was due to the fact, that the developer had more experience in building Android applications than building applications for iOS. The developer finished the application for Android first to be sure, that the second application on iOS will look and work the same way, respecting the features of the operating system.

The goal was to provide a full-functional application for both operating systems which was ready to publish in the application Store and Google Play

## 4. Developing the applications

Store. The task was to implement only features which have been discussed in section 3.1. Some more features would help to increase the value of the app, but could not have been implemented during this master thesis, due to technical difficulties or the amount of time that the features would need. The applications build during this master thesis are build on core features to help the user to keep an overview of her/his events, learn goals, notes and courses of the MOOC platform and increase the learning progress.

The explanation following in this chapter describes the way how the apps are built. Even though every aspect, feature and screen is described, to understand the following, a basic knowledge of Android and iOS application development is required. Special implementation and libraries chosen from the developer are explained more detailed, but to re-create this application with the explanation given in this master thesis, a basic knowledge and/or further research is required.

### 4.1. Strategy

This section describes the strategy followed for both types of applications, Android and iOS. The first question that arose was how the applications should be developed. For this, the provided IDEs<sup>1</sup> Android Studio for the Android version and XCode for the iOS version were used. There exist different models of how an application could be built. In this case, the developer chose to use MVVM(Model View ViewModel)<sup>2</sup>. MVVM is a blueprint of how a project should be set up. MVVM is used to separate the logic of the user interface and the view of the user interface. The Model holds all information for a class and defines the attributes. The ViewModel contains the logic for the user interface and the communication between the logic and the view. This communication could happen with data binding. The logic would be for example a button. If a user clicks a button in the view,

---

<sup>1</sup>"An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development." *Integrated development environment* (2019)

<sup>2</sup>"MVVM facilitates a separation of development of the graphical user interface – be it via a markup language or GUI code – from development of the business logic or back-end logic (the data model)." *Model-view-viewmodel* (2019)

## 4.1. Strategy

the ViewModel will handle what happens after a user clicks on this button. The view contains all elements displayed to the user. Explicit explanation is following in section 4.2 and section 4.3.

Furthermore, there is the fact that the application is communicating with an external database. This is a fact to respect during creating a strategy. In the way the application was supposed to look like, the developer chose to use libraries to assist the communication between the application and the external database. The logic of communication is also happening within a ViewModel. If a user lands on a screen, the ViewModel for this screen loads the needed information of the external database with a request. Due to security issues that could arise when the application communicates with the external database of the MOOC-platform, the developing team decided to only communicate with requests with the method GET. So the application was not able to store any information in the external database. This leads to another fact that should be respected: The application has to store further information on the device of the user.

There are different possibilities to store information on the device of the user. This also depends on the device and the operating system. In this case, the developer chose to use a library that is working on both platforms, Android and iOS. The library is called Realm. It is used to store information in a local database of the device. Further explanation for this library and how to use it is given in subsection 4.2.1 and subsection 4.2.2.

The Login-Screen was also a feature to think about how to solve it. The external database needs an authentication from the application side to respond with corresponding data. Due to security issues, it was not possible to build a native login screen. This is because there is no possibility to communicate in a bidirectional way like explained before. The application cannot proceed with any request except requests with a method of GET. Also, due to security, it was not possible to send an e-mail address and password to the server. So the decision was to use the already existing login view of the MOOC-website itself. How this problem was solved is described in subsection 4.2.2 and subsection 4.3.2.

Seen in the figures in section 3.3, the application includes icons which will represent actions within the application. The first thought was to create icons that are used in Android and iOS. This, however, would collide with

## 4. Developing the applications

the fact that the applications should focus on user experience. Therefore, the developer decided to save time by not creating custom icons and use icons that users already know from other apps. The icons within the app, no matter whether Android or iOS, are all from *Material Design* (2019).

### 4.2. Android

This section describes the development of the Android version. It provides a detailed explanation of how each screen, the navigation, data storage and API communication was implemented. The first to follow is a list of which libraries are used in the application and an explanation of what they are used for. The section for libraries also includes the project setup and what to concentrate on in the setup. Examples of applications of different libraries are given in the subsequent subsection 4.2.2.

#### 4.2.1. Project setup and libraries

This section provides an overview of how the Android project was set up and which libraries were used to provide an application like explained in chapter 3. The first to choose was an IDE. The IDE chosen to develop in was Android Studio. This IDE provides all functionalities needed to create the application wanted. After creating the project, using Gradle (version 2.3) and Java as the programming language, the first thing to think about was, which Android versions the application should be built for. After research what Android operating systems are used primarily, the development team decided to build the application for the target SDK<sup>3</sup> 26. SDK version 26 corresponds to the operating system Android 8.0 Oreo<sup>4</sup>. The minimum SDK version required to use this application is set to 22. SDK version 22 corresponds to Android 5.1 Lollipop<sup>5</sup>.

---

<sup>3</sup>"A software development kit (SDK) is a collection of software development tools in one installable package." *Software development kit* (2019)

<sup>4</sup>*Migrating to Android 8.0* 2019.

<sup>5</sup>*Android 5.1 APIs* 2019.

The application itself contains three folders. The folders are called "manifests", "java" and "res" and are automatically created when creating a new project in Android Studio. Under the folder "java", the application was set up due to the restrictions of MVVM. The following folders were created: "api", "models", "services", "utils" "viewmodels" and "views". The "api"-folder contains all the logic needed for the communication with the server and database of the MOOC-platform. The "models"-folder contains all models and objects used in this application. The "services"-folder contains the logic and classes needed for the notification because these features are solved with a service. Further explanation is given in subsection 4.2.2. The "utils"-folder contains all constants and a class called "HelperUtil", which contains functions that are needed overall in the whole project. The "viewmodels"-folder contains all ViewModels which are needed for the logic of the views. The "views"-folder includes all Fragments<sup>6</sup> and Activities<sup>7</sup> that are needed for setting up the views, which are designed in the "res"-folder. For further explanation and overview of how an Android Studio-project is structured, see *Projects overview* (2019).

After making the basic setup, the libraries used were about to be chosen. One library used mainly is called "Retrofit". Retrofit turns a HTTP API into a Java interface. The declaration needs a method (GET, POST, DELETE, etc.), a URL and a function name. Furthermore, it is possible to define parameters for queries, bodies or dynamic URL adjustment<sup>8</sup>. The retrofit version used is 2.3.0. Detailed explanation of how Retrofit was used in this application is explained in subsection 4.2.2.

Another mainly used library in this application is Realm. The feature used by Realm is Realm Database. Realm Database makes it possible to store, update, delete and read whole objects to and from a database. This makes it possible for the developer not to worry about setting up databases and tables to store information. The developer has the opportunity to extend

---

<sup>6</sup>"A Fragment represents a behaviour or a portion of user interface in a FragmentActivity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities." *Fragments* (2019)

<sup>7</sup>"An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View)." *Activity* (2019)

<sup>8</sup>*Retrofit* 2019.

#### 4. Developing the applications

an Object as an Object of Realm. This makes it possible to store the whole object to a database. So for example, a user gets to the login screen and has to provide the information for an email address and password. If a user fills out the email and password and clicks a button "Log in", the developer can store the information in an Object with the attributes "email" and "password". Afterwards, it is possible to save this object in the Realm Database. Even further, the developer can define primary keys and ignore some attributes of the object, if there exist objects that should not be saved to the Realm Database<sup>9</sup>. A detailed example and how Realm works within this application is given in subsection 4.2.2.

Following the explanation given in section 3.3, the screens within the figures were described as list views containing items. To provide the possibility of using a list view in Android, the library included in this project was RecyclerView-v7. Respecting the MVVM-model described in section 4.1, the library makes it possible to include a RecyclerView in the view. The ViewModel defines which items the RecyclerView is containing. To describe how an item in the RecyclerView looks like, another XML-file is needed which describes the layout. Which XML-file is used for the items in the RecyclerView is described in the RecyclerView. Furthermore, each item in a RecyclerView is held by a separate ViewModel which defines the logic for the item. Further explanation of how this was solved in detail is given in subsection 4.2.2.

The listing 4.1 contains the whole Gradle file for the application module. Due to security and safety, lines 8, 11, 12, 18 and 25 are modified to protect the application and the backend of the MOOC-platform from attacks. This is the basic setup on which the application was built on.

```
1 apply plugin: 'com.android.application'
2 apply plugin: 'realm-android'
3
4 android {
5     compileSdkVersion 26
6     buildToolsVersion "25.0.3"
7     defaultConfig {
8         applicationId "xx.tugraz.xxxxxx"
9         minSdkVersion 22
10        targetSdkVersion 26
11        versionCode x
```

---

<sup>9</sup>Realm Database 2019.

## 4.2. Android

```
12     versionName "x.x"
13     testInstrumentationRunner "android.support.test.runner.
AndroidJUnitRunner"
14 }
15 buildTypes {
16     debug {
17         debuggable true
18         buildConfigField "String", "API_HOST", '"sample-url.at"'
19         buildConfigField "String", "API MOCK_CONFIG", ''''
20         signingConfig signingConfigs.debug
21         applicationIdSuffix ".debug"
22     }
23     release {
24         minifyEnabled true
25         buildConfigField "String", "API_HOST", '"sample-url.at"'
26         buildConfigField "String", "API MOCK_CONFIG", ''''
27         proguardFiles getDefaultProguardFile('proguard-android.txt'), '
proguard-rules.pro'
28     }
29 }
30
31 dataBinding {
32     enabled = true
33 }
34 }
35
36 dependencies {
37     compile fileTree(include: ['*.jar'], dir: 'libs')
38     androidTestCompile('com.android.support.test.espresso:espresso-core
:2.2.2', {
39         exclude group: 'com.android.support', module: 'support-annotations'
40     })
41     androidTestCompile 'com.android.support:support-annotations:26.1.0'
42     androidTestCompile 'com.android.support.test:runner:0.5'
43     androidTestCompile 'com.android.support.test:rules:0.5'
44     compile 'com.android.support:appcompat-v7:26.1.0'
45     compile 'com.android.support:design:26.1.0'
46     compile 'com.squareup.retrofit2:retrofit:2.3.0'
47     compile 'com.squareup.retrofit2:converter-gson:2.3.0'
48     compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
49     compile 'com.android.support.constraint:constraint-layout:1.0.1'
50     compile 'com.android.support:support-v4:26.1.0'
51     compile 'com.android.support:recyclerview-v7:26.1.0'
52     testCompile 'junit:junit:4.12'
53 }
```

Listing 4.1: Gradle for module app

## 4. Developing the applications

### 4.2.2. Implementing the Android version

The application developed in Android is created in Android-Studio<sup>10</sup>. It is based on Gradle and the language chosen is Java. As described in subsection 4.2.1, some external libraries are used which are not automatically provided by the Android SDK. These libraries are explained during this chapter.

#### Login

To use this application a user has to be registered at the MOOC-platform. The login screen is not built natively. The login screen uses a WebView, which is supported and provided by the Android SDK. An activity is responsible for creating the login screen. The corresponding XML-file to the activity contains the WebView. The activity accesses the WebView during onCreate from the XML-file and loads the login web page of the MOOC platform. The application has to know whether a user has logged in successfully. To check whether the application has signed in successfully, the WebView uses a WebViewClient. This WebViewClient contains functions which make it possible to do actions when a certain event arises. In the WebViewClient used in this activity, only two functions are needed. The method onPageStarted when the WebView is about to load a URL and onPageFinished when the WebView web page loaded. When a user types in her/his login credentials, email and password, and hits the login button on the web page, there are two possible outcomes. The first outcome is, that the user has used wrong credentials, which is leading to the fact that the user is not directed to another web page but stays on the already loaded web page. The web page informs the user with an error message, that she/he has used wrong credentials. The second outcome is that the user has provided valid credentials. This case leads to a new web page. This web page contains the access token of the user in its HTML-code. To extract the access token, the WebViewClient used for the WebView is necessary. In the

---

<sup>10</sup>“Android Studio is the official integrated development environment (IDE) for Google’s Android operating system, built on JetBrains’ IntelliJ IDEA software and designed specifically for Android development.” *Android Studio* (2019)



## 4.2. Android

function onPageFinished, the WebViewClient uses JavaScript to extract the access token which is contained in the HTML-code of the new loaded web page. Afterwards, the access token is stored in the SharedPreferences. This access token is needed for the API-requests following in the application. If the application starts a request, it has to add the access token to the request to gain a valid response from the server. The Figure 4.1 shows the screen of the login in Android.

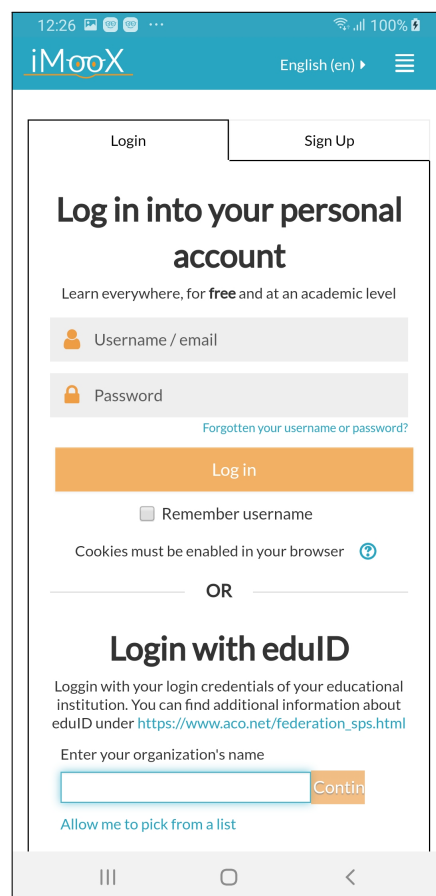


Figure 4.1.: Login screen of Android application

#### 4. Developing the applications

##### **Bottom Navigation**

The Bottom Navigation is an Activity and it is supported by Android SDK and Android Studio. With a right-click on a folder, click on "New", click on "Activity" and click on "Bottom Navigation Activity", Android Studio creates an Activity with the corresponding XML-file. After modification of the XML-file, it includes a ConstraintLayout as the root layout. This ConstraintLayout contains another XML-file and the BottomNavigationView. The other XML-file contains an AppBarLayout and a Toolbar, which is responsible for the top navigation bar. Furthermore, it contains a FrameLayout, which is responsible for loading the Fragments which represent the screens within the Bottom Navigation. The BottomNavigationView uses another XML-file for the menu-attribute. This menu-XML-file includes the menu-tag which contains several item-tags, including an id, an icon and a title. These items represent the clickable icons and texts which are visible in the Bottom Navigation Bar. In this app, the Bottom Navigation Bar contains four different items. One for "Events", one for "Courses", one for "Notifications" and one for "Profile". The texts used for titles are stored in the strings-file. The icons used for the Bottom Navigation Bar were used from the web page *Material Design* (2019). Further explanation of what the items in the Bottom Navigation stand for is given in section 3.3. The Activity used for the Bottom Navigation is the main part of the application. It controls the behaviour of the navigation and sets up basic information needed within the application. The first information stored in the SharedPreferences is the installation date of the application. When the user starts the application for the first time, this information is stored. The installation date is necessary for the application review. After a certain number of days, the application asks the user whether she/he wants to review this application for the master thesis. This feature is excluded from the application after the finalisation of this master thesis. Furthermore, the activity is setting up the Bottom Navigation Bar and sets the title for the top navigation bar. After that, the Activity loads the first Fragment which is visible to the user and corresponds to the first item in the Bottom Navigation Bar. The first Fragment is the one responsible for showing all events to the user. A detailed explanation of this screen is given in ???. After loading the first Fragment, the activity initialises the level of the user if it does not already exist. As explained in section 3.2, the user

is rewarded for completing events and learn goals. The handling of the level is only happening within the application and is not stored in some external database. The level is stored in a local database of the device using Realm. The basic idea of Realm has already been described in subsection 4.2.1 and section 4.1. The Activity calls a function `initailizeRealmLevelIfNotThere`. This function creates an object `Login`. This object `Login` contains the attributes `id`, `level`, `xp` and `xp_next_level`. The Object `Login` is extending the class `RealmObject`, which is necessary to store the whole Object in the Realm Database. The `id`-attribute is declared as the `PrimaryKey` of the database entity. The attribute `level` contains the current level of the user. The attribute `xp` contains the current amount of experience points the user has. The `xp_next_level` contains the number of experience points that are needed to reach the next level. The level of the user is initialised with the following parameters: 1 for `id`, 1 for `level`, 0 for `xp` and 10 for `xp_next_level`. After that, the Object is stored in the Realm database. To read the Object from the database, the application reads the first Object of type `Login` from the Realm Database. Since there is and will only be one Object of type `Login` in the Realm Database, this is enough. The code shown in list 4.2 shows the whole Object `Realm_Level` which is used to store the information for the level of the user in the Realm Database. It also includes the functions used to store and read Objects from the Realm Database.

```

1 public class Realm_Level extends RealmObject {
2
3     @PrimaryKey
4     private int id;
5     private int level;
6     private int xp;
7     private int xp_next_level;
8
9     public Realm_Level() {
10    }
11
12    public Realm_Level(int id, int level, int xp, int xp_next_level) {
13        this.id = id;
14        this.level = level;
15        this.xp = xp;
16        this.xp_next_level = xp_next_level;
17    }
18
19    public int getId() { return id; }
20
21    public void setId(int id) { this.id = id; }
22
23    public int getLevel() { return level; }
24
25    public void setLevel(int level) { this.level = level; }

```

## 4. Developing the applications

```
26 public int getXp() { return xp; }
27
28 public void setXp(int xp) { this.xp = xp; }
29
30 public int getXp_next_level() { return xp_next_level; }
31
32 public void setXp_next_level(int xp_next_level) { this.xp_next_level =
33 xp_next_level; }
34
35 public static void initializeRealmLevelIfNotThere()
36 {
37     Realm realm = Realm.getDefaultInstance();
38     Realm_Level level = realm.where(Realm_Level.class).findFirst();
39     if(level == null)
40     {
41         Realm_Level realm_level = new Realm_Level(1, 1, 0, Constants.
42 MAX_XP);
43         realm.beginTransaction();
44         realm.copyToRealm(realm_level);
45         realm.commitTransaction();
46     }
47
48 public static void copyOrUpdateLevelToRealm(Realm_Level level) {
49     if (level != null) {
50         Realm realm = Realm.getDefaultInstance();
51         realm.beginTransaction();
52         realm.copyToRealmOrUpdate(level);
53         realm.commitTransaction();
54     }
55 }
56
57 public static Realm_Level getRealmLevel()
58 {
59     Realm realm = Realm.getDefaultInstance();
60     Realm_Level results = realm.where(Realm_Level.class).findFirst();
61     if(results != null)
62         return realm.copyFromRealm(results);
63
64     return null;
65 }
66 }
```

Listing 4.2: Object Realm\_Level in Android

After initialising the Level for the user, the Activity starts an AlarmManager with a daily interval, starting at 10am, and a Service. This Service called NotificationBackgroundService is used for the notification-feature within the application. The functionality of the notification-feature is explained in section 4.2.2. In the function onResume in the Activity, the installation date of the application is checked. As described earlier, the installation date is saved at the first time the user starts the application and is stored in

SharedPreferences. The reason why the Activity checks the installation date in the `onResume`-function is, that this function will be called every time the user "opens" the Activity. So every time the user starts the app, the installation date is checked. If the installation date is more than some days ago, an `AlertDialog` is shown to the user, which asks the user for a feedback of the application for the master thesis. Furthermore, the Activity for the Bottom Navigation contains functions which handle the navigation of the Fragments. The functions are needed to tell the application what to do when a certain item in the Bottom Navigation Bar is clicked. The Activity sets the title of the Top Navigation Bar according to the item clicked in the Bottom Navigation Bar and it also checks for new and old notifications. If there are new or old notifications, the Activity sets the badging of the notification-item in the Bottom Navigation Bar, like described in section 3.3. Further explanation is given in section 4.2.2. The Activity also contains the logic for what happens if a user wants to create a new event and what happens if a user clicks on an item in the Profile-section. Detailed explanation is given in section 4.2.2 and section 4.2.2.

