University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

MASTER THESIS

Pilsen, 2015

Jan Strejc

University of West Bohemia

Faculty of Applied Sciences

Department of Computer Science and Engineering

Master Thesis

Multi-platform mobile application for Scrum

Pilsen, 2015

Jan Strejc

Original requirements

Declaration

I hereby declare that this master thesis is completely my own work and that I used only the cited sources.

Pilsen 25th June 2015, Jan Strejc

Acknowledgment

I would like to thank to Manfred Nowotny and Mario Wessely from Austrian company OnTec for giving me the opportunity to work on this thesis. They were very helpful to me, when I was creating this application and needed any consultation. By working on this thesis I gained a lot of valuable experience.

Abstract

This work aims at project management using scrum on mobile devices. In this text I explore existing mobile applications for scrum, compare available solutions for multiplatform mobile development and describe possible ways of distributed mobile device synchronization without a dedicated server.

Next part is about implementation of a multiplatform mobile application for scrum, which is based on the above mentioned research. At the end I perform tests on the created application and evaluate the results.

Abstrakt

Tato práce se zaměřuje na projekt management s použitím scrumu na mobilních zařízeních. V tomto textu prozkoumávám existující mobilní aplikace pro práci se scrumem, porovnávám dostupná řešení pro multiplatformní mobilní vývoj a popisuji možné způsoby distribuované komunikace mezi mobilními zařízeními bez dedikovaného serveru.

Další část popisuje implementaci mobilní multiplatformní aplikace pro práci se scrumem, která vychází z výše uvedeného výzkumu. Na konci vytvořenou aplikaci testuji a zhodnotím výsledky.

Key words

Scrum, agile development, multi-platform, cross-platform, mobile application, Android, iOS, Windows, Cordova, SQLite, GWT, distributed communication, mDNS, zero-conf

Table of contents

1 Introduction **1**

2 Scrum **2**
2.1 Introduction to scrum . . . 2
2.2 Iterative development . . . 3
2.3 Incremental development . . . 3
2.4 Combining iterative and incremental development . . . 4
2.5 Comparison with plan driven development . . . 5
2.5.1 Just in time approach . . . 6
2.6 Scrum overview . . . 7
2.6.1 Scrum principles . . . 7
2.7 Scrum roles . . . 8
2.7.1 Product owner . . . 8
2.7.2 Scrum master . . . 9
2.7.3 Development team . . . 11
2.8 Scrum activities and artifacts . . . 12
2.8.1 Product backlog . . . 12
2.8.2 Sprints . . . 13
2.8.3 Sprint backlog . . . 14
2.8.4 Stories and tasks . . . 14
2.8.5 Sprint planning . . . 15
2.8.6 Sprint execution . . . 16
2.8.7 Daily scrum . . . 16
2.8.8 Estimation . . . 16
2.8.9 Velocity . . . 18
2.8.10 Sprint review . . . 18
2.8.11 Sprint retrospective . . . 18
2.8.12 Scrum board . . . 19
2.8.13 Burndown charts . . . 19

3 Multiplatform mobile development **22**
3.1 Mobile operating systems . . . 22
3.1.1 Apple iOS . . . 23
3.1.2 Google Android . . . 25
3.1.3 Microsoft Windows . . . 26
3.2 Native vs multiplatform development . . . 28
3.2.1 Native . . . 28
3.2.2 Multiplatform . . . 28
3.3 Multiplatform development frameworks . . . 29
3.3.1 Apache Cordova (PhoneGap) . . . 30
3.3.2 Qt . . . 31
3.3.3 Xamarin . . . 33

3.3.4 Appcelerator . . . 36
3.3.5 Sencha touch . . . 36
3.3.6 Online app builders . . . 37
3.3.7 Pure Webview application . . . 39
3.3.8 Games . . . 39
3.4 Conclusion . . . 40

4 Existing solutions **41**
4.1 Scrum poker applications . . . 41
4.2 JIRA Connect Enterprise . . . 42
4.3 AgileScrum Pro . . . 43
4.4 Lion Monkey Scrum . . . 44
4.5 Pivotal tracker . . . 45
4.6 Conclusion . . . 46

5 Communication **47**
5.1 SMS . . . 47
5.2 Communication over Internet . . . 47
5.2.1 Push notifications . . . 47
5.2.2 Email, File sharing services . . . 48
5.3 Local communication . . . 48
5.3.1 Bluetooth . . . 48
5.3.2 Wi-Fi . . . 49
5.4 Conclusion . . . 51

6 Data storage **52**
6.1 Files . . . 52
6.2 Key value storage . . . 52
6.3 SQLite database . . . 53
6.4 Conclusion . . . 54