### List events

The first Fragment that the `BottomNavigationActivity` loads is the `Event-Fragment`, which is responsible for displaying all the events of all subscribed courses to the user. As described in section 3.3, the application uses a list view that shows items. Furthermore, at the top of the screen, there is a "Plus"-sign which leads to the screen for creating an event if clicked. If a user clicks on an item in the list view, she/he is redirected to the detailed view of the Event, which is described in section 4.2.2. The Fragment makes use of data binding. This means, the data of the `EventsViewModel`, which is responsible for the data and logic of the View, must not be assigned explicitly to the list view of the View. This means, if the data changes in the `EventsViewModel`, the list view in the View automatically updates according to the new data. The Fragment makes use of Androids `RecyclerView`. The View of the Fragment contains the `RecyclerView`. Furthermore, it contains three texts at the top which are aligned horizontally. The texts say "Outstanding", "Completed" and "All". This changed since the process of chapter 3 and section 3.3. During the prototype process these three texts

#### 4. Developing the applications

were "Future", "Past" and "All". Due to the fact that there would be units for "Future" and "Past", the events would split up, no matter whether they were finished or not. That is why this filtering of events changed to "Outstanding", "Completed" and "All". By default, the "Outstanding"-unit is chosen. If this one is chosen, the application shows all events to the user which are not finished, no matter whether they are in the past, present or future. The "Completed"-raster shows all events that are already finished, no matter whether they are in the past, present or future. The "All"-raster shows all Events, no matter whether they are in the past, present, future, completed or not completed. Furthermore, the View of the Fragment contains a ProgressBar which indicates when the application is loading data and a TextView which shows an error message to the user if something went wrong during the loading process of data. To make the items look like in the section 3.3, another XML-file is needed. This XML-file defines the look-a-like of the items within the RecyclerView. There are some things that changed during the development process compared to the designs in section 3.3. As Figure 4.2 shows, there exist progress bars within the items. These indicate the progress of the user within this Event. So if a user finished one task out of two in this Event, she/he has a progress of 50 %. Furthermore, the date visualisation is extended by the year the event is happening. In section 3.3 this was limited to only the day and the month of the Event. The items also contain a text which displays the title of the course the event is assigned to, so the user automatically sees, which course the event belongs to. The EventFragment is responsible for setting up the data binding between the RecyclerView and the EventsViewModel using a RecyclerViewAdapter. In the function onResume of the Fragment, the Fragment uses the ViewModel to load the data from the API. The rest of the logic is handled by the ViewModel. If the data is loaded successfully, the ViewModel adds the received data from the API and the events and learn goals created by the user to three different ArrayLists and sorts them by the due date. The events and learn goals created by the user are received from the Realm Database. The three ArrayLists contain events for all three different types "Outstanding", "Completed" and "All". So one ArrayList contains all events which are "Outstanding" and not completed by the user. One ArrayList contains all events which are "Completed". The last ArrayList contains all Events, no matter whether completed or not. Furthermore, there is an ObservableArrayList which is connected to the RecyclerView of the Fragment. So, if this

ObservableArrayList is changed, the View is adapted according to the data. If the screen is loaded, the default type chosen is "Outstanding". So, the ViewModel assigns the data of the ArrayList containing all uncompleted events to the ObservableArrayList which is connected to the RecyclerView. If a user clicks on the different units "Outstanding", "Completed" and "All", the ViewModel changes the data of the ObservableArrayList according to the type chosen. When changed, the RecyclerView is adapted. The items of the RecyclerView itself are also connected to another ViewModel, which defines what data to display in an item and what to hide. For example, if no due date is set for an Event, the ViewModel responsible for the item hides the part for the date within the item. Furthermore, the ViewModel of the RecyclerView contains the logic, that the RecyclerView is automatically scrolling to the event which is the nearest to the current date. This feature is useful, since the user does not have to scroll down until she/he finds the event which is the next to come. The application does it for the user. If no data is provided, this can happen if the request to the server fails or there are no events due to the fact that the user is not assigned to any Courses, the RecyclerView is hidden and an error message is displayed via a TextView. The screen of the Fragment and how it finally looks like can be seen in Figure 4.2. On the screen itself, there is a "+"-sign which leads to the screen where a user can create an Event, a learn goal or a note. This is described in section 4.2.2. If a user clicks on an item within the RecyclerView, the user is redirected to a detailed view of the event or learn goal. This screen will be described in the following section 4.2.2.

### **Detail View - Event**

The detail view screen of an event exists to provide additional information of the event and to provide further action for the user to interact with an Event. One action is to set a "reminder-date". Another action depends on the origin of the event or learn goal. If an event is received from the MOOC-platform itself, the user has the possibility to click a button at the bottom of the screen which leads the user to the event at the MOOC-platform in the browser of the device. If the event or learn goal is created by the user within the application and is therefore not originally received by the MOOC-platform, the user has the possibility to delete the event or learn goal with a

#### 4. Developing the applications

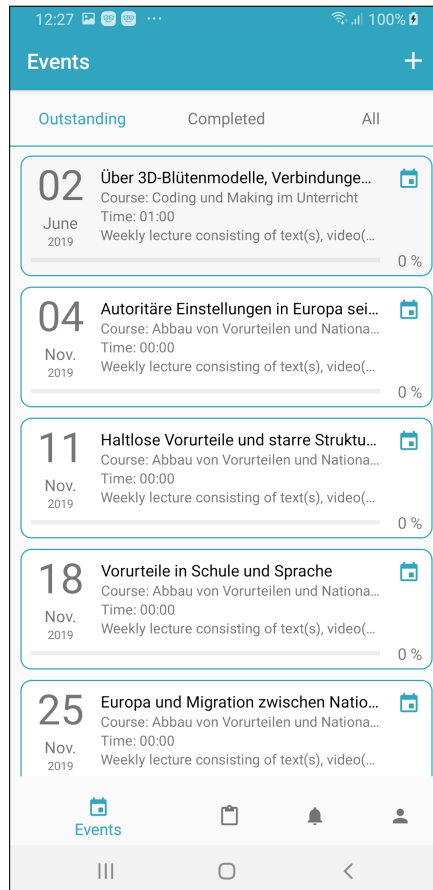


Figure 4.2.: List of events and learn goals in Android application



button at the bottom of the screen. The screen itself is build on an Activity, a Fragment and a ViewModel. The Activity is needed, because there is no Bottom Navigation needed at this screen. Therefore a new Activity is started, because the navigation of the application changes within this screen. The Activity calls a Fragment which contains part of the logic and holds the View. The Fragment sends a request to the Realm Database to receive the requested event providing a primary key. Furthermore, the Fragment contains the logic for a DatePickerDialog. This DatePickerDialog provides the possibility that the user can pick a date from a calendar, which is common on Android devices. The DatePickerDialog is opened if a user clicks on the icon for the "reminder-date" or the TextView for the "reminder-date". The user does not have the possibility to write a date directly to the according TextView. This prevents the user from providing an invalid date. The rest of the logic is handled by the ViewModel. The Fragment forwards the event received from the Realm Database to the ViewModel. The ViewModel then provides the necessary information for the View. The View itself is build like the items in the RecyclerView explained in section 4.2.2. If a user wants to change the "reminder-date", she/he clicks on the according section on the screen and provides a date. After that, the user can save the event and therefore save the "reminder-date", so she/he gets notified about this event on this date. If the saving-process is successful, the user automatically gets redirected to the previous screen. If the saving-process failed, the user gets informed about the error. The saving-process is handled by the ViewModel. If the user clicks on the button at the bottom of the screen which was described before in this section, two different actions can happen. The first action is, that the user will be redirected to the link of the event on the MOOC-platform. The second action is, that the user is able to delete the Event. What action happens depends on the origin of the Event. This data is provided by the Fragment which loads the event from the Realm Database and therefore knows the information whether it is created by the user or is received from the backend of the MOOC-platform. Also, in comparison to the items in the RecyclerView described in section 4.2.2, this detailed view contains more information. The description is fully written out on this screen. So if a user sees a part of the description of the event in the RecyclerView and wants to read the full description, she/he can click on the event and will see the detail view of the Event, which contains the full description. Figure 4.3 shows the screen described in this section. At the top right corner, the described

#### 4. Developing the applications

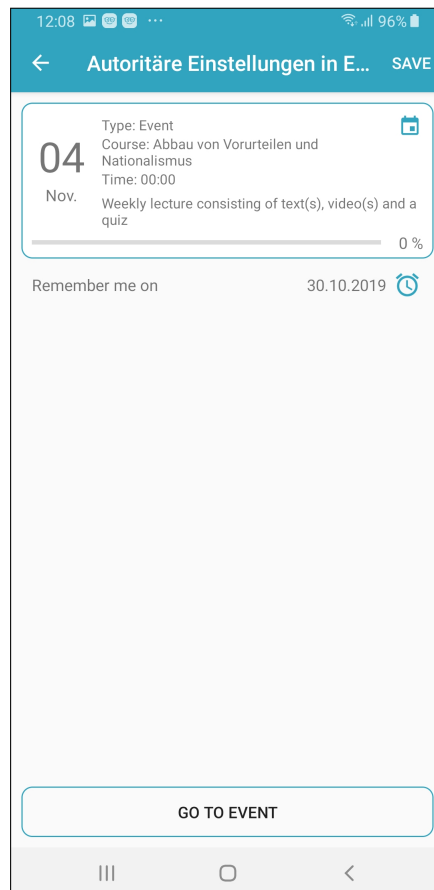


Figure 4.3.: Detail view event - Android application

"Save"-action is shown. Below the section which is describing the event itself is the section for setting the "reminder-date". In the figure it is called "Remember me on". At the bottom of the screen the button is located. It changes depending on the origin of the Event. It either contains the text "GO TO EVENT" or "DELETE". The "DELETE" action is only provided if the event or learn goal is created by the user. How a user can create an Event, a learn goal or a note is described in the following section section 4.2.2 Create Event.

### Create Event

A user has two possibilities to reach the screen for creating an event within the application. As described in section 3.3, one way to reach the screen is by clicking the "+"-sign on the main screen which is responsible for showing all events and learn goals. Another way is to click the "+"-sign which is shown in the screen which is responsible for showing all events, learn goals and notes related to a course, which the user can choose on the screen responsible for showing all courses. The screen for showing all courses is described in section 4.2.2. The screen responsible for showing all events and learn goals for a course is described in section 4.2.2. The creation screen makes it possible to create an event, a learn goal or a note. Further detail about what the creation screen is doing exactly and how it is supposed to work is given in section 3.3. If a user clicks one of the "+"-signs, the application is starting a new activity. The reason for that is the same like with in section 4.2.2. At this screen there is no need for a Bottom Navigation, since there should happen no navigation except from navigating back to the previous screen. When the Activity is started, it runs the according Fragment for showing the creation screen with the SupportFragmentManager. Furthermore, this Activity sets up the logic for two buttons which are lying in the top navigation bar. One button is for navigating to the previous screen and the other button is for saving a new event, learn goal or note with the information given by the user. This button is indicated by the text "SAVE". The logic for saving an event, learn goal or note is provided in the Fragment started by the Activity. The Fragment itself inflates the according View with data binding. Therefore it creates a ViewModel and defines it as the ViewModel responsible for the logic

#### 4. Developing the applications

for the inflated View. The next part is to receive some information from the Activity. The Activity can hold information about the creation process, depending on what the previous screen was. If a user starts the creation process from the main screen, which is responsible for showing all events and learn goals, no information is provided. If a user starts the creation process from the detailed view of a course, the Activity holds information about the course identifier, the course title and the type (event, learn goal or note). Depending on that information, the Fragment sets up the screen. If a course identifier and a course title is available, the Fragment automatically fills out the part for the course the user has provided. Every event, learn goal and note has to be assigned to a course. Furthermore, the Fragment sets the type automatically if information is provided. So, if the Activity holds the information that the user wants to create a learn goal, the Fragment automatically sets the type to learn goal. This becomes clear in Figure 4.4. Besides setting up the View by information given by the Activity, the Fragment also sets up the logic for DatePickerDialogs, a TimePickerDialog, the radio buttons responsible for the type, the switch responsible for the "reminder-date" and an AlertDialog which is used for choosing a course. One DatePickerDialog triggers if a user clicks on the section for the due date of the event. If a user has chosen a date from the DatePickerDialog, the information is stored to the ViewModel. The ViewModel then automatically sends the information to the View and the date is shown formatted to the user. The other DatePickerDialog is triggered if a user clicks on the section for "reminder-date". To reach this section, the user has to turn on the Switch which is responsible for showing the "reminder-date". The behaviour of this DatePickerDialog is the same like with the one described before. The TimePickerDialog is opened if a user clicks on the TextView for the time of the event or learn goal. If a user has chosen a time, the information is send to the ViewModel which redirects the information to the View, so the formatted time is shown to the user. The radio buttons shown in Figure 4.4 are custom-build. The screen changes depending on which type the user has chosen. If no information about the type is given by the Activity, the default value when starting the screen is "Event". If "Event" is chosen, the screen is like shown in Figure 4.4. If the user chooses "Learn Goal", the screen is changing. The Fragment changes the title in the top navigation bar from "Creating Event" to "Creating Learn Goal" to show the user she/he is about to create a learn goal. If a user chooses to create a note, the screen changes

even more. As described in section 3.3, there exists information that is not needed for a note. One example for that is the due date which is needed for events and learn goals. Therefore, the ViewModel hides all sections except the sections for title, description and course. The screen always shows the user which information is needed to create an event, learn goal or note. Which information is required depends on the type the user has chosen. If a user has finally provided all necessary information, clicks the "SAVE"-button in the top navigation bar and the saving process is successful, the user receives a feedback from the application that she/he has created an event, a learn goal or a note successfully and is redirected to the previous screen. If the saving process was not successful, the user is informed about the error. If a user has not provided all necessary information to create an event, a learn goal or a note but clicks on the "SAVE"-button, the ViewModel informs the user about what information is missing.

### Courses

The screen for listing the courses is part of the main navigation. It shows all courses to the user that she/he has subscribed to. The screen itself uses a Fragment. This Fragment contains the logic for the data binding of the View and the ViewModel. The View makes use of a RecyclerView to display all courses. Therefore, the screen needs a RecyclerViewAdapter and an ItemViewModel which tells the RecyclerView how the items should be displayed. The ViewModel contains the logic for requesting the courses at the backend of the MOOC platform. When receiving the response, the ViewModel sorts the courses by their title and displays it in the RecyclerView with an ObservableArrayList. Furthermore, the ViewModel loads all courses from the Realm Database if there exist any. If so, the ViewModel compares the identifiers of the courses received by the backend and the courses received by the Realm Database. This way the ViewModel can tell if the user has assigned any colour to the course. If the identifiers match and the course from the Realm Database contains information about the colour, the ViewModel assigns it to the corresponding Object in the ObservableArrayList. Details about how colouring works is defined in section section 4.2.2. whether the user is not assigned to any courses, the View shows an error message to the user which is containing a corresponding text. The logic

#### 4. Developing the applications

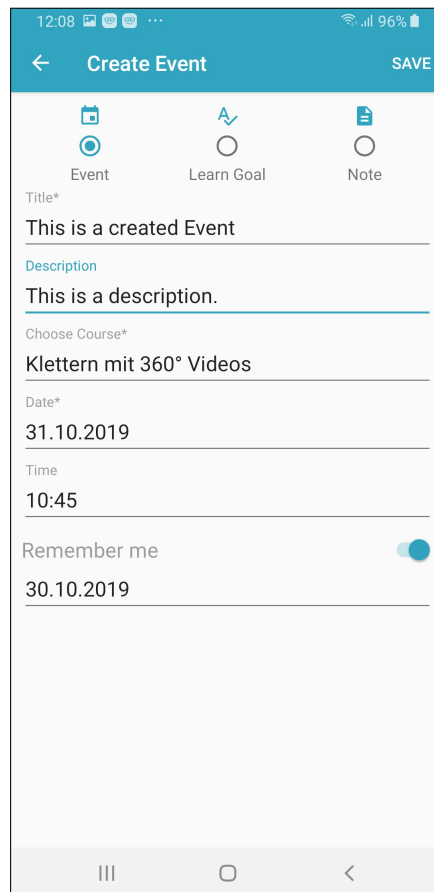


Figure 4.4.: Creating event, learn goal and note - Android application

## 4.2. Android

whether the error message is visible or not is handled via the data binding between the View and the ViewModel. The logic of what happens when a user clicks on an item in the RecyclerView is defined in the Fragment of the screen. If a user clicks on a item, she/he is redirected to a screen which is responsible for displaying a detailed view of the course. This screen is described in section section 4.2.2.