7 ScrumApp specification **55**
7.1 Basic features . . . 55
7.2 Optional features . . . 56
7.3 Technologies . . . 56
7.4 Basic architecture overview . . . 57
7.5 Use cases . . . 58
7.6 Testing . . . 59

8 Implementation **60**
8.1 Multiplatform and native part . . . 60
8.2 Technology stack . . . 61

9 Database **62**
9.1 Database model . . . 62
9.2 Database tables . . . 63
9.2.1 Database update . . . 63
9.3 Data synchronization . . . 63
9.3.1 Distributed keys . . . 64

9.3.2 Database synchronization 64
9.3.3 Slave pulls master for updates 64
9.3.4 Slave pushing his changes to master 65
9.3.5 Slave asks master to update the database for him 65

10 Native android part **67**
10.1 Cordova library 68
10.2 SQLite Cordova plugin 69
10.3 JmDNS library 69
10.4 Database DAO 69
10.5 ScrumApp server 69
10.5.1 Background server service 70
10.5.2 TCPServer 71
10.5.3 Scrum server 71
10.6 ScrumApp Client 72
10.7 File import and export 72
10.7.1 File import 72
10.7.2 File export 73
10.8 Cordova plugins 74
10.8.1 Plugin principle 74
10.9 Multiplatform HTML, CSS and JavaScript code 76

11 Native iOS part **77**
11.1 Cordova library 77
11.2 SQLite plugin 78
11.3 ScrumApp Client 78
11.4 File import and export 78
11.5 Cordova plugins 79
11.5.1 Plugin principle 79
11.6 Multiplatform HTML, CSS and JavaScript code 80

12 Communication protocol **81**
12.1 Server method list 82
12.2 Message content examples 82
12.2.1 Project list 83
12.2.2 Vote 83
12.2.3 Sync issues 84

13 GWT multiplatform part **85**
13.1 GWT overview 85
13.1.1 Developing using GWT 85
13.1.2 JSNI 86
13.1.3 mGWT 87
13.1.4 GWT PhoneGap 87
13.2 ScrumApp GWT 88
13.2.1 Database plugin 88
13.2.2 Client 89
13.2.3 Connection 89

13.2.4 DAO 90
13.2.5 Files 91
13.2.6 JSNI 91
13.2.7 Resources 91
13.2.8 UI 91
13.2.9 Util 91
13.2.10 Activities 91
13.2.11 Pages 92

14 Tests **96**

15 Results **97**
15.1 What is done 97
15.2 What isn't done 98
15.3 Encountered problems 99
15.4 Supported devices 100
15.5 Performance 100
15.6 Battery usage 101
15.7 Software used for development 101
15.8 Statistic 101

16 Conclusion **102**

1

Introduction

My goal in this work is to create a multiplatform mobile application for project management using scrum. I will start by analyzing existing mobile applications for project management using scrum. Then I will describe the scrum process, as scrum and agile development are generally very popular ways of developing software these days.

In present time, there is an urge to have a mobile app for almost everything. This creates a demand for rapid and cheap development of apps of all kind. Problem for fast deployment is incompatibility and big differences between major operating systems. Many mobile multiplatform development frameworks were created over the past years. I will describe current leading mobile platforms and compare the most popular mobile multiplatform development solutions.

For this mobile application I will explore possible ways of using distributed data synchronization between mobile devices without using a dedicated server. I will have to deal with several problems here, like creating some form of device discovery that allows two devices to find each other. Another important thing will be storing complex data on mobile devices, where traditional database servers are unavailable and synchronizing this data from one device to another.

I will analyze use cases for the application and create a specification with basic and optional features. Using one of the multiplatform development frameworks I will design and implement the multiplatform mobile application. The application will be deployed and tested on real devices with different operating systems.

2

Scrum

In this chapter I will describe the scrum process generally, because scrum does not force you to do things in only one way and can be easily customized. I will describe how I decided to create my implementation of scrum, suiting my use case, in the next chapter.

2.1 Introduction to scrum

Scrum (in software development) is an iterative and incremental agile product or service development framework. Key features are flexibility, self organization, collaboration and communication (in the development team itself and also with the customer). Scrum recognizes and orients itself on the fact that customers can and usually change their mind about what they want or need during the development process. Traditional ways of software development such as plan driven development have problems with reacting on sudden changes, because everything must be specified and planed at the start. Scrum instead of focusing on understanding and defining everything, focuses on fast delivery and ability to respond to changing requirements. Of course scrum is not the ultimate solution for everything, some projects can be handled better with more traditional ways (usually military or government projects). **[esc]**

The basic overview on scrum can be seen on figure 2.1. The principles of scrum will be described in the following pages. This chapter is mostly based on Essential Scrum **[esc]**, which is a book written by Kenneth S. Rubin that aggregates a lot information about the scrum framework and is based on many other famous books about scrum, agile development and product development process generally.