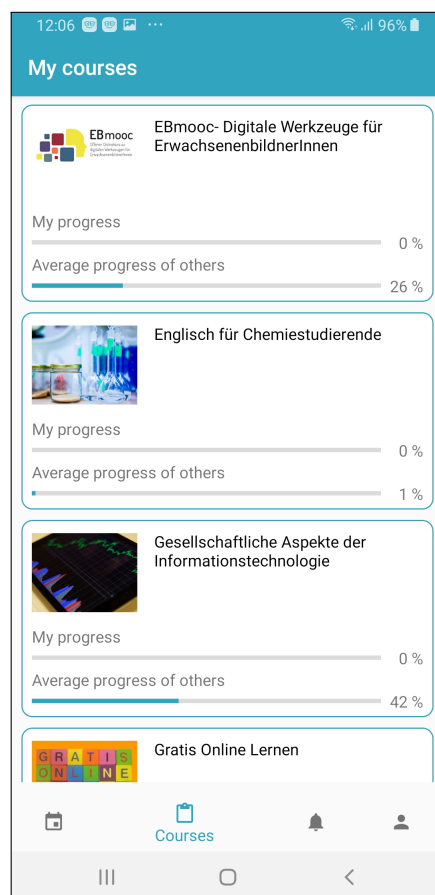


Figure 4.5.: List of subscribed courses in Android application

## 4. Developing the applications

### **Detail View - Course**

The detail view of the courses screen is similar to the screen which is responsible for displaying all the events to the user. The difference is, that it only shows the events, learn goals and notes for the corresponding course the user has chosen. If a user clicks on a Course, described in section section 4.2.2, she/he gets redirected to the detail view screen. The difference to the other screens is, in this case the detail view is held by a different Activity. This is due to the fact, that this screen has it's own Bottom Navigation. The Bottom Navigation holds three different items. The Activity is a container for loading a Fragment which takes care of displaying the events, courses and notes to the user. If a user clicks on one of the three tabs in the Bottom Navigation, the Fragment loads different items into the RecyclerView. The Fragment of this screen makes use of a RecyclerView. The corresponding View to the Fragment contains it. The ViewModel attached to the View holds one ObservableArrayList which defines the items that are displayed in the RecyclerView. The ViewModel changes this ObservableArrayList depending on what the user has chosen. In fact there are seven different possibilities the user can choose. One tab of the Bottom Navigation is responsible for telling the ViewModel to use the events as the items to display. If chosen, the user has further possibilities, like with the screen described in section section 4.2.2, to choose between "Done", "Outstanding" and "All" events. If a user chooses "Done" while the tab for the events is chosen, the ViewModel only loads the events into the RecyclerView which are related to the chosen course and are already finished by the user. Further description is already given in section section 4.2.2. The same applies to learn goals. In sum these are six different possibilities of what the screen can load. The seventh possibility is described by the tab for notes. As already described in section 3.3, notes cannot have a due date and they cannot be set as "Done". Notes only contain information which can be created by the user for a course. Therefore, it makes no sense to give the user the possibility to choose between "Done", "Outstanding" and "All" notes. The top tab bar, shown in Figure 4.6, is not necessary if the user has chosen to display all notes. When the Fragment is loaded, the ViewModel loads all the events, learn goals and notes which are related to the chosen course from the Realm Database. After that, the ViewModel sorts



them by their due date, except the notes, and stores them in seven different ArrayLists, depending on what type they are. For example, one item is a learn goal and it is already done. Therefore the ViewModel saves it in the ArrayList which is responsible for holding all learn goals that are already finished by the user. If a user then chooses the tab learn goals in the Bottom Navigation and chooses "Done" in the Top Bar, this ArrayList is loaded into the ObservableArrayList which loads the items into the RecyclerView. If a user clicks on one of the items, no matter whether event, learn goal or note, she/he gets redirected to the screen which is responsible for displaying the detail view of an event, described in section section 4.2.2. The Bottom Navigation used is from the standard library provided by AndroidStudio. The top bar is custom-build. If a user has no events, learn goals or notes, a corresponding message is displayed to the user about the situation. An example of how the whole screen can look like is given in Figure 4.6.

### **Notification**

This section includes explanations about two different features. One feature is about the screen of displaying notifications to the user. The other feature is about how the user gets notified about certain events. The user can receive three different types of notifications on her/his device. As described in section section 4.2.2 and in section section 4.2.2, the user can set a due date to events and learn goals. If the due date of an event or learn goal is reached, the user receives a corresponding notification and gets remembered that there is a due date set. The third type is that a user gets informed about new available courses on the MOOC platform. If there is a new course starting soon at the MOOC platform, the user gets informed about it. To make the notification work, the application uses a BroadcastReceiver and a Service. The BroadcastReceiver tells the application when to start the mentioned Service. In the case of the Android application, this happens every day at a specific time. The Service is responsible for sending the notification to the user. Every day at a specific time this Service gets invoked. It then checks whether the user has set any due dates for the current date. If so, the Service sends out notifications for all events and learn goals which due date is the current date. Also, the Service starts a request to the backend of the MOOC platform to check whether there are new available courses.

#### 4. Developing the applications

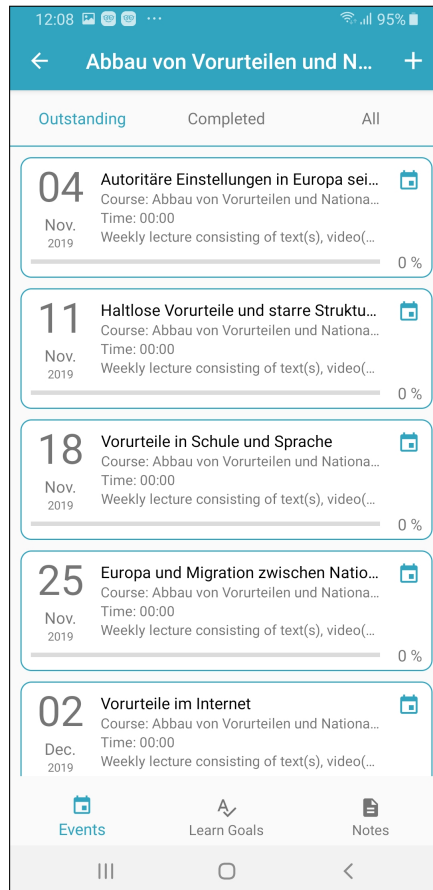


Figure 4.6.: List of events of specific subscribed course - Android application

If there are any notifications send to the user, the Service stores a custom Object to the Realm Database. This Object contains information about which event, learn goal or new available course the user has been informed. This is needed since the user should not get informed twice about the same event, learn goal or new available course. Storing the notifications also makes it possible for the user to check her/his notifications within the application.

The user can reach the screen, which is responsible for displaying the notifications to the user via the main Bottom Navigation. This screen works like most of the other screens. It makes use of a Fragment, a ViewModel, an ItemViewModel and a RecyclerView. The ViewModel is connected to the View, which is holding the RecyclerView, via data binding. It tells the RecyclerView to show which items. The ItemViewModel is connected to the View which tells the RecyclerView how the items should look like. In this case, the item contains an ImageView and a TextView. The ImageView displays the icon depending on the type of the notification. As explained before in this section, there exist three different types of notifications. When the Fragment is started by the main Activity, the Fragment tells the ViewModel to load all notifications stored in the Realm Database. The ViewModel receives the notifications by the Realm Database and sorts them by the date they have been sent to the user and loads them into the RecyclerView. This way, the user can always check her/his newest notifications. If a user clicks on one of the notification items, she/he is either redirected to the detail view of the event or learn goal or is redirected to the new available course in the browser of the device. This should help the user not to forget about events or learn goals which are about to end. Furthermore, this feature is supported by a type of badge. When the user opens the application, the main Activity checks whether there have been any new notifications since the user has viewed the notification screen the last time. If so, the Main Activity enables the badging for the notification icon in the main Bottom Navigation. This means, the tab item of notifications in the Bottom Navigation is coloured in red to signal that there a new notifications. If the user opens the notification screen, the Fragment responsible for displaying the notifications removes the badging from the Bottom Navigation tab item to signal that the user has viewed all notifications. An example of how the screen looks like is given in Figure 4.7.

#### 4. Developing the applications

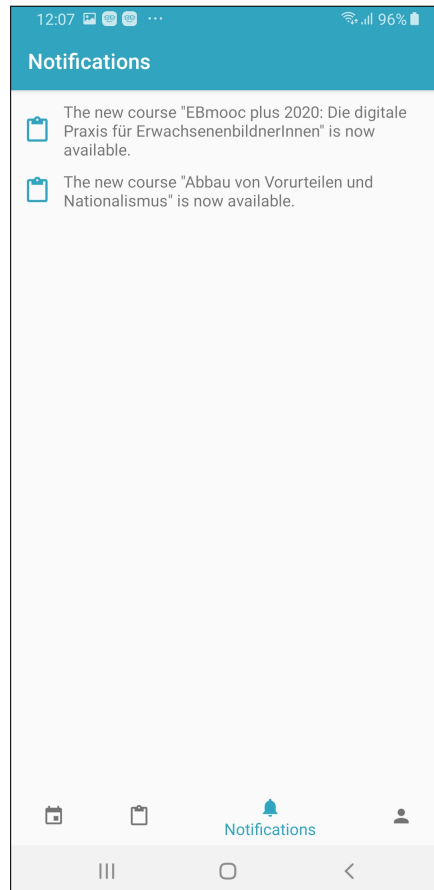


Figure 4.7.: List of received notifications in Android application

## Profile

The profile is another main section of the main Bottom Navigation. It is the last item from left to right. The screen makes use of a Fragment, a View and a ViewModel. On this screen, there is no need for a RecyclerView. At the top of the screen is the part for the gamification described in section 3.2. The part contains an ImageView, a ProgressBar and two TextViews. The ImageView contains the image corresponding to the Level of the user. The two TextViews also adapt to the level of the user. The ProgressBar tells the user how many experience points she/he has earned and how many she/he needs to reach the next level. As described in section 3.2, a user can earn experience points by finishing events and learn goals. The needed information about the level is loaded by the ViewModel. When the Fragment is started, it tells the ViewModel to request the needed information at the backend of the MOOC platform. When finished, the ViewModel loads the information for the described Views. The View and the ViewModel are connected with data binding. Beneath the part for the level, the section contains further information and links. Each part is defined by a TextView and an ImageView showing an arrow. The following explanation describes the different parts from top to bottom. The first item is "All courses". If a user clicks on this item, the browser of the device opens a URL that leads the user to all courses that are available at the MOOC platform. The next item "iMoox.at" also leads to the browser opening the URL of the MOOC platform. The following item is named "Colors". This specific feature is described in the section section 4.2.2. This item leads to the screen which is responsible for colouring the courses and events, described in section 3.3. The next item shows "Play Tutorial". If a user clicks on this item, she/he is starting the tutorial for the application. The tutorial feature consists of four Activities. Each Activity holds information and details about how to use the application and the features. In the tutorial, the user can navigate back and forth between the different screens. The tutorial is also started when the user opens the application for the first time. The next item is called "Master Thesis Feedback". Since the applications for Android and iOS were developed as part of this master thesis, the users can give feedback. The feedback of the application is discussed in chapter 5. After the part for the feedback, there follow items for the imprint and data privacy. If a user clicks

#### 4. Developing the applications

on one of these, she/he is opening an URL in the browser of the device leading to the imprint or data privacy of the application and the MOOC platform. The last item in the list is responsible for signing out the user. If the user clicks on this item, the application deletes the token used for the authorisation in the backend of the MOOC platform and gets redirected to the login screen. The profile screen is shown in Figure 4.8.

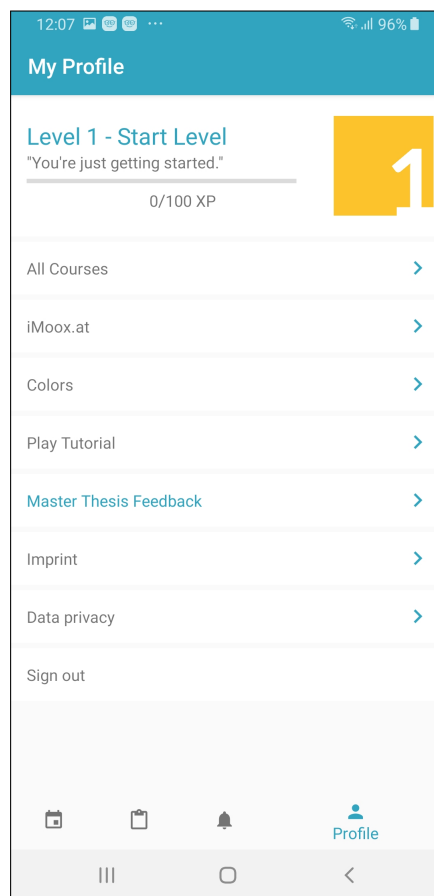


Figure 4.8.: Profile including gamification - Android application

## Coloring

As described in section 3.3, the feature of colouring is used to colour the events, learn goals, notes and courses within the application. It should help the user to differentiate between the courses and which event, learn goal or note belongs to which course. The user can choose a colour on her/his own. This should help the user remember which colour is assigned to which course. The feature colouring makes use of two Activities and one Fragment. The first Activity starts the Fragment. The Fragment holds the ViewModel and the View. The View and the ViewModel are connected with data binding. The View contains a RecyclerView. When the Fragment is started, the ViewModel loads all courses the user is assigned to from the Realm Database. After loading the courses from the database, the ViewModel sorts the courses by their title and stores them in an ObservableArrayList which is connected to the RecyclerView. If a user clicks on one item of the RecyclerView and therefore chooses a course, she/he is redirected to the next Activity. This Activity is responsible for displaying seven colours to the user. The colours are displayed in a GridLayout with two columns. Each section of the GridLayout contains a RelativeLayout, which background is the colour displayed to the user. This RelativeLayout contains an ImageView. The ImageView shows a tick. By default, the visibility of all ImageViews is set to invisible. When a user clicks on one item of the GridLayout, the ImageView of the section she/he clicked on becomes visible and all other ImageViews are set to invisible. Therefore the user always sees which colour is currently active. In the toolbar at the top of the screen, there is a button that is saying "SAVE". If one colour is active and the user clicks this button, the Activity stores the colour code to the course Object in the Realm Database and the user is redirected to the previous screen, which shows all courses to the user. The user then can choose another course and pick a colour for this course and so on. Currently, the number of colours is limited to seven. The colours are defined by designers of the MOOC platform and match certain requirements in recognisability and discriminability. The screen which is responsible for displaying all colours to the user is shown in Figure 4.9.

#### 4. Developing the applications



Figure 4.9.: Colouring - screen of Android application



## 4.3. iOS

This section describes the development of the iOS version. It provides a detailed explanation of how each screen, the navigation, data storage and API communication was implemented. The first to follow is a list of which libraries are used in the application and an explanation of what they are used for. The section for libraries also includes the project setup and what to concentrate on in the setup. Examples of applications of different libraries are given in the subsequent subsection 4.3.1. The following sections refers to an "app" or an "application". It means only the application for the iOS version.

### 4.3.1. Project setup and libraries

This section provides an overview of how the iOS project was set up and which libraries were used to provide an application like explained in chapter 3. The IDE chosen was Xcode. This IDE provides all functionalities needed to create the application wanted. The application is programmed in Swift. After research what iOS operating systems and versions are used primarily, the development team decided to build the application for the deployment target of version 10.0<sup>11</sup>. Furthermore, the application allows a device orientation in portrait mode and both landscape modes. The structure of the iOS project is the same as the structure of the Android version. After creating a new project with Xcode, the folders "Api", "Models", "Views", "ViewModels", "Util" and "Resources" were created. The "Api"-Folder contains all the logic needed for the communication with the server and database of the MOOC platform. The "Models"-folder contains all models and objects used in this application. The "Views" folder contains all Cells needed for TableViews and ViewControllers. The "ViewModels" folder contains all ViewModels which control the logic of the application. The folder "Util" contains all classes and structs that contain generic functions and constants used across the whole application. The "Resources" folder contains the Assets file, which holds all used icons and the Localizable-Strings files which are used for multilingual support.

---

<sup>11</sup>Share of Apple devices by iOS version worldwide from 2016 to 2018 2018.

## 4. Developing the applications

After making the basic setup, the libraries used were about to be chosen. One library used mainly is called Alamofire. Alamofire is an HTTP networking library. It makes it possible to start a request with a given URL and define the type of Object that is returned. The Object is automatically mapped from the JSON contained in the response of the backend.

Another mainly used library is Realm. Due to the different programming languages, the way of storing and retrieving data from the Realm Database in iOS is different from Android, but the core functionality is the same. For further detail about what Realm is about, check out subsection 4.2.1. An example of how to retrieve and store data in Realm in iOS is given in subsection 4.3.2.

### 4.3.2. Implementing the iOS version

The application developed in iOS is created in Xcode. The programming language chosen is Swift. As described in subsection 4.3.1, some external libraries are used which are not automatically provided by Xcode. These libraries are explained during this chapter.

#### Login

To use this application a user has to be registered at the MOOC platform. The login screen is not built natively. The login screen uses a WKWebView. For creating the Login screen, a ViewController is needed. This ViewController initialises a WKWebView in its loadView method. Furthermore, the ViewController extends the needed protocols WKUIDelegate and WKNavigationDelegate. In the loadView method of the ViewController, these protocols get assigned to the WKWebView. After the View of the ViewController was built, a loading symbol is created to show the user that the page is loading. After that, the WKWebView starts a request and loads the login URL of the MOOC-platform. This happens in the viewDidLoad method of the ViewController. After the page was loaded, the application has to know whether a user has logged in successfully. To check whether the user has signed in successfully, functions from the WKNavigationDelegate

protocol are needed. After the user has provided the credentials, e-mail and password, on the login page and clicks on the login button, there are two possible ways to follow. The first one is that the login process fails. The reason for that could be that the credentials were wrong. If such happens, the user gets noticed by the login page itself. The second way is that the credentials were valid. If so, the user gets redirected to a page where the authorisation token is located inside the HTML of the new page. The authorisation token is necessary to request data from the MOOC platform backend later on in the application. So in the delegate function of WKNavigationDelegate which gets triggered if a page was loaded successfully, the ViewController checks whether the loaded page is the page which contains the authorisation token. This happens with a URL check. If the URL is the right one, the ViewController parses the HTML-code of the page to retrieve the authorisation token. If the authorisation token was retrieved correctly, the ViewController saves the authorisation token for further use and loads a new ViewController. The new ViewController is responsible for displaying the start screen to the user. This is the first time the authorisation token is needed. The ViewController requests the level of the user. The level is described in section 3.2. The backend responds with an image which is displayed to the user. After a certain time, the main Navigation Controller gets loaded which is described in the following section 4.3.2.

#### **Bottom Navigation**

The Bottom Navigation in the iOS version of the application is achieved by using a UINavigationController. This UINavigationController is responsible for showing the top bar of the screen which is including a back button, icons and the title of the screen that is visible. The UINavigationController also includes a UITabBarController. This UITabBarController is responsible for showing the four different items at the bottom of the screen. The four different items are "Events", "Courses", "Notifications" and "Profile". The icons used for the Bottom Navigation were retrieved from the web page *Material Design* (2019). The items used in the Bottom Navigation Bar are described in detail in this chapter. The UITabBarController includes functions that help to show different items on different screens when navigating through the application. One example would be a function which is responsible for

#### 4. Developing the applications

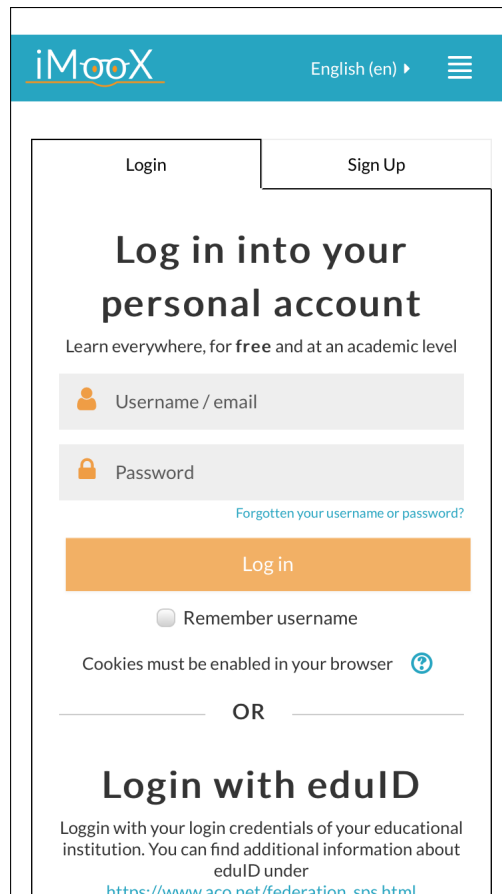


Figure 4.10.: Login screen of iOS application

### 4.3. iOS

showing the "Plus-Sign" which is used for creating events. Furthermore, the `UITabBarController` is responsible for initialising the level. Why and how the level is used is described in section 3.2 and section 4.2.2. The `UITabBarController` calls a function `initializeRealmLevelIfNotThere` which is a static function in the `Object RealmLevel`. The `Object` is built the same as in Android. The function checks whether there is already a level stored in the Realm Database. If there is no `Object` stored in the Realm Database, this means the level has not been initialised so far and gets created and stored by the `Object`. The code shown in listing 4.3 shows the whole `Object RealmLevel` which is used to store the information for the level of the user in the Realm Database in Swift. It also includes the functions used to store and read `Objects` from the Realm Database.

```
1 class Realm_Level: Object {
2
3     dynamic var id: Int = 0
4     dynamic var level: Int = 0
5     dynamic var xp: Int = 0
6     dynamic var xp_next_level: Int = 0
7
8     override static func primaryKey() -> String? {
9         return "id"
10    }
11
12
13    convenience init(id: Int, level: Int, xp: Int, xp_next_level: Int) {
14        self.init()
15        self.id = id
16        self.level = level
17        self.xp = xp
18        self.xp_next_level = xp_next_level
19    }
20
21    static func initializeRealmLevelIfNotThere() {
22        let database = try! Realm()
23        let level = database.object(ofType: Realm_Level.self, forPrimaryKey:
24    1)
25        if level == nil {
26            try! database.write {
27                let rl = Realm_Level(id: 1, level: 1, xp: 0, xp_next_level:
28    Constants.MAX_XP)
29                database.add(rl, update: true)
30            }
31        }
32    }
33
34    static func copyOrUpdateLevelToRealm(level: Realm_Level?) {
35        if let l = level {
36            let database = try! Realm()
37            try! database.write {
38                database.add(l, update: true)
39            }
40        }
41    }
42 }
```

#### 4. Developing the applications

```
38     }
39
40     }
41
42     static func readLevelFromRealm() -> Realm_Level? {
43         let database = try! Realm()
44         return database.object(ofType: Realm_Level.self, forPrimaryKey: 1)
45     }
46
47 }
```

Listing 4.3: Object Realm.Level in iOS

The UITabBarController also controls the badging of the navigation tab item in the Bottom Navigation Bar. It reads from the Realm Database and checks whether the user has received new notifications. If so, the UITabBarController is badging the navigation item in the Bottom Navigation Bar. Further description about the notification feature is given in section 4.3.2. After the UITabBarController was loaded, this Controller loads the first UIViewController which is appearing in the Bottom Navigation Bar. This UIViewController is responsible for showing the events to the user. This UIViewController is used to make the basic setup for the application. As with the Android Version, this UIViewController stores the date of the first use of the application. The date is stored in UserDefaults. Each time this UIViewController gets invoked, it loads and checks the installation date of the application from the UserDefaults and checks whether it has been ten days since the application was first used. If so, the UIViewController creates an UIAlertController which asks the user to give feedback to this application for the master thesis. This feature is removed after the finalisation of this master thesis. Also, if the application is used the first time, this UIViewController loads the tutorial, which is controlled by another UIViewController. Further description of this UIViewController is given in section 4.3.2. In general, if a user clicks on an item in the Bottom Navigation Bar, a new UIViewController is started corresponding to the item clicked. Compared to the Android version, where the functionality of the Bottom Navigation is controlled in the code in the Activity, the logic of what happens in the UITabBarController is controlled in the Storyboards of the application.

## List Events

The first UIViewController loaded by the UITabBarController is the one that is responsible for displaying the events to the user. ?? shows the screen. On the top of the screen is the top navigation bar. It shows the title of the screen and a "Plus"-Sign. If a user clicks on this icon, she/he gets redirected to the screen which is responsible for creating a new event, learn goal or note. This is described in section 4.3.2. Beneath the top bar, there is a UISegmentedControl. This UISegmentedControl is used to switch between the different types of events and learn goals. On this screen, the user can decide between outstanding, completed and all events. The logic for the UISegmentedControl is located in the UIViewController. The UIViewController holds a list which is connected to the UITableView. Depending on what element of the UISegmentedControl the user chooses, the UIViewController is changing the content of the list which is connected to the UITableView. If a user clicks on the element "Outstanding", the UIViewController replaces the list with the corresponding list from the ViewModel. The ViewModel is responsible for loading the events and learn goals. The ViewModel is receiving the events from the backend of the MOOC platform and the local Realm Database. After the ViewModel has received all the necessary data, it is sorting the events and learn goals. Depending on which type they are (outstanding or completed) the ViewModel is assigning the events to different lists. The UITableView makes use of several methods. One method used is the method which tells the UITableView how each element looks like. Therefore, the method defines a UITableViewCell which is defined in the Storyboard. The UITableViewCell is configured by an ItemViewModel. The method provides a parameter cellForRowAt. This defines which row the TableView is currently defining. With this value, the method takes the element from the list connected to the UITableView which index is equal to the value and creates an ItemViewModel. This ItemViewModel works similar to the ItemViewModels used in Android. It provides functions to receive the information needed like the title, description, due date or time of the event. With this ItemViewModel, the UITableView method can define the UITableViewCell. The UITableViewCell uses the ItemViewModel to retrieve information and forward it to the View. This way, the iOS version is mocking the data binding behaviour of the Android version and this approach

#### 4. Developing the applications

is used among the whole iOS application. Another method used for the UITableView tells the UIViewController what happens if a user clicks on an item in the UITableView. The method provides the value didSelectRowAt. With this value, the method receives the chosen event from the list connected to the UITableView. The method then pushes a new UIViewController to the UINavigationController, which leads the user to a new screen. This new screen is responsible for showing a detail view of the event or learn goal. This screen is described in Figure 4.15.

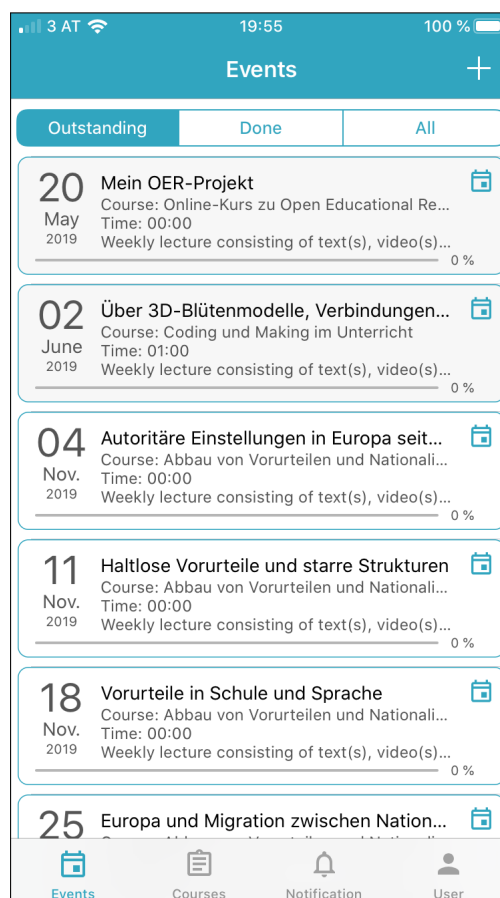


Figure 4.11.: List of events and learn goals in iOS application



### Detail View - Event

If a user clicks on an item described in section 4.3.2, she/he is redirected to a new screen. This new screen contains a detailed overview of the event, learn goal or note. In terms of the application, this screen is realised by two `UITableViewController`s. One `UITableViewController` controls the logic if the item is an event or learn goal and the other `UITableViewController` is opened if the item is a note. If the item chosen is of type event or learn goal, the detail view looks different than if a note is chosen. In the `viewDidLoad` method of the `UITableViewController`, the logic checks whether the event or learn goal chosen is created from the user or not. Depending on that, the `UITableViewController` loads the event or learn goal from the Realm Database. After that, a `ViewModel` is created with the loaded event or learn goal. The `ViewModel` supports the `UITableViewController` and contains functions that the `UITableViewController` calls to fill the element in the View. For example, the `ViewModel` provides functions to receive the course title of the event or the colour if predefined by the user. The `UITableViewController` contains a connection to all Labels and Views in the Storyboard. The `UITableViewController` adds a `UITapGestureRecognizer` to the label and image for the reminder date. If a user clicks on the label or the image, she/he is redirected to a new `UITableViewController`. This `UITableViewController` provides only one functionality. It makes it possible to choose a date. If a user has chosen a date, the application loads the previous screen again and stores the date chosen. Furthermore, the detail view screen provides the possibility for the user to trigger a Switch when the event or learn goal is completed. If a user has created an event or a learn goal and then completes it, she/he is getting experience points for the gamification part which is described in section 3.2. At the bottom of the screen, the `UITableViewController` holds a button that is changing depending on the origin of the event or learn goal. If the event or learn goal is received by the backend of the MOOC platform, the button says "Go to Event". If a user clicks on it, the `UITableViewController` opens the URL of the event via `UIApplication`. If it is not received from the backend of the MOOC platform it must have been created by the user. If so, the button says "Delete". If a user then clicks on the button, the `UITableViewController` loads an `UIAlertController` which asks the user for confirmation of the deletion. If the user hits cancel on the `UIAlertController`, the `UIAlertController` is closed and

#### 4. Developing the applications

nothing happens. If the user clicks "OK", the event or learn goal is deleted and the user is lead to the previous screen. The UIViewController also adds a button to the top of the screen which says "Save". If a user has changed some data of the event or Learn Goal, she/he has to click this button to store the information. If a user clicks the button, the UIViewController checks for the changes the user has made and stores the adjusted event or learn goal in the Realm database.

The other UIViewController which is responsible for displaying a detail view for notes is a light version of the already explained UIViewController. In this UIViewController there is no need for dates, time and other information like explained in section 3.3. A note contains a title, a description and a course that it is assigned to. The UIViewController just loads the corresponding note from the Realm database. In this case it makes no sense to analyse where the note is coming from since a note can only be created by a user within the application. After it is loaded, a ViewModel is created with the given note. With the support of the ViewModel the Labels and Views are adjusted. On this screen there is no need for a "Save"-button at the top of the screen, because the user cannot change the note or any information. At the bottom of the screen there is a button which says "Delete". If a user clicks on it, the UIViewController loads a UIAlertController and asks the user for confirmation of deletion. If the user clicks "OK" the note is deleted. If the user clicks "Cancel" the UIAlertController is closed an nothing happens. To make it possible to delete a note, event or Learn Goal, the user first has to create one. This creation process is described in the following section 4.3.2. The Figure 4.12 shows how the screen described in the current section looks like in the iOS version of the application.

#### **Create Event**

As described in section 3.3, the user is able to create events, learn goals and notes within the application. By clicking the "plus"-sign at top right corner of the screen, the user gets redirected to the screen where she/he can create an event, learn goal or note. At the top of the screen, there is a UISegmentedControl. With this UISegmentedControl, the user can choose whether she/he wants to create an event, a learn goal or a note. The different

### 4.3. iOS

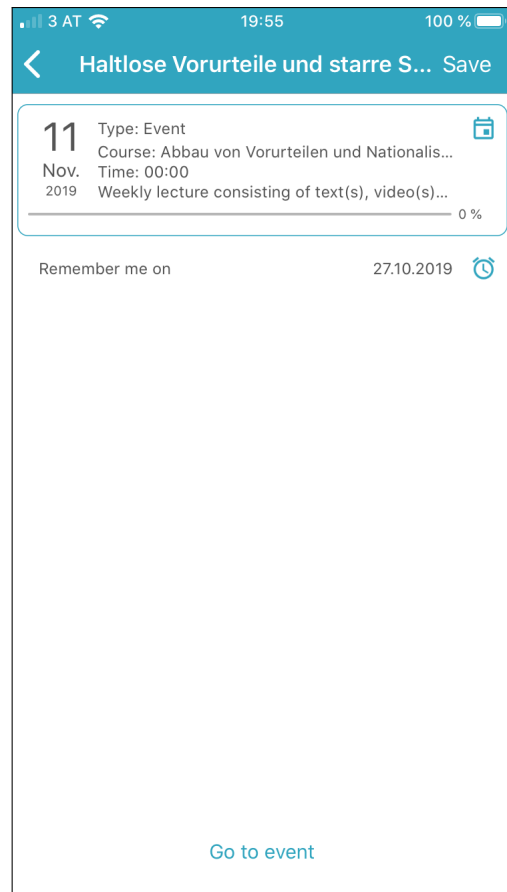
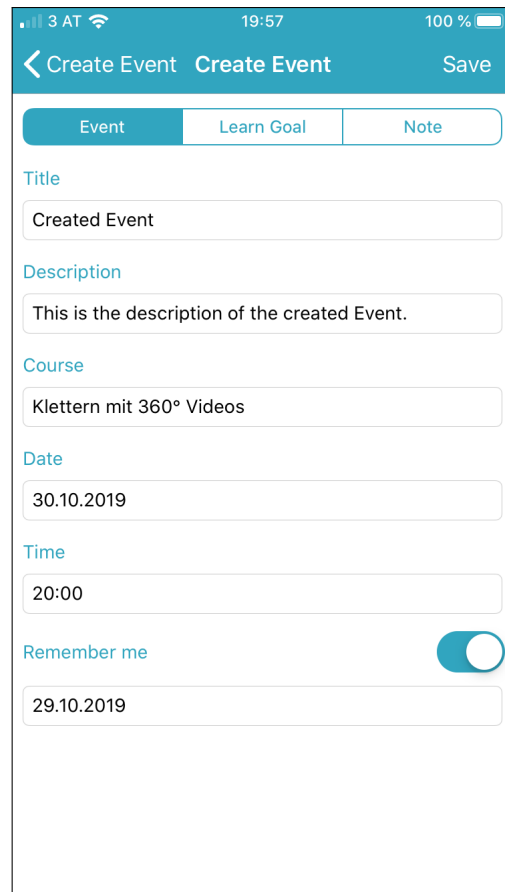


Figure 4.12.: Detail view event - iOS application

#### 4. Developing the applications

UIViews defined in the Storyboard are connected to an UIViewController. This UIViewController, in combination with a ViewModel, holds the logic for the control of the screen. Depending on what the user chooses in the UISegmentedControl, the UIViewController changes the title of the screen. Furthermore, since for a note there is no need for information like due date, time and the reminder feature, the corresponding elements get hidden if the user chooses to create a note. To retrieve information for the title and the description of the event, the application makes use of UITextFields. If a user taps on the UITextField responsible for choosing a course, the UIViewController triggers a function that builds an UIAlertController. This UIAlertController contains all courses that the user has subscribed to. To create an event or a learn Goal, the user has to choose one course she/he wants the event or learn goal to be assigned to. For choosing a time or a due date, the screen makes use of two more UIViewControllers. If a user clicks on the UITextField of the time, the UIViewController opens a new UIViewController which contains a UIDatePicker to choose a time. If a user has chosen the time and hits the UIButton, which is at the bottom of the screen, the UIViewController gives the chosen time to the UIViewController responsible for creating events. Then the UIViewController gets popped. The same process triggers if a user clicks on the UITextField responsible for displaying the date, only that the loaded UIViewController holds a UIDatePicker configured to choose a date. At the bottom of the form the user can trigger a switch, whether she/he wants to get reminded for the created event or learn goal. If the user toggles the UISwitch to true, another UITextField appears on the screen. If a user clicks on the UITextField, she/he gets redirected to the UIViewController responsible for picking a date. The UIViewController used for creating an event adds one UIBarButtonItem to the top navigation bar. This UIBarButtonItem says "Save". If the user clicks on it, the UIViewController checks whether the input values are valid, stores the event, learn goal or note in the Realm Database and informs the user about the successful storage. If the input values are not valid, the UIViewController loads a UIAlertController which informs the user about the invalid parameters. Figure 4.13 shows a part of the screen which is responsible for creating an event, learn goal or note.

### 4.3. iOS



The screenshot shows the 'Create Event' form in an iOS application. The form is titled 'Create Event' and has a 'Save' button. It features three tabs: 'Event', 'Learn Goal', and 'Note'. The 'Event' tab is selected. The form contains the following fields and controls:

- Title:** Created Event
- Description:** This is the description of the created Event.
- Course:** Klettern mit 360° Videos
- Date:** 30.10.2019
- Time:** 20:00
- Remember me:** A toggle switch is turned on.
- Additional Date:** 29.10.2019

Figure 4.13.: Creating event, learn goal and note - iOS application

## 4. Developing the applications

### **Courses**

This screen is responsible for showing all courses to the user she/he has subscribed to. It is part of the main navigation UINavigationController. The screen itself makes use of an UINavigationController which contains the UITableViewDataSource and UITableViewDelegate. The UITableView defined in the Storyboard is connected to the UINavigationController. At the start of the UINavigationController, the corresponding ViewModel gets initialised and the UITableView gets configured. The setup of this UINavigationController is the same as the UINavigationController described in section 4.3.2. When the UINavigationController is started, the initialised ViewModel shows the loading animation and loads all courses from the backend of the MOOC platform. After successfully retrieving the response, the ViewModel compares the courses in the Realm Database with the courses received from the backend. It then compares both lists with each other and checks whether there is a colour set by the user. If so, the course is adapted accordingly and gets added to the list which is connected to the UITableView. The ViewModel also contains a function getItemViewModel(forRow: Int). With this function, the UITableView can create the needed ItemViewModel for the course it is currently adding to the screen. The ItemViewModel is needed to define the UITableViewCell. The ItemViewModel contains functions that define title, border colour, icon colour, description and progress of a course. These information are displayed to the user. If a user clicks on one of the courses in the UITableView, she/he is redirected to the screen described in ???. The Storyboard also contains a UIView which holds an UILabel. Both are connected to the UINavigationController. These Views are used to display a message to the user if she/he is not subscribed to any courses. The principle of the UINavigationController, the ViewModel and the ItemViewModel is the same as within section 4.3.2. The Figure 4.14 shows an example of how the screen may look like during the usage of the application.