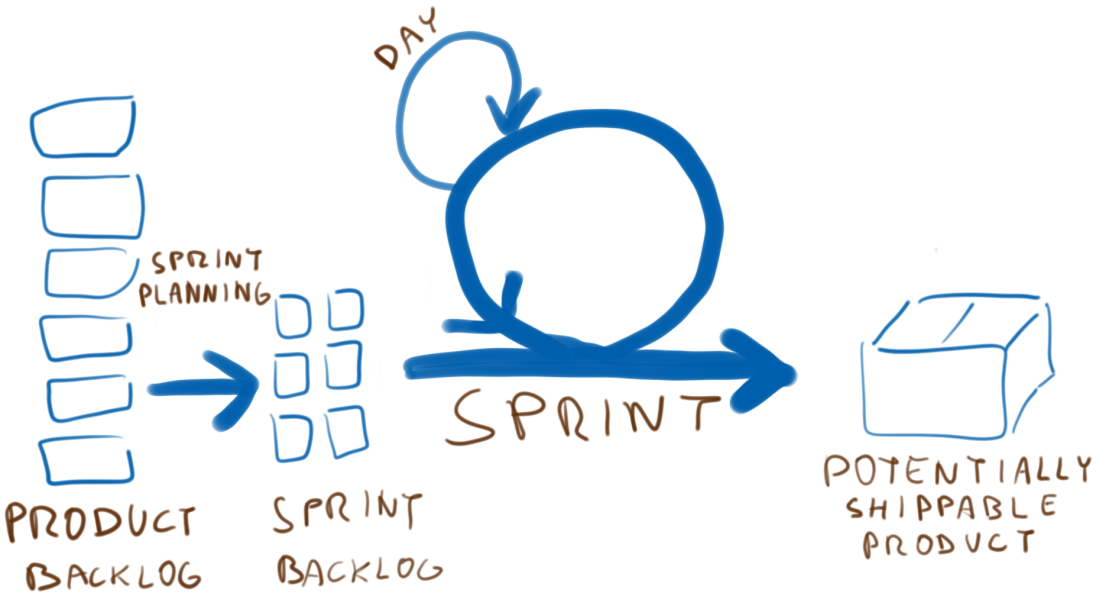


Figure 2.1: Scrum overview

2.2 Iterative development

Creating a product in iterative development is based on reworking the product over and over again, until we are satisfied with it. We start by creating a prototype and than we present it to the customer, users etc. Based on their feedback we improve the product and present it again. Each pass is called an iteration. We repeat iterations until we are finished. It's important to note, that it's very hard to predict at the start how many iterations we will need to complete the product. **[esc]**

2.3 Incremental development

Rather than building the project as one big thing, where all the pieces come together at the end, we try to split the project into smaller pieces and build them one by one and test how they work. This way we can incorporate feedback into future work. We can also find problems and learn from them sooner and apply our new knowledge on further development. This gives us the opportunity to adapt the way we continue the development. On the other hand we risk missing the big picture, because we focus only on the small pieces. **[esc]**

2.4 Combining iterative and incremental development

Scrum tries to take the best parts from both the incremental and iterative development and combine them together, while removing the drawbacks. In scrum we don't work on a phase at a time (design, implementation, testing ...), but we work on a feature at a time. To do so, scrum uses a set of iterations called sprints. In each iteration we build a working product increment by performing all the necessary steps:

- Analysis
- Design
- Implementation
- Integration
- Testing

This way, by doing all the feature related work in one sprint, we build a working increment which we can present to get feedback. Than we can adapt our future work. If the feedback on the finished feature wasn't so good, we can schedule it for future improvement (iterative development), or if it was accepted, we can than choose the next feature for the next sprint. We don't need to decide how many iterations (sprints) are needed, this will come naturally from the feedback and incremental way of work (figure 2.2). **[esc]**