### **Detail View - Course**

If a user clicks on an item in the list view described in section 4.3.2 she/he gets redirected to the screen which is responsible for displaying all events, learn goals and notes related to the chosen course. This screen consists

### 4.3. iOS



Figure 4.14.: List of subscribed courses in iOS application

#### 4. Developing the applications

of four Controllers. The first Controller is a UITabBarController, which is responsible for the navigation within this feature. This UITabBarController holds three items, which all lead to the other three UIViewController. In the logic of the UITabBarController, some details for the selected course are set. For each of the three possible UIViewController the user can choose, the UITabBarController sets the title of the course and the identifier of the course, so the UIViewController can load the corresponding events, learn goals and notes. By default, the UIViewController responsible for showing the corresponding events of the course is selected and shown to the user. The UIViewController contains a UISegmentedControl, an UITableView and an UILabel responsible for showing an error message if now events are available for the selected course. The UISegmentedControl works the same as the one in section 4.3.2. Depending on which item ("Outstanding", "Done", "All") the user chooses from the UISegmentedControl, the corresponding ViewModel loads the related events from the Realm Database and sets the result to the list connected to the UITableView. For example, if a user chooses the item "Done" from the UISegmentedControl, the ViewModel loads all events from the Realm Database which are connected to the course and which are already completed. The same procedure holds for the items "Outstanding" and "All". If a user touches an item in the UITableView, which shows all events to the user with its own UITableViewCell, she/he gets redirected to the screen described in section 4.3.2. The UITableViewCell is the same like the one described in section 4.3.2. The UITableViewCell gets filled with data by an ItemViewModel which gets created by the ViewModel of the UIViewController. So when the UITableView is loaded, it iterates over the whole list which is connected to it. For each item in the list, the UIViewController creates an ItemViewModel with the help of the ViewModel. This ItemViewModel holds functions which can fill the UITableViewCell. With this procedure, every UITableViewCell gets filled with the information of an ItemViewModel. The UIViewController responsible for displaying the learn goals to the user works nearly the same like the UIViewController responsible for the events. The difference is, that it does not load the corresponding events, but the corresponding learn goals of the chosen course. The third and last UIViewController making this feature complete is the one responsible for displaying all notes to the user. This UIViewController does not need an UISegmentedControl, since it is not possible for notes to be "Completed" or "Outstanding". Besides the missing UISegmentedControl,



this UIViewController also follows the same approach like the UIViewControllers already described, except that it does only load the notes from the Realm Database once, since the list cannot change during runtime of the application. The features described in this section also provide one further functionality. The UITabBarController creates an UIBarButtonItem with the barButtonItemSystemItem “.add” and defines it as a rightBarButtonItem in the navigationItem. If a user clicks on this UIBarButtonItem, the UITabBarController loads the screen described in section 4.3.2. The difference here is, that the UITabBarController sets information about which type of event the user wants to create. Depending on which UIViewController is currently loaded in the UITabBarController, the UITabBarController gives the information to the UIViewController responsible for creating new events, learn goals and notes. This way, the UIViewController already chooses the correct type and automatically fills out the course the UITabBarController was loaded for. For example, if a user has chosen the UIViewController responsible for displaying all notes to the user and touches the UIBarButtonItem, the screen loaded also has chosen the type “Note” and automatically filled out the course for the user. For further description see section 4.3.2. The Figure 4.15 shows the screen described in this section.

## Notification

This application makes use of notifications in two ways. The first way is to notify the user with push notifications about some event or learn goal. The other one is to display all notifications to the user within the application itself, so the user can always view her/his notifications. The push notification is triggered within the AppDelegate.swift. First, it loads all events and learn goals from the Realm Database which have the current date as due date. For each loaded event and learn goal, the background fetch checks whether the user has already been notified regarding this event or learn goal. This prevents the user from getting notified about the same event and learn goal multiple times. If the user has already been notified, the for loop skips an iteration. If the user has not been notified about this specific event or learn goal, the title and the description for the notification is set, respecting the language of the device. After that, a notification is pushed to the user device, notifying her/him about the event or learn goal.

#### 4. Developing the applications

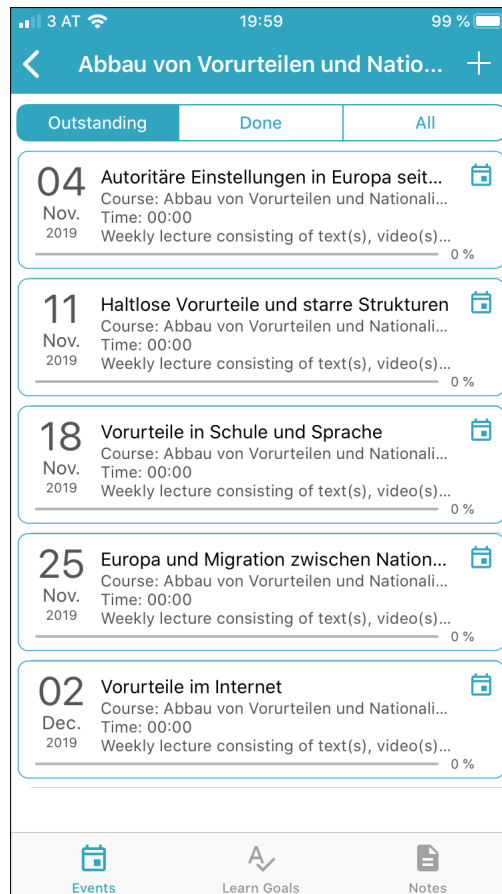


Figure 4.15.: List of events of specific subscribed course - iOS application

Furthermore, a new RealmObject holding information about the event or learn goal is created. This RealmObject is stored to the Realm Database, so the application can later on show the user the notification and inform her/him about new notifications. After the notifications for the events and learn goals has been sent out, the background fetch starts an API request. This request checks, whether there are any new courses that the user should be notified about. If the request is successful and there are new courses the user should be informed about, the background fetch iterates over all the courses. Same as with the events and learn goals, the background fetch checks whether the user has already been notified about the new course. If not, the background fetch again builds a notification with a title and a description and pushes it to user. Again, a RealmObject with the information of the new course is stored to the Realm Database. If the user opens the application, the main UITabBarController checks whether there were any new notifications since the user last opened the UIViewController responsible for showing the notifications. This is checked with the help of the Realm Database. If there are any new notifications, the UITabBarController sets a badgeValue to the notification UITabBarItem. If a user opens the UIViewController for the notifications, the UIViewController removes this badgeValue from the UITabBarItem. Furthermore, this UIViewController contains the logic for displaying the notifications which have been stored in the Realm Database by the background fetch. The UIViewController contains a UITableView. The corresponding ViewModel loads all notifications from the Realm Database and stores the resulting items in the list which is connected to the UITableView. This UITableView builds the items with the help of a UITableViewCell and an ItemViewModel created by the ViewModel. The exact procedure has already been described in section 4.3.2. Depending on whether the user has been notified about an event, a learn goal or a new course, the icon and the text displayed in the UITableViewCell changes. The Figure 4.16 shows the screen which displays all notifications to the user.

## Profile

The section for the profile is part of the main UITabBarController. This UIViewController does not contain a UITableView like the other UIViewController, but it also makes use of a ViewModel. This ViewModel is

#### 4. Developing the applications

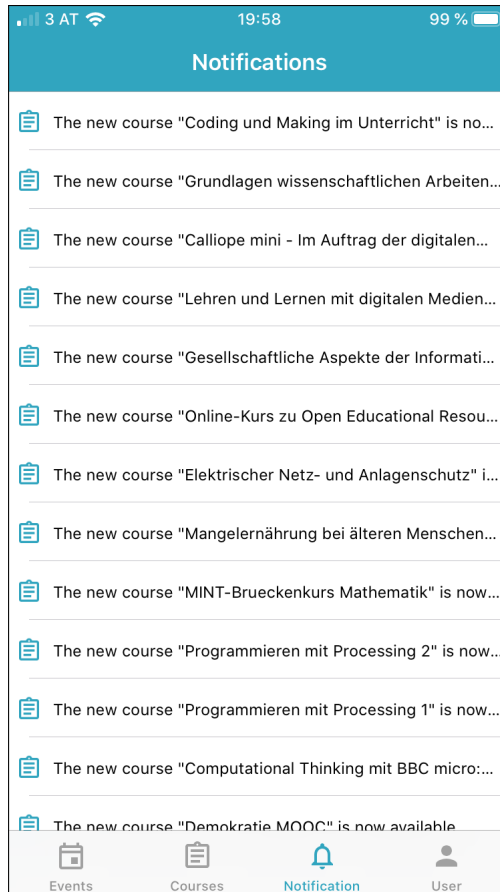


Figure 4.16.: List of received notifications in iOS application

responsible for handling the gamification part of this screen, which is settled at the top of the screen. First, the ViewModel loads the current level of the user from the Realm Database. The user gains experience points for raising the level by finishing events and learn goals. When the level is read from the Realm Database, an API request is started. This request contains the current level of the user. The response contains information about the icon and the text that is displayed to the user. When the request finished successfully, the information gets delivered to the View. The View loads the icon, level, corresponding text and experience points from the ViewModel. The section below the part for the gamification contains eight UIViews. Each of these UIViews is connected to the UIViewController and holds a Tap Gesture Recognizer. The UIViews for "All Courses", "iMoox.at", "Master Thesis Feedback", "Imprint" and "Data Privacy" all open the browser of the device and load a corresponding URL. The UIView for "Colors" loads the screen which is responsible for the feature of colouring the courses, events and learn goals. This feature is described in section 3.3 and Figure 4.18. The UIView for "Play Tutorial" starts the tutorial of the application. The UIView for "Sign Out" stores the value nil for the authorisation token in the UserDefaults and loads the login screen described in section 4.3.2. Figure 4.17 shows the screen of the profile section and all its elements.

### Coloring

If a user touches the UIView for "Colors" in the profile section described in section 4.3.2, she/he is able to use the feature for colouring the courses, events, learn goals and notes described in section 3.3. This feature consists of two UIViewControllers and one UITableViewCell. If a user clicks on the section described before, she/he gets redirected to a screen which lists all courses the user has subscribed to. The corresponding ViewModel starts a request when the UIViewController is loaded. The response of the request contains all courses the user is assigned to. With the help of the ViewModel, the UITableView, settled in the UIViewController, is able to create an ItemViewModel. This ItemViewModel is used by the UITableViewCell to receive the information needed to display the courses to the user. In this case, the UITableViewCell contains an UILabel and an UIImageView. The UILabel contains the name of the course and the

#### 4. Developing the applications

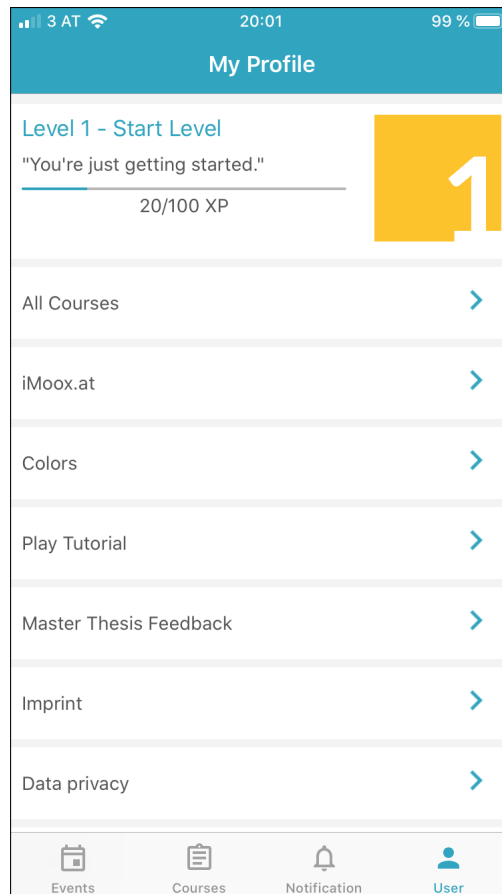


Figure 4.17.: Profile including gamification - iOS application

UIImageView is an arrow which indicates that a click on the item leads to the next step. Furthermore, the UIViewController contains an UIView that contains an UILabel which displays an error message to the user in the case that the request started by the ViewModel fails or the returned list of courses is empty. If a user clicks on an item in the UITableView, she/he gets redirected to the UIViewController which makes it possible for the user to choose an UIColor for the selected course. The UIViewController contains seven different UIViews. Each one of the UIViews contains a background colour and an UIImageView. The background colour shows the UIColor the user can choose. The UIImageView shows a tick to indicate the user which UIColor she/he has currently chosen. Each UIView has a Tap Gesture Recognizer on it. For example, if the user clicks on the second UIView, the UIViewController disables all ticks and enables the tick for the UIView the user has touched. Furthermore, the UIViewController adds a UIBarButtonItem to the navigationItem which says "Save". If a user has finished the selection process, she/he can touch the UIBarButtonItem to save the UIColor as Hex-Code to the chosen course in the Realm Database. This way, the UIColor is stored within the RealmObject and the application can later on set the UIColor for the course, events, learn goals and notes. After the successful data storage process the user is redirected to the screen before, which is the UIViewController responsible for listing all courses the user has subscribed to. The user can then carry on to assign UIColors to more courses or navigate through the application. The Figure 4.18 shows the screen of the application where the user can choose the colours for the courses.

#### 4. Developing the applications



Figure 4.18.: Colouring - screen of iOS application



## 5. Proof of concept

This chapter provides a summary of the ideas and features targeted at the phase of the prototype. The original ideas are defined in the chapter 3. Based on the ideas gathered before the development of the application, this chapter defines which goals have been fulfilled, which goals have been partly fulfilled and which goals have not been realised. Furthermore, this chapter provides a summary of the survey which has been generated and given out to the users of the application to give feedback about the application. This survey indicates whether the application and its features have been successfully implemented, partly successfully implemented or poorly implemented.

### 5.1. Prototype vs. application

This section provides a detailed comparison between the prototype designed before the development of the application and the application itself. At first, all features from the prototype process are listed in a table and whether they have been implemented, partially implemented or not been implemented in the application. After that a description for each feature listed in the table and why it has been successfully or only partly implemented follows. The next part of this section describes each feature. It describes how the feature was supposed to work and how it is implemented in each version, Android and iOS, of the application. Features in the application may differ from the prototype and this section describes why it was decided to change the feature.

## 5. Proof of concept

<b>Feature</b>	<b>Android</b>	<b>iOS</b>
Offline usage	poorly	poorly
Coloring of courses	yes	yes
Coloring of events	yes	yes
Coloring of learn goals	yes	yes
Coloring of notes	yes	yes
Complete self created events	yes	yes
Complete self created learn goals	yes	yes
Create events	partly	partly
Create learn goals	partly	partly
Create notes	partly	partly
Detail view of courses	yes	yes
Detail view of events	yes	yes
Detail view of learn goals	yes	yes
Detail view of notes	yes	yes
Filtering events	partly	partly
Filtering learn goals	partly	partly
Gamification	yes	yes
Learning progress	partly	partly
List courses	yes	yes
List events from MOOC platform	yes	yes
List self created events	yes	yes
List notifications	yes	yes
Login	partly	partly
Multilingual	partly	partly
Push notifications	partly	partly
Type icons	yes	yes

Table 5.1.: Comparison of features described during the prototype process and features implemented in the application

### **Offline usage**

Offline usage was a part of the discussion during the prototype process. It was planned to store information in a local database on the device, but not to which extent. It should be possible to use the application offline. This feature has not been sufficiently implemented. This is due to the fact, that for the optimal usage of the application, a stable internet connection is needed. Even though the data is stored locally, an adjustment with the external database of the MOOC platform is needed to update the events, courses and progress. The external database of the MOOC platform controls these elements. This part has been implemented sufficiently but needs improvement. What is possible, is that the user can see her/his events, learn goals, courses and notes if she/he has used the application once and retrieved information successfully, but the data is then out-dated.

### **Coloring of events, learn Goals, notes and courses**

During the prototype process, the colouring of events, learn goals, notes and courses was described as a feature that helps the user to interact with the application and differ between the courses. This feature should make the application more readable. This feature is located in the profile screen of the prototype mockup. This feature has been successfully implemented and the feature works as it was supposed to.

### **Complete self-created events and learn goals**

This feature was described during the prototype process as the possibility of finishing self-created events or learn goals. If a user has finished a self-created event or learn goal, she/he should be able to complete the task within the application and receive experience points for it. Experience points are described in section 3.2. This feature has been successfully implemented in the applications for Android and iOS and is located in the detail view of self-created events or learn goals.

## 5. Proof of concept

### **Create events, learn goals and notes**

As defined in subsection 3.1.3, besides the events retrieved from the MOOC platform, the user should be able to create events, learn goals and notes if needed. In the Table 5.1, these three types are defined as partly implemented. This is due to the fact, that the design was changed during the development process. During the prototype process, the creation screen was designed to match the design of events that are displayed to the user on the events screen, described in section 3.3. Originally it was thought that the user has a feeling of how the event, learn goal or note will look in the application after creating it. This has changed due to the fact of bad user experience since giving input data needs more space than displaying information. Therefore this screen has changed to a form. Due to this fact, this feature is described as partly successfully implemented. The key feature is still available, but the design of the creation process has changed.

### **Detail view of courses**

The detail view of courses is accessible from the screen showing a list of all subscribed courses. By tapping on a course in the list, the user is redirected to a screen where she/he can see all events and learn goals that are related to the subscribed course. The feature includes an own and independent Bottom Navigation. The tabs describe "Events", "Learn Goals" and "Notes", so the user can differ between every type. Furthermore, events and learn goals can be filtered in more detail. As described in the previous section, filtering changed during the development process. This feature is described as successfully implemented in the Table 5.1, since it works as intended. Furthermore, the user can be redirected to the screen where she/he can create an event, learn goal or note. Depending on which course the user has chosen, the application automatically fills out the creation screen, as described in section 4.3.2 and section 4.2.2. This feature works as described in the prototype process.

### **Detail view of events, learn goals and notes**

During the prototype process, a feature was defined that should display a detailed view about the event, learn goal or note, no matter whether self-created or retrieved by the MOOC platform. The detail view of events should also include the feature to determine a reminder date. On this screen, the user should have the possibility to set a date with the help of an operating system specific date picker. Furthermore, in the case of self-created events or learn goals, the user should have the possibility to mark them as completed if necessary. Notes differ from events and learn goals. In this case, the user cannot set a reminder date or mark them as completed. The detail view of notes is defined with a title and a description. In the case of self-created events, learn goals or notes, the user also can delete them if necessary. This feature has been implemented successfully in the application and is therefore marked so in the Table 5.1.

### **Filtering events and learn goals**

The idea during the prototype process was, that the user should have the possibility to filter the events and learn goals in the list views. This should help the user to intentionally navigate through the application and find the events/learn goals she/he is looking for. In Table 5.1 this feature is marked as partially implemented in the applications for Android and iOS. This is due to the fact, that the feature changed during the development process and was supplemented by another feature. In the prototype process, the possibilities for filtering events were limited to "Upcoming", "Past" and "All". During the development process, these filter elements were changed to "Completed", "Outstanding" and "All". This leaves the user with the possibility to filter for tasks that she/he has not finished, more comfortably. The supplementing feature is that the user is still able to indicate which due date of events or learn goals is in the past. This is done by a grey overlay in both versions of the application. Even though it improves the feature itself, it is marked as partly implemented for both versions, since it differs from the idea which was defined during the prototype process.

## 5. Proof of concept

### **Gamification**

During the prototype process, gamification was one of the main aspects of the application and needed more attention than other features. There have been many ideas about how to solve this problem. One described idea in chapter 3 was, that the application makes use of avatars combined with experience points the user can gather. Because creating and drawing avatars does not suit this application, the prototype process decided that for this master thesis, it is satisfactory that the application uses levels and experience points. Depending on the current level of the user, the application retrieves dynamic information about the level from the backend of the MOOC platform, like an individual text that describes the level and an image that is shown to the user. This was the accepted idea during the prototype process and this idea has been successfully implemented in both applications for Android and iOS.

### **Learning progress**

The learning progress feature defined during the prototype process differs from the feature implemented in the applications for Android and iOS. Originally, it was defined as a feature that is located in the profile screen of the application. There, the user should receive a list of all events, learn goals and courses and see her/his progress and the progress of others. During the development process, this screen was removed and the feature was moved somewhere else. The feature is still in the applications, but it is located directly within the events, learn goals and courses. With this change of the feature, one screen was removed and now the user has the possibility to see the courses, events and learn goals with their progress and other details. This feature is marked as partly successfully implemented since it differs from the idea given during the prototype process. The key idea of the feature still exists in the applications and the user can view her/his progress.

### **List courses**

In the prototype process, listing courses is one of the main screens in the main navigation. It is one of the four tabs in the Bottom Navigation. This screen should list all courses the user has subscribed to, providing the title of the course, a description and a image. If no image is available, a standard image should be displayed. This feature is provided, so the user can see which courses she/he is assigned to with one click. Furthermore, the user should have the possibility to touch one of the courses and be redirected to a screen which provides all events, learn goals and notes related to this course. This is described later in this chapter. This feature has been successfully implemented in the application for Android and iOS and works as intended.

### **List events from MOOC platform**

During the prototype process, listing events was defined as the possibility for the user to check all events that are available at the MOOC platform. A user, who has subscribed to courses, has events that take place during a course. These events should be listed to the user. This feature has been successfully implemented in the Android and iOS version of the application. Furthermore, during the process of developing the applications, the feature was introduced to enable the user to see which events' due dates are already past and which are present or in the future, at first sight. This feature was implemented afterwards. In addition to that, both applications in Android and iOS, provide the feature, enabling the application to automatically scroll to the event which is the nearest to the current date. This is also a feature that was not mentioned in the prototype process but came up during the development process. In respect to the explanation given in section 4.3.2 and section 4.2.2, this feature is defined as successfully implemented and compared to the prototype, it comes with two additional defined features.

## 5. Proof of concept

### **List self-created events and learn goals**

Besides the listing of events available at the MOOC platform, the user should be able to see self-created events and learn goals. These self-created events and learn goals should be combined with the events retrieved from the MOOC platform. They should act in the same way as events from the MOOC platform, except for the possibility that the user can also create a learn goal and can complete the event or learn goal by her/his own manually. This idea was planned during the prototype process and has been successfully implemented in the applications for Android and iOS.

### **List notifications**

As described in the section for section 5.1, listing notifications is the second part of the notifications feature. In the prototype process, it is defined as the listing of all notifications that have been pushed to the user. This feature should help the user not to miss any notifications and to review them again. It is also defined as one of the main screens of the navigation and is located as a tabbed item in the Bottom Navigation. This feature has been successfully implemented, including the dynamically created text and icon depending on the type of the push notification, and therefore is marked as successfully implemented in Android and iOS in the Table 5.1.

### **Login**

During the prototype process, the login was defined as a native screen where a user can log in to the application or register. The login is necessary to receive an authorisation token which is needed to successfully receive information about the courses the user has subscribed to. In Table 5.1, this feature is marked as partly implemented in Android and iOS. It is marked as partly implemented because the login screen is native, but is located inside a WebView, which is not optimal. This is because the complex authorisation process of the MOOC platform and building an own native login process for the application would have caused security issues. That is why the application does not have a login process and login screen



## 5.1. Prototype vs. application

but uses the login and registration process of the MOOC platform itself with the aid of a WebView. Even though it is not optimal, the login and registration process works and the user can retrieve an authorisation token to successfully interact with the application. This feature is therefore defined as partly successfully implemented.

### **Multilingual**

During the prototype process, it was considered that the application has to be multilingual to satisfy the requirements of the users. Therefore it was planned during the prototype process to provide applications that offer the languages English and German. Both applications, Android and iOS, make use of both languages as planned, but there is room for improvement. To satisfy more needs of the users and to adapt to the native languages of the users, further translations of the applications are required. This feature is marked as partly successfully implemented because it implements the idea of the prototype but still needs improvement.

### **Push notifications**

In chapter 3, notifications have two different meanings. On the one hand, push notifications are described. These push notifications get sent to the user when something new is happening. The other type is the screen of notifications. This screen lists all push notifications the user has received. This section describes the part for the push notifications. The push notifications are marked as partly implemented in Android and iOS in Table 5.1. During the prototype process, the push notifications were planned to take place as soon as something new happens. In this case, the notifications should be sent to the user as soon as a reminder date for an event or a learn goal arises or a new course has been provided by the MOOC platform. In both cases, it was not possible to implement it in a way that real-time push notifications are possible. Since this was not possible, the feature was changed. Push notifications are sent once a day in both versions, Android and iOS. The application retrieves the data once a day, analyses it and according to the data reacts with push notifications to it if needed. The feature is not fully

## 5. Proof of concept

working as intended. Therefore this feature is defined as partly successfully implemented.

### **Type icons**

The prototype process has shown that it is hard for the user to determine whether an item is an event, a learn goal or a note if no additional information is provided. Therefore, during the prototype process, the idea was born to differ the types of items with the aid of icons. Each type, event, learn goal and note, no matter whether self-created or not, has its own icon to show which type the item is of. Furthermore, courses also have own icons. This is useful when the user is heading to the screen which lists notifications. At this screen, the user can differ whether a notification was about an event, a learn goal or a course. This feature has successfully been implemented in both versions of Android and iOS. Furthermore, both versions share the same set of icons.

In sum, 25 features are listed in Table 5.1. It is debatable whether there are more features implemented, but these are the core features to make the applications useful. These core features were designed in the prototype process and this chapter declares to which rate these features have been implemented in the applications for Android and iOS. 16 features are defined as successfully implemented. Eight features are defined as partly implemented. One feature, the offline usage of the application, is defined as poorly implemented and needs improvement. In general, the application satisfies the needs given in the prototype process and therefore one part for the proof of the concept is given. The second part of this chapter provides a survey and the final evaluation whether the concept has proven successful.

## **5.2. Survey & Evaluation**

This section provides a survey which was provided to the users of the application and the evaluation of the answers given from the users. It finishes the first part of the proof of concept of this chapter. The first part of

this chapter defines the idea of the prototype as successfully implemented. This section evaluates whether the ideas given in the prototype and the implementation in the applications have led to the intended goal. The goal of the application is to give users an overview of their courses and events and improve their learning progress. This section gives answers to the question, whether the design and the implementation is satisfying the users. Furthermore, it defines the final question whether the users could/can improve their learning progress with the aid of the mobile applications.

### 5.2.1. The survey

The survey started on the 5th of October 2018 and ended on the 17th of October 2019. The survey was distributed to the users via the applications, the forum of the MOOC platform and in MOOC on the platform. This section provides the questions asked during the survey and the meaning behind the answers. Furthermore, it describes why the questions were asked and why they were asked this way. The survey was provided to the users with the aid of Google forms. The survey created there is available in the applications. In the section for the profile, there is a subsection for "Master Thesis Feedback". If users click on this section, they will be redirected to the survey. Furthermore, people are notified ten days after the first installation of the application. The notification includes a hint, that the applications were part of a Master thesis and the users can answer a survey to prove the concept of this application. In sum, the survey contains ten questions. Eight of these questions are single choice questions with four possible answers and are obligatory. Two of the ten questions can be filled out optionally. This means, that the users can provide any feedback in these questions. It is not defined whether the users use the application on an Android or iOS device since the core functionality and the design does not differ. In the following, all questions and possible answers are explained in detail.

#### **Question #1: Do you like the design/appearance of the app?**

This question is asked to clarify whether the design of the application is pleasing the users. The design is one main part of this application. With

## 5. Proof of concept

this question, users can define whether they like the way that events, learn goals and notes are displayed, whether the colour scheme is appealing, whether everything is readable, whether the icons are obvious and whether they like how information is displayed to them. The possible answers for users to give ranges from #1 to #4. #1 is defined as "No, I don't like it at all". #4 is defined as "Yes, I like it". So if a user does not think that the design/appearance of the application is appealing, she/he must pick the answer #1. If a user somewhat likes the design/appearance of the application, she/he must pick the answer #2. If a user likes the design/appearance of the application, she/he must pick the answer #3. If a user is fully convinced by the design/appearance of the application, she/he must pick the answer #4.

### **Question #2: Do or did you have problems controlling the app?**

This question is asked to clarify whether users have or had general problems controlling the application. This question is not meant to define whether the users are pleased with the navigation controls of the application. For this definition, question #3 is asked. This question defines whether the users know how to reach the detail view of events or they know how to create an event, learn goal or note. In general, it should define whether the users know where to find which feature and whether they know all features by clicking once through the whole application. The possible answers for users to give range from #1 to #4. Answer #1 is defined as "Yes, I had many problems". Answer #4 is defined as "No, no problem at all". So if a user had or has many problems when controlling the application, she/he must pick the answer #1. If a user had or has some problems when controlling the application, she/he must pick the answer #2. If a user had or has minor problems when controlling the application, she/he must pick the answer #3. If a user had or has no problems when controlling the application, she/he must pick the answer #4.

### **Question #3: Do you think the navigation of the app is confusing?**

This question is asked to clarify whether users have problems when navigating through the application. In contrast to the question #2, this question does not define whether users have problems controlling the application. This question defines whether users think that they understand how the Bottom Navigation works, how navigating back works or whether they understand what happens when they click on an item in a list. The possible answers for users to give range from #1 to #4. Answer #1 is defined as "Yes, I don't know how to handle it". Answer #4 is defined as "No, the app is self-explaining". So if a user does not understand how the navigation in the application works, she/he must pick the answer #1. If a user somehow understands how the navigation in the application works, she/he must pick the answer #2. If a user has minor problems with the navigation in the application, she/he must pick the answer #3. If a user has no problems with the navigation in the application, she/he must pick the answer #4.

### **Question #4: Do you use the app at least once a week in addition to an iMooX-Course?**

This question is asked to clarify whether users think that this application is useful to use it in addition to the existing MOOC platform. Furthermore, it defines, how often the users are using the application in addition to a subscribed course of the MOOC platform. This question does not define how often the users use the application in general since it explicitly says "in addition to an iMooX-Course". The possible answers for users to give range from #1 to #4. Answer #1 is defined as "No, never". Answer #4 is defined as "Yes, always". So if a user does not regularly use the application within a week in addition to a course at the MOOC platform, she/he must pick the answer #1. If a user sometimes regularly uses the application within a week in addition to a course at the MOOC platform, she/he must pick the answer #2. If a user often regularly uses the application within a week in addition to a course at the MOOC platform, she/he must pick the answer #3. If a user always regularly uses the application within a week in addition to a course at the MOOC platform, she/he must pick the answer #4.

## 5. Proof of concept

### **Question #5: After playing the tutorial, did you have the feeling you understand the app?**

At the first opening of the application, the user can play a tutorial. The user also can play the tutorial as often as she/he wants with the designated section within the profile screen of the application. This question defines whether users had the feeling they understand the purpose of the application and what it was designed for after they have clicked through the tutorial. The possible answers for users to give range from #1 to #4. Answer #1 is defined as "No, I didn't have a clue". Answer #4 is defined as "Yes, I understood it immediately". So if a user had many problems to understand what the purpose of the application is, she/he has to pick answer #1. If a user had some problems to understand what the purpose of the application is, she/he has to pick answer #2. If a user had minor problems to understand what the purpose of the application is, she/he has to pick answer #3. If a user had no problems understanding what the purpose of the application is, she/he has to pick answer #4.

### **Question #6: After playing the tutorial, did you have the feeling that you can control the app without any problems?**

As described in question #5, the user can play a tutorial within the application. This question is an addition to question #2. Besides the explanation given in question #2, this question defines whether the users had problems controlling the application after playing the tutorial. The possible answers for users to give range from #1 to #4. Answer #1 is defined as "No, I still had many problems". Answer #4 is defined as "Yes, no problems at all". So if a user had or has many problems when controlling the application after playing the tutorial, she/he must pick the answer #1. If a user had or has some problems when controlling the application after playing the tutorial, she/he must pick the answer #2. If a user had or has minor problems when controlling the application after playing the tutorial, she/he must pick the answer #3. If a user had or has no problems when controlling the application after playing the tutorial, she/he must pick the answer #4.

### **Question #7: Does the app help you to keep an overview of your courses and events (Termine)?**

This question is asked to clarify if users think the application is a useful addition to the MOOC platform to keep an overview of their courses and events. Furthermore, it should describe whether users think it is handy how the courses and events are listed within the application and whether the possibility of filtering is useful. The possible answers for users to give range from #1 to #4. Answer #1 is defined as "No, I can't keep an overview". Answer #4 is defined as "Yes, I have the perfect overview". So if a user has no appropriate overview of her/his courses and events at all, she/he has to pick the answer #1. If a user has a little appropriate overview of her/his courses and Events, she/he has to pick the answer #2. If a user has an appropriate overview of her/his courses and Events, she/he has to pick the answer #3. If a user has an optimal overview of her/his courses and Events, she/he has to pick the answer #4.

### **Question #8: Does the app increase your learning progress?**

This question is one of the most important indicators to verify whether the application is successful. This question is asked to clarify whether users think that using the application helped them increase their learning progress regarding the events and courses of the MOOC platform. One of the main goals of this application was to indicate whether an application in addition to the MOOC platform can raise the learning progress of the users of the MOOC platform. The other questions are asked to clarify whether the basic requirements for positive feedback to this question are given and that a positive user experience (design, controls, navigation) is provided. The possible answers for users to give range from #1 to #4. Answer #1 is defined as "No, my learning progress didn't increase at all". Answer #4 is defined as "Yes, my learning progress increased". So if a user had no increase in learning progress at all regarding the MOOC Courses, she/he has to pick answer #1. If a user had a little increase in learning progress regarding the MOOC Courses, she/he has to pick answer #2. If a user had some increase in learning progress regarding the MOOC Courses, she/he has to pick

## 5. Proof of concept

answer #3. If a user had a significant increase in learning progress regarding the MOOC Courses, she/he has to pick answer #4.

### **Question #9: What did you like best about the app? (Optional)**

This question is the first of two questions where users can give feedback with free text and it is optional. This question is asked to clarify whether any features stand out and what features the users like the most.

### **Question #10: What did you like least about the app? (Optional)**

This question is the second of two questions where users can give feedback with free text and it is optional. This question is asked to clarify whether any features were missing and what features the users liked the least.

This section sums up the questions of the survey and explains it. Eight of ten questions were single choice and mandatory. Two of the ten questions were free to answer and optional. In general, these questions clarify whether acceptable usability, design, navigation and controls of the application is given. Furthermore, question #8 "Does the app increase your learning progress?" defines whether the application was a success and therefore proves the concept.

## **5.2.2. The evaluation**

This section describes the evaluation of the survey provided and explained in subsection 5.2.1. The methodology followed is the detailed explanation of the answers given for each question. The answers for the free to answer questions are interpreted independently. Furthermore, after the analysis of the survey, an overall status of the application is given and whether the proof of concept is positive. In sum, 30 users of the applications in Android and iOS took part in the survey. 30 users answered the first eight questions, which were single choice questions and an answer was mandatory. The ninth and tenth question were both optional and therefore not all users have given feedback for these questions. For single choice questions, a total



## 5.2. Survey & Evaluation

of four possible answers was provided to the user. One of the answers is negative, one of the answers is rather negative, one of the answers is rather positive and one of the answers is positive. For the sake of simplicity and to evaluate whether a question was answered positively or negatively, the two answers rated as negative are joined together and the two answers rated as positive are joined together. A question is marked with positive feedback if two out of three users have given a positive answer. So, if more than 66,6% of the users have provided positive feedback to one specific question, it is a positive indicator for the proof of concept. The questions are not explained in this section. For a detailed explanation about the questions and the survey look up subsection 5.2.1.

### **Question #1: Do you like the design/appearance of the app?**

This question has the possible range of answers from #1 "No, I don't like it at all" til #4 "Yes, I totally like it". No user (0%) has answered this question with answer #1 (negative). Four users (~13,3%) have answered this question with answer #2 (rather negative). 12 users (40%) have answered this question with answer #3 (rather positive). 14 users (~46,7%) have answered this question with answer #4 (positive). This means, that four users (~13,3%) have given negative feedback regarding this question. 26 users (~86,7%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users are satisfied with the design/appearance of the application. Therefore, this indicator for the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.1

### **Question #2: Do or did you have problems controlling the app?**

This question has the possible range of answers from #1 "Yes, I had many problems" til #4 "No, no problems at all". One user (~3,3%) has answered this question with answer #1 (negative). Four users (~13,3%) have answered this question with answer #2 (rather negative). Eight users (~26,7%) have answered this question with answer #3 (rather positive). 17 users (~56,7%) have answered this question with answer #4 (positive). This means, that five users (~16,6%) have given negative feedback regarding this question.

## 5. Proof of concept

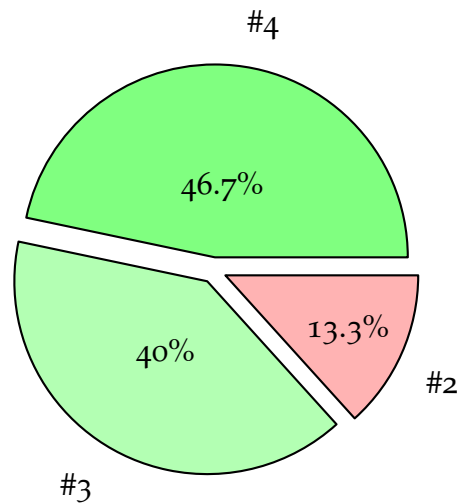


Figure 5.1.: Pie chart of answers for "Do you like the design/appearance of the app?" #1: "No, I don't like it at all", #4: "Yes, I totally like it"

25 users (~83,4%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users are satisfied with the controls of the application. Therefore, this indicator for the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.2

### Question #3: Do you think the navigation of the app is confusing?

This question has the possible range of answers from #1 "Yes, I don't know how to handle it" til #4 "No, the app is self-explaining". No user (0%) has answered this question with answer #1 (negative). Three users (~10%) have answered this question with answer #2 (rather negative). 14 users (~46,7%) have answered this question with answer #3 (rather positive). 13 users (~43,3%) have answered this question with answer #4 (positive). This means, that three users (~10%) have given negative feedback regarding this question. 27 users (~90%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users are satisfied with the navigation of the application. Therefore, this indicator for

## 5.2. Survey & Evaluation

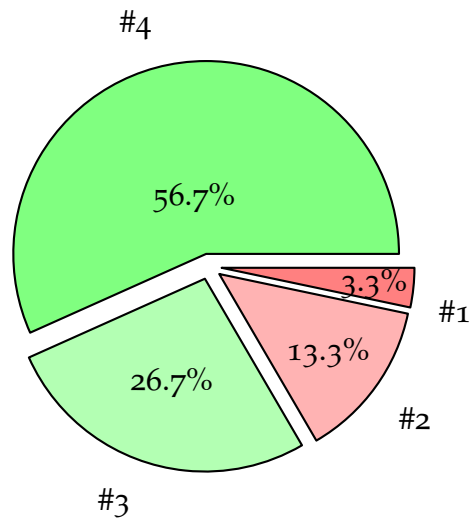


Figure 5.2.: Pie chart of answers for "Do or did you have problems controlling the app?"  
#1: "Yes, I had many problems", #4: "No, no problems at all"

the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.3.

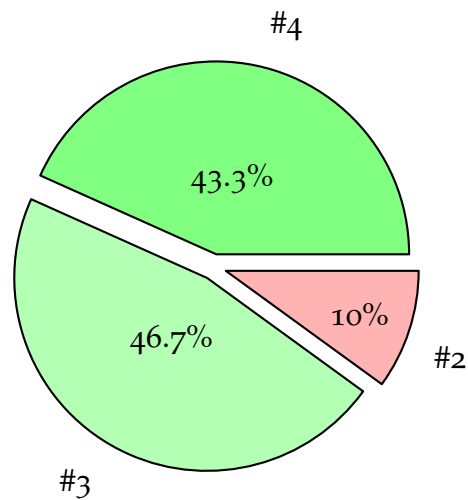


Figure 5.3.: Pie chart of answers for "Do you think the navigation of the app is confusing?"  
#1: "Yes, I don't know how to handle it", #4: "No, the app is self-explaining"

## 5. Proof of concept

### Question #4: Do you use the app at least once a week in addition to an iMooX-Course?

This question has the possible range of answers from #1 "No, never" til #4 "Yes, always". Two users (~6,7%) have answered this question with answer #1 (negative). 14 users (~46,7%) have answered this question with answer #2 (rather negative). Seven users (~23,3%) have answered this question with answer #3 (rather positive). Seven users (~23,3%) have answered this question with answer #4 (positive). This means, that 16 users (~53,4%) have given negative feedback regarding this question. 14 users (~46,6%) have given positive feedback regarding this question. This leads to the result, that about 53,3% of the users are not using the application at least once a week in addition to an iMooX-course. Therefore, this indicator for the proof of concept is negative. A visualised version of the given answers is delivered in Figure 5.4.

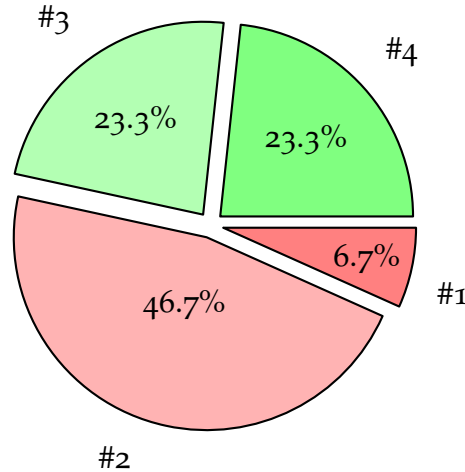


Figure 5.4.: Pie chart of answers for "Do you use the app at least once a week in addition to an iMooX-Course?" #1: "No, never", #4: "Yes, always"

**Question #5: After playing the tutorial, did you have the feeling you understand the app?**

This question has the possible range of answers from #1 "No, I didn't have a clue" til #4 "Yes, I understood it immediately". One user (~3,3%) has answered this question with answer #1 (negative). Two users (~6,7%) have answered this question with answer #2 (rather negative). 13 users (~43,3%) have answered this question with answer #3 (rather positive). 14 users (~46,7%) have answered this question with answer #4 (positive). This means, that three users (~10%) have given negative feedback regarding this question. 27 users (~90%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users were able to understand the application after playing the tutorial. Therefore, this indicator for the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.5.

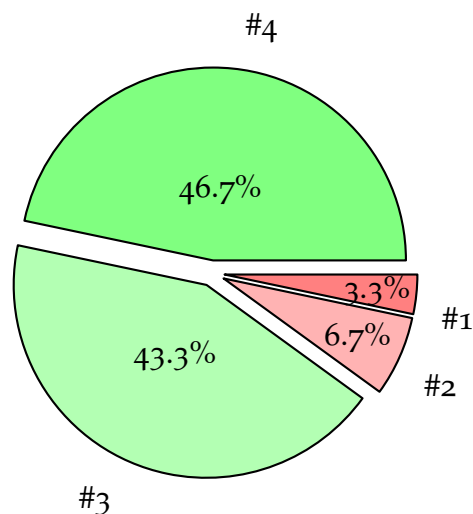


Figure 5.5.: Pie chart of answers for "After playing the tutorial, did you have the feeling you understand the app?" #1: "No, I didn't have a clue", #4: "Yes, I understood it immediately"

## 5. Proof of concept

### Question #6: After playing the tutorial, did you have the feeling that you can control the app without any problems?

This question has the possible range of answers from #1 "No, I didn't have a clue" til #4 "Yes, I understood it immediately". No user (0%) has answered this question with answer #1 (negative). Two users (~6,7%) have answered this question with answer #2 (rather negative). 13 users (~43,3%) have answered this question with answer #3 (rather positive). 15 users (50%) have answered this question with answer #4 (positive). This means, that two users (~6,7%) have given negative feedback regarding this question. 27 users (~93,3%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users were able to control the application after playing the tutorial. Therefore, this indicator for the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.6.

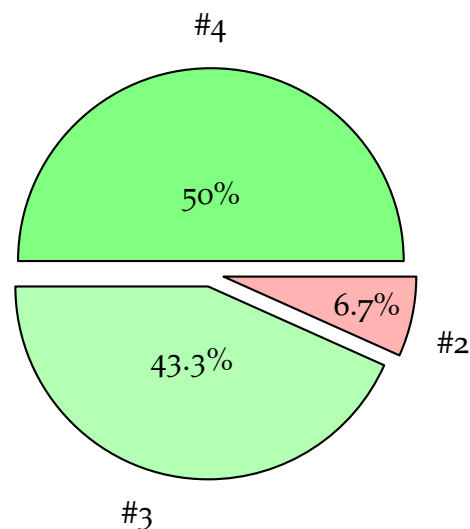


Figure 5.6.: Pie chart of answers for "After playing the tutorial, did you have the feeling that you can control the app without any problems?" #1: "No, I still had many problems", #4: "Yes, no problems at all"

**Question #7: Does the app help you to keep an overview of your courses and events (Termine)?**

This question has the possible range of answers from #1 "No, I didn't have a clue" til #4 "Yes, I understood it immediately". One user (~3,3%) has answered this question with answer #1 (negative). five users (~16,7%) have answered this question with answer #2 (rather negative). Eleven users (~36,7%) have answered this question with answer #3 (rather positive). 13 users (~43,3%) have answered this question with answer #4 (positive). This means, that six users (~20%) have given negative feedback regarding this question. 24 users (~80%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users can keep an overview of their courses and notes with the aid of the application. Therefore, this indicator for the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.7.

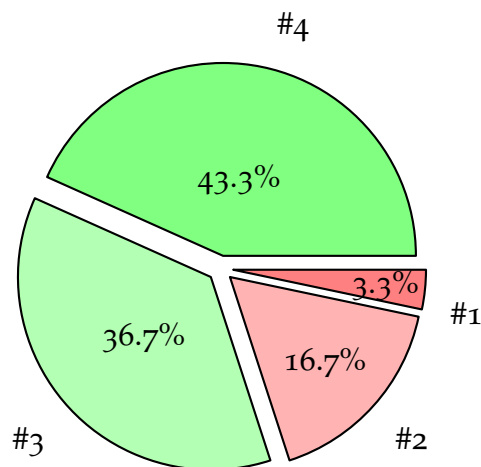


Figure 5.7.: Pie chart of answers for "Does the app help you to keep an overview of your courses and events (Termine)?" #1: "No, I can't keep an overview", #4: "Yes, I have the perfect overview"

## 5. Proof of concept

### Question #8: Does the app increase your learning progress?

This question has the possible range of answers from #1 "No, I didn't have a clue" til #4 "Yes, I understood it immediately". Four users (~13,3%) have answered this question with answer #1 (negative). Five users (~16,7%) have answered this question with answer #2 (rather negative). Nine users (~30%) have answered this question with answer #3 (rather positive). 12 users (~40%) have answered this question with answer #4 (positive). This means, that nine users (~30%) have given negative feedback regarding this question. 21 users (~70%) have given positive feedback regarding this question. This leads to the result, that more than 66,6% of the users have increased their learning progress with the aid of the application. Therefore, this indicator for the proof of concept is positive. A visualised version of the given answers is delivered in Figure 5.8.

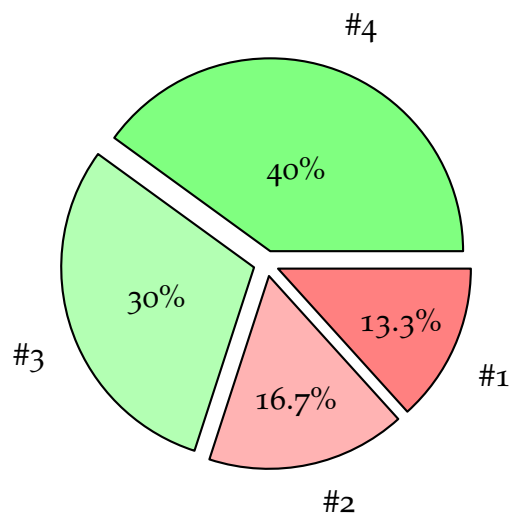


Figure 5.8.: Pie chart of answers for "Does the app increase your learning progress?"  
#1: "(No, my learning progress didn't increase at all", #4: "Yes, my learning progress increased"



### **Question #9: What did you like best about the app? (Optional)**

With this optional question, users could give positive feedback regarding the mobile application. Due to the fact, that not all 30 participants have answered this question and that it is not possible to find a meaningful indicator for the proof of concept, this question is without evaluation. In total, six users (~19,8%) have provided feedback to this question. The following lines are adopted without modification.

- *Overview*
- *The overview over the courses and timetables*
- *That I did not have to go through the online forms everytime*
- *The overview of all courses, the progress.*
- *The design of the application is very nice looking.*
- *The possibility to set a reminder.*

### **Question #10: What did you like least about the app? (Optional)**

With this optional question, users could give negative feedback regarding the mobile application. Due to the fact, that not all 30 participants have answered this question and that it is not possible to find a meaningful indicator for the proof of concept, this question is without evaluation. In total, five users (~16,5%) have provided feedback to this question. The following lines are adopted without modification.

- *The fact that I can't control my courses (participation)*
- *I'm taking courses that are finished and it's complicated to use the app to reach 'completed' lessons*
- *The application takes a long time to load, the courses appear timely.*
- *It would be nice to register for courses in the app.*

## 5. Proof of concept

- *No course registration.*

Seven out of eight questions collected positive feedback. This means, that 87,5% of the indicators have a positive evaluation. Because only one question has not been answered positively (~53,3% negative answers), the proof of concept is considered as positive. Therefore, the application works and the users are satisfied with the results of the prototype and development process. Furthermore, the most important indicator, the learning progress, was also positive. ~70% of the users have increased their learning progress with the aid of the application. The following hypothesis proves right: Using the application in addition to the MOOC platform leads to an increase in the learning progress with seven out of ten users.

## 6. Conclusion

This chapter reprocesses the prototype, the development of the application, the evaluation of the application and the thesis itself. During the prototype process and the development process, problems occurred that lead to a temporal blocker and needed more implementation time than expected. These problems are described in the section for lessons learned. These lessons learned should help future work to avoid mistakes that already have been made. In the section 6.2, current limitations for the application and the thesis are listed. Afterwards, features are explained that could lead to an improvement of the user experience if these limitations are eliminated. Finally, a summarising about the thesis and an outlook about further research is provided.

### 6.1. Lessons learned

During the whole process from creating a prototype up to publishing the applications, problems occurred that could have been avoided. Therefore, this section provides a listing of problems that occurred during this thesis, why they were a problem and how successors can prevent these problems from happening.

#### **The prototype**

During the definition of the prototype, some problems occurred. The problems that happened during the process were not critical, but another approach could have saved time. The first mistake that happened was, that the Mockup was created before all ideas and use cases have been written down.

## 6. Conclusion

There was an initial kick-off meeting about the main idea and the features that should get implemented, but the mistake was, that it was not thought through how the features get implemented. Before creating the mockup, there existed a set of features, but there were no use cases written down how the features should get implemented. Some of these use cases could have been:

- As a user, I want to see a list with all events and learn goals.
- As a user, I want the application to automatically scroll to the most current event or learn goal.
- As a user, I want to have a profile section where I have the option for colouring my courses, events, learn goals and notes.
- As a user, I want to create events.
- As a user, I want to create learn goals.
- As a user, I want to create notes.
- As a user, I want to define which type of event I create.
- As a user, I want to have a screen that displays all events, learn goals and notes related to one specific course.
- As a user, I want the application to automatically select the type and fill out the course when creating an event, a learn goal or a note when heading from the detail view of a course.

These are only a few of many use cases that could have been written down for the prototype process. Furthermore, several use cases could have been assigned to one feature to have an overview. One feature is completed, if all of these use cases are considered. Collecting use cases would have saved time. The way it was, not all parties involved were fully clear about the features and what they should do. Therefore the mockup was created multiple times until the final mockup was settled. Another aspect in terms of the mockup is to use a tool that allows you to modify one mockup all the time. Some of the tools for creating a mockup are pay-to-use. For this

thesis, a tool was used that was free. Therefore, after creating a mockup, there were limitations in editing the mockup. The lesson learned is, when creating a prototype, the idea should be defined, the features should get defined, use cases should get defined and a tool should be used where the mockup can be edited after the creation.

### **Application - Android & iOS**

The problem that occurred during the implementation of the Android version is regarding the structure of the project. Before starting to implement any code, a structure should be defined. A rule for coding. If no structure is applied, the project gets messed up and looking for code gets difficult. A clear structure where ViewModels, ItemViewModels, Views, Fragments, Activities, ViewControllers, Images and Helper Functions belong to is advisable. Therefore, before even writing any code, a structure with the help of packages/folders within the project is helpful. Furthermore, a mockup about the behaviour of the application could help to avoid spending unnecessary time about thinking where to put what. This could be a handwritten paper. The first step could be drawing each screen. The next step would be defining which type of screen it will be, for example, a Fragment, Activity, UIViewController, UITableViewController, etc. The third step would be to define which screen holds which ViewModel and/or ItemViewModel and which logic is located in the ViewModel or in the Views. Models can also be applied and defined. With this approach, during the development process, the developer does not have to think through the process and how to handle what. Also, the contradictions between code and Views can be avoided. With the help of writing down the structure, the developer can check which function is needed where and how to generalise functions if needed. For example, it is necessary for a screen to show an alert to the user. Therefore, the developer can plan a Helper Function that can be accessed from any class and shows an alert to the user. The developer can economise and centralise the code. It is not necessary to draw each screen in detail since the look-a-like of the different screens has already been defined during the prototype process. In this part of development, it is just about the logic of all elements. This approach was not used during the development of the

## 6. Conclusion

applications in this thesis and therefore more time was needed to arrange the project and the code.

### **Survey & Evaluation**

The problem of the enforcement of the survey was the lack of participation. Only 30 users have given feedback to the master thesis, even though the URL to the survey was provided in the application and the users get notifications after ten days after the installation of the application. Enough time was spent on the survey to define which questions are asked and how they can lead to an evaluation of the hypothesis that a learning diary in addition to a MOOC platform can increase the learning progress. The problem was the propagation and that the users did not earn anything for giving feedback. An approach would be, to leave it as it is but to reward users for taking the survey. For example, a user can immediately achieve the next level by completing the survey. This could lead to the fact that users are more motivated for giving feedback since they can earn something from it. Another approach would be to implement the feedback within the application itself and not in an external tool. This way, the application could force the users to give feedback, otherwise, they could not further use the application. This would be a suboptimal approach, since forcing users to give feedback and blocking the application from work is not in the interest of the user. It is still possible to use this idea and bring it to the application by not forcing the user to make a feedback. Another idea would be to use the native implemented survey and reward the users and to see how other users have voted for all the questions. This could arise the curiosity for doing the survey. For this approach, an external centralised database is necessary to store the data given for each survey. To sum up this section, spending time on the survey is essential.

## **6.2. Limitations & future work**

This section describes the limitations that were given for the applications. Furthermore, this section describes what would be meaningful next steps

## 6.2. Limitations & future work

to accomplish when improving the applications. The main restriction/limitation is the lack of backend. The applications itself only works with GET-Requests. This means, that only information can get retrieved from the backend of the MOOC platform. Due to security issues and lack of resources, it was not possible to arrange the backend in a way that the application can store information. When accomplishing this limitation, some improvements to the application are possible.

### **Courses**

The first feature that should get implemented in further work is the ability for users to register for courses and manage them within the application. When doing the survey, users have mentioned that they miss the ability for course registration. For this to accomplish, a POST API call is needed, so the application can send the required information to the backend and register the user for a course. Furthermore, it should be able for users to unsubscribe from a course if necessary. This feature could also be accomplished with a different POST API call.

### **Gamification**

The aspect of gamification could be improved in three ways. The first one would be to store the information, which currently are only stored locally, in a centralised external database. This would lead to the advantage, that users are device-independent. Right now, the user could use two different devices with the same credentials and the level and experience points would be different on both devices. This is suboptimal. When storing the information in a centralised external database, the level and experience points would be the same for all devices the user is using. Furthermore, when deleting the application, this would not lead to a loss of the gamification progress. The second aspect to improve would be the gamification itself. As described in section 3.2, there was the idea to provide the user with the possibility to design an avatar her-/himself. With the increase of the level, the user would have more possibilities to design her/his avatar. The third way of improving the gamification is in addition to the other two. With an external

## 6. Conclusion

database, the application could provide a leaderboard. In this leaderboard, the users could see the rankings, levels, experience points and avatar of all other users. The advantage created is that users have a comparison to other users and the motivation to improve the level could increase.

### **Login**

As described in chapter 4, in both versions, Android and iOS, the login screen is handled with a Web View. The applications use Web Views because there could arise security issues when a user is logging in from an external platform other than the MOOC platform. This is a limitation to target, since providing a Web View could lead to a bad performance in the applications. A native built login screen would be faster than loading the login screen in a WebView. Furthermore, the design could be optimised and adapted to the needs of a mobile application on a smartphone. For this to happen, the application needs an API endpoint that can handle user registration/login and responses with an authorisation token.

### **Offline usage**

As mentioned in chapter 5, offline usage is one feature of the application which was not implemented to sufficiency. When using the application offline, the application is not used to full potential. For implementing offline usage, a structure and an approach should be defined. Every time the application gets started and a stable internet connection is given, the application should retrieve all information of the courses and events from the MOOC platform and store them locally, so the user can use the courses and events when the application has no internet connection.

### **Forum**

One idea that has not been mentioned yet in this thesis is the feature of a forum. The forum could be implemented in two ways. The first one would be to use the forum that is already present at the MOOC platform with a



### 6.3. Summary & Outlook

Web View. Even though this would be a faster solution than the second one, this would lead to a performance reduction. The second way would be to implement the forum with a backend and build it natively in the application. This means, that the application would need API calls for retrieving, posting and updating posts within the forum. This way, the users also could access the forum within the application and do not have to move to the web application of the MOOC platform.

To sum up this section, the main limitation is that it is currently not possible for the application to post data to a backend. Fixing this limitation would lead to a variety of potential useful features like course registration, course management and improvement of gamification. Furthermore, the application is currently working as an addition to the MOOC platform. To provide the optimal user experience, the applications should not be an addition to the MOOC platform but work as a self-responsible platform. This means, the optimum would be that the applications provide the same core features as the MOOC platform does.

### 6.3. Summary & Outlook

Learning diaries, also known as learning journals, can be established in different types and forms. They appear in written and digital forms and their usage and advantages are discussed in papers and researches. As described in section 2.1, a learning diary is a tool that helps students to improve the learning process and learning progress. In section 2.3, some applications are listed that already successfully provide the feature of a learning diary. Gamification is a word that is quite common in the digital industry, even though it does not have to be connected to educational areas. As described in section 2.2, gamification is the application of game elements to non-game contexts. The prototype described in chapter 3 tried to break down a possible learning diary related to a MOOC platform to its necessary core features. Ideas and how the design of the features could look like are explained. The development of the application described

## 6. Conclusion

in chapter 4 explains both versions in a way, that developer with know-how in mobile application development could rebuild both applications. It also described how features have been implemented for both operating systems, even though they were consciously built similar. The evaluation provided in chapter 5 shows, that the hypothesis of this master thesis proves right. The outlook for this application is promising. One main feature for improving the outlook of this application and the master thesis itself is gamification. In this master thesis, to use the word gamification is correct, since a learning diary is a non-game context. Applying gamification to it results in a non-game context educational area in combination with gaming-elements. This is a combination that holds potential. The learning diary itself can still be improved as discussed in section 6.2. Applying these features would improve the overall user experience of the application and respect the feedback that was given by the users described in subsection 5.2.2. Especially the part of gamification can be improved immensely. Due to the resources given by a master thesis, it was not able to maximise the potential of gamification in this context. Applying more game-like elements like avatars, currencies and awards would result in a push of motivation of the users. Furthermore, implementing an overall leaderboard could result in a competition that pushes the users even further to accomplish their tasks in time. Improving the design, controls, navigation and gamification of the application would result in more sufficiency of the users. This could result in the main thing this master thesis is all about: The learning progress. The learning progress was the main reason for this master thesis. Even though the results are satisfying, the application still can be improved. The goal is to make more students use the application. More students would result in more competition with a leaderboard. This could result in an improvement of the learning progress. The design, navigation, controls and the gamification itself are elements that are necessary to increase the overall learning progress of the users. The outlook for further work would be the well thought out enhancement of each of these elements. In general, this master thesis proved its hypothesis and the applications which resulted from the prototype are established in Apples App Store and Google Play Store. The applications and the background itself hold potential for the future. Respecting the features provided and the suggestions for further work could lead to an improvement in the overall application experience and the learning progress of the users of the MOOC platform.

# Appendix



## **Appendix A.**

### **Survey of applications**

	Do you like the design/appearance of the app?	Do or did you have problems controlling the app?
User 1	3	4 (No, no problems at all)
User 2	4 (Yes, I totally like it)	4 (No, no problems at all)
User 3	4 (Yes, I totally like it)	4 (No, no problems at all)
User 4	4 (Yes, I totally like it)	4 (No, no problems at all)
User 5	3	3
User 6	4 (Yes, I totally like it)	4 (No, no problems at all)
User 7	3	2
User 8	4 (Yes, I totally like it)	4 (No, no problems at all)
User 9	2	3
User 10	4 (Yes, I totally like it)	4 (No, no problems at all)
User 11	4 (Yes, I totally like it)	4 (No, no problems at all)
User 12	4 (Yes, I totally like it)	4 (No, no problems at all)
User 13	3	4 (No, no problems at all)
User 14	2	2
User 15	4 (Yes, I totally like it)	1 (Yes, I had many problems)
User 16	3	3
User 17	3	4 (No, no problems at all)
User 18	4 (Yes, I totally like it)	3
User 19	4 (Yes, I totally like it)	4 (No, no problems at all)
User 20	3	3
User 21	2	3
User 22	3	2
User 23	3	3
User 24	2	3
User 25	3	2
User 26	3	4 (No, no problems at all)
User 27	3	4 (No, no problems at all)
User 28	4 (Yes, I totally like it)	4 (No, no problems at all)
User 29	4 (Yes, I totally like it)	4 (No, no problems at all)
User 30	4 (Yes, I totally like it)	4 (No, no problems at all)

	Do you think the navigation of the app is confusing?	Do you use the app at least once a week in addition to an iMooX-Course?
User 1	3	4 (Yes, always)
User 2	4 (No, the app is self-explaining)	2
User 3	4 (No, the app is self-explaining)	2
User 4	4 (No, the app is self-explaining)	4 (Yes, always)
User 5	4 (No, the app is self-explaining)	2
User 6	4 (No, the app is self-explaining)	2
User 7	3	2
User 8	4 (No, the app is self-explaining)	3
User 9	2	1 (No, never)
User 10	3	3
User 11	3	3
User 12	3	3
User 13	3	4 (Yes, always)
User 14	4 (No, the app is self-explaining)	4 (Yes, always)
User 15	3	2
User 16	2	1 (No, never)
User 17	3	4 (Yes, always)
User 18	3	3
User 19	4 (No, the app is self-explaining)	2
User 20	4 (No, the app is self-explaining)	2
User 21	3	2
User 22	3	3
User 23	3	2
User 24	3	2
User 25	2	2
User 26	4 (No, the app is self-explaining)	2
User 27	3	4 (Yes, always)
User 28	4 (No, the app is self-explaining)	2
User 29	4 (No, the app is self-explaining)	4 (Yes, always)
User 30	4 (No, the app is self-explaining)	3

	<b>After playing the tutorial, did you have the feeling you understand the app?</b>	<b>After playing the tutorial, did you have the feeling that you can control the app without problems?</b>
<b>User 1</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 2</b>	4 (Yes, I understood it immediately)	3
<b>User 3</b>	4 (Yes, I understood it immediately)	3
<b>User 4</b>	3	4 (Yes, no problems at all)
<b>User 5</b>	3	3
<b>User 6</b>	1 (No, I didn't have a clue)	2
<b>User 7</b>	3	4 (Yes, no problems at all)
<b>User 8</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 9</b>	2	3
<b>User 10</b>	3	2
<b>User 11</b>	3	3
<b>User 12</b>	3	3
<b>User 13</b>	3	3
<b>User 14</b>	3	3
<b>User 15</b>	3	4 (Yes, no problems at all)
<b>User 16</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 17</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 18</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 19</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 20</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 21</b>	3	3
<b>User 22</b>	3	3
<b>User 23</b>	3	3
<b>User 24</b>	2	3
<b>User 25</b>	3	3
<b>User 26</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 27</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 28</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 29</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)
<b>User 30</b>	4 (Yes, I understood it immediately)	4 (Yes, no problems at all)



	<b>Does the app help you to keep an overview of your courses and events (Termine) ?</b>	<b>Does the app increase your learning progress?</b>
<b>User 1</b>	4 (Yes, I have the perfect overview)	1 (No, my learning progress didn't increase at all)
<b>User 2</b>	3	4 (Yes, my learning progress increased)
<b>User 3</b>	3	4 (Yes, my learning progress increased)
<b>User 4</b>	3	4 (Yes, my learning progress increased)
<b>User 5</b>	4 (Yes, I have the perfect overview)	3
<b>User 6</b>	2	3
<b>User 7</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)
<b>User 8</b>	4 (Yes, I have the perfect overview)	3
<b>User 9</b>	1 (No, I can't keep an overview)	1 (No, my learning progress didn't increase at all)
<b>User 10</b>	2	2
<b>User 11</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)
<b>User 12</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)
<b>User 13</b>	4 (Yes, I have the perfect overview)	3
<b>User 14</b>	4 (Yes, I have the perfect overview)	2
<b>User 15</b>	2	4 (Yes, my learning progress increased)
<b>User 16</b>	4 (Yes, I have the perfect overview)	3
<b>User 17</b>	3	4 (Yes, my learning progress increased)
<b>User 18</b>	3	3
<b>User 19</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)
<b>User 20</b>	3	1 (No, my learning progress didn't increase at all)
<b>User 21</b>	3	2
<b>User 22</b>	3	3
<b>User 23</b>	2	2
<b>User 24</b>	3	1 (No, my learning progress didn't increase at all)
<b>User 25</b>	2	3
<b>User 26</b>	3	3
<b>User 27</b>	3	2
<b>User 28</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)
<b>User 29</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)
<b>User 30</b>	4 (Yes, I have the perfect overview)	4 (Yes, my learning progress increased)

	What did you like best about the app?	What did you like least about the app?
<b>User 8</b>	Overview	-
<b>User 14</b>	The overview over the courses and timetables	The fact that I can't control my courses (participation)
<b>User 15</b>	That I did not have to go through the online forms everytime	I'm taking courses that are finished and it's complicated to use the app to reach 'completed' lessons
<b>User 26</b>	The overview of all courses, the progress.	The application takes a long time to load, the courses appear timely.
<b>User 29</b>	The design of the application is very nice looking.	It would be nice to register for courses in the app.
<b>User 30</b>	The possibility to set a reminder.	No course registration.

# Bibliography

- Activity* (2019). URL: <https://developer.android.com/reference/android/app/Activity> (visited on 04/15/2019) (cit. on p. 47).
- Android 5.1 APIs* (2019). URL: <https://developer.android.com/about/versions/android-5.1> (visited on 04/15/2019) (cit. on p. 46).
- Android Studio* (2019). URL: [https://en.wikipedia.org/wiki/Android\\_Studio](https://en.wikipedia.org/wiki/Android_Studio) (visited on 04/25/2019) (cit. on p. 50).
- Burke, Brian (2014). *Gamify - How Gamification Motivates People to do Extraordinary Things*. Gartner, Inc (cit. on p. 7).
- Cazan, Ana-Maria (2012). "Enhancing self regulated learning by learning journals." In: *Procedia - Social and Behavioral Sciences* 33. PSIWORLD 2011, pp. 413-417. ISSN: 1877-0428. DOI: <https://doi.org/10.1016/j.sbspro.2012.01.154>. URL: <http://www.sciencedirect.com/science/article/pii/S1877042812001620> (cit. on p. 1).
- Clipa, Otilia, Aurora-Adina Ignat, and Mihai Stanciu (2012). "Learning diary as a tool for metacognitive strategies development." In: *Procedia - Social and Behavioral Sciences* 33. PSIWORLD 2011, pp. 905-909. ISSN: 1877-0428. DOI: <https://doi.org/10.1016/j.sbspro.2012.01.253>. URL: <http://www.sciencedirect.com/science/article/pii/S1877042812002613> (cit. on p. 1).
- Dowling-Feet, Michael (2015). *My Interactive Learning kit "MILK", the student work diary app*. URL: <https://www.rm.com/blog/2015/june/my-interactive-learning-kit-milk-the-student-work-diary-app> (visited on 09/29/2019) (cit. on p. 13).
- Dowling-Fleet, Michael (2015). *My Interactive Learning Kit "MILK", the student work diary app*. URL: <http://rmunify.j2bloggy.com/RM-Unify-Blog/my-interactive-learning-kit-milk-the-student-work-diary-app/> (visited on 11/19/2019) (cit. on p. 14).
- Fragments* (2019). URL: <https://developer.android.com/guide/components/fragments> (visited on 04/15/2019) (cit. on p. 47).

## Bibliography

- Gamification* (2019). URL: <https://en.wikipedia.org/wiki/Gamification> (visited on 04/02/2019) (cit. on p. 26).
- Hsin-Yuan, Huang Wendy and Soman Dilip (2013). "Gamification of Education." In: Roman School of Management, University of Toronto (cit. on pp. 8–12).
- Integrated development environment* (2019). URL: [https://en.wikipedia.org/wiki/Integrated\\_development\\_environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (visited on 11/16/2019) (cit. on p. 44).
- Lerntagebuch* (2017). URL: <https://de.wikipedia.org/wiki/Lerntagebuch> (visited on 03/07/2019) (cit. on p. 5).
- Market share of mobile operating systems worldwide 2012-2019* (2019). URL: <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/> (visited on 10/02/2019) (cit. on pp. 1, 2).
- Material Design* (2019). URL: <https://material.io> (visited on 04/15/2019) (cit. on pp. 46, 52, 77).
- Metakognition* (2019). URL: <https://de.wikipedia.org/wiki/Metakognition> (visited on 10/15/2019) (cit. on p. 1).
- Migrating to Android 8.0* (2019). URL: <https://developer.android.com/about/versions/oreo/android-8.0-migration> (visited on 04/15/2019) (cit. on p. 46).
- Mockup* (2019). URL: <https://en.wikipedia.org/wiki/Mockup> (visited on 11/17/2019) (cit. on pp. 19, 28).
- Model-view-viewmodel* (2019). URL: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel> (visited on 11/17/2019) (cit. on p. 44).
- Nicholson, Scott (2015). "A RECIPE for Meaningful Gamification." In: *Gamification in Education and Business*. Ed. by Torsten Reiners and Lincoln C. Wood. Cham: Springer International Publishing, pp. 1–20. DOI: 10.1007/978-3-319-10208-5\_1. URL: [https://doi.org/10.1007/978-3-319-10208-5\\_1](https://doi.org/10.1007/978-3-319-10208-5_1) (cit. on p. 1).
- Petko, Dominik (2013). "Lerntagebuch schreiben mit Weblogs. Didaktische Grundlagen und technische Entwicklungen am Beispiel von [lerntagebuch.ch](http://lerntagebuch.ch)." In: *E-Portfolio an der Schnittstelle von Studium und Beruf*. 48046 Münster: Waxmann Verlag GmbH, pp. 206–214. ISBN: 978-3-8309-2818-8 (cit. on pp. 6, 7).

- Projects overview* (2019). URL: <https://developer.android.com/studio/projects> (visited on 04/15/2019) (cit. on p. 47).
- Radio button* (2019). URL: [https://en.wikipedia.org/wiki/Radio\\_button](https://en.wikipedia.org/wiki/Radio_button) (visited on 04/08/2019) (cit. on p. 32).
- Realm Database* (2019). URL: <https://realm.io/products/realm-database> (visited on 04/15/2019) (cit. on p. 48).
- Retrofit* (2019). URL: <https://square.github.io/retrofit/> (visited on 04/15/2019) (cit. on p. 47).
- Seaborn, Katie and Deborah I. Fels (2015). "Gamification in theory and action: A survey." In: *International Journal of Human-Computer Studies* 74, pp. 14–31. ISSN: 1071-5819. DOI: <https://doi.org/10.1016/j.ijhcs.2014.09.006>. URL: <http://www.sciencedirect.com/science/article/pii/S1071581914001256> (cit. on pp. 1, 7).
- Seesaw (2016). "Seesaw for Schools efficacy study." In: Seesaw Learning, Inc. URL: <https://help.seesaw.me/hc/en-us/articles/115005752703-Seesaw-For-Schools-efficacy-study> (visited on 11/19/2019) (cit. on pp. 15, 16).
- Seesaw* (2019). URL: <https://web.seesaw.me> (visited on 10/05/2019) (cit. on p. 15).
- Share of Apple devices by iOS version worldwide from 2016 to 2018* (2018). URL: <https://www.statista.com/statistics/565270/apple-devices-ios-version-share-worldwide/> (visited on 07/08/2019) (cit. on p. 75).
- Söbke, Heinrich and Steffi Zander (2018). "Motivationsdesign durch Verschränkung von Gamifikation und didaktischem Kontext." In: *DeLFI 2018 - Die 16. E-Learning Fachtagung Informatik*. Ed. by Detlef Krömker and Ulrik Schroeder. Bonn: Gesellschaft für Informatik e.V., pp. 141–152 (cit. on p. 8).
- Software development kit* (2019). URL: [https://en.wikipedia.org/wiki/Software\\_development\\_kit](https://en.wikipedia.org/wiki/Software_development_kit) (visited on 11/16/2019) (cit. on p. 46).
- Spiel, Christiane et al. (2010). *Das Lerntagebuch in der Hochschullehre: Ein hochschuldidaktischer Ansatz zur Förderung selbstgesteuerten Lernens*. Hogrefe Verlag GmbH Co. KG. ISBN: 978-3-8017-2081-0 (cit. on p. 6).
- Switch* (2019). URL: <https://developer.android.com/reference/android/widget/Switch> (visited on 04/08/2019) (cit. on p. 33).
- What is Seesaw?* (2019). URL: <https://help.seesaw.me/hc/en-us/articles/115003713306-What-is-Seesaw-> (visited on 11/19/2019) (cit. on p. 17).