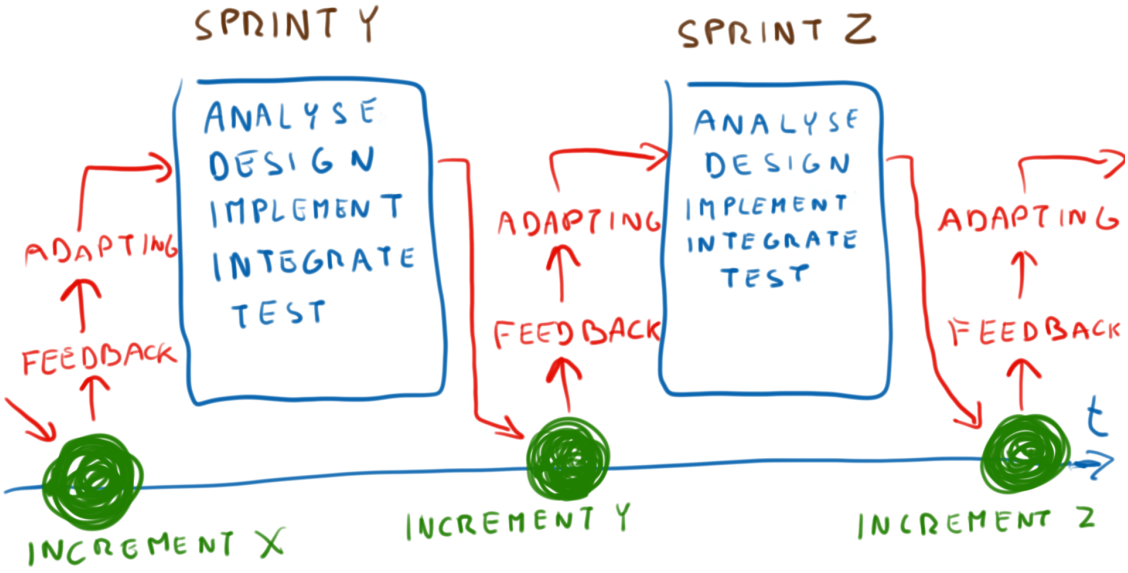


Figure 2.2: Iterative and incremental development

2.5 Comparison with plan driven development

In plan driven development we specify the requirements at the start of the product development and we let the customer review them immediately so we can advance to the next phase. The problem is, we make decisions before we acquire deeper knowledge about the product (see figure 2.3). This creates an illusion of having everything under control and can lead to creation of large amount of low quality requirements. **[esc]**

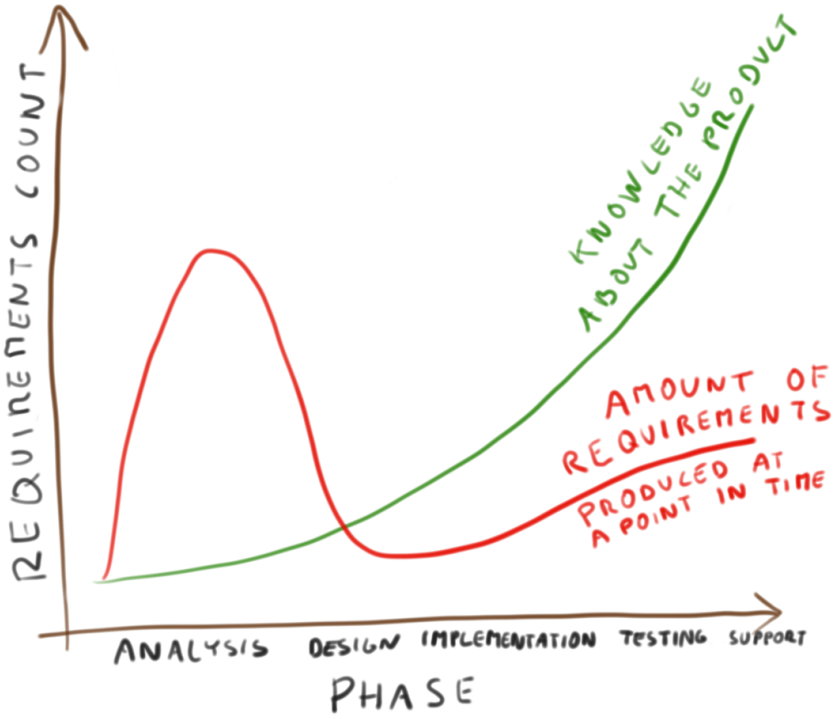


Figure 2.3: Requirements in plan driven development

On the other hand in scrum we think, that we should not make premature decisions, just because the process wants us to. In scrum we try to have our options open and thus we make decisions at the last responsible moment. This is a point when cost of not deciding would be greater than cost of making a decision (figure 2.4). We don't want to make critical and irreversible decisions until we have deeper knowledge about the product, so our decision can be better. If we make a wrong critical decision at the start of the product the cost of going back can be big, especially if we realize our mistake several phases later. **[esc]**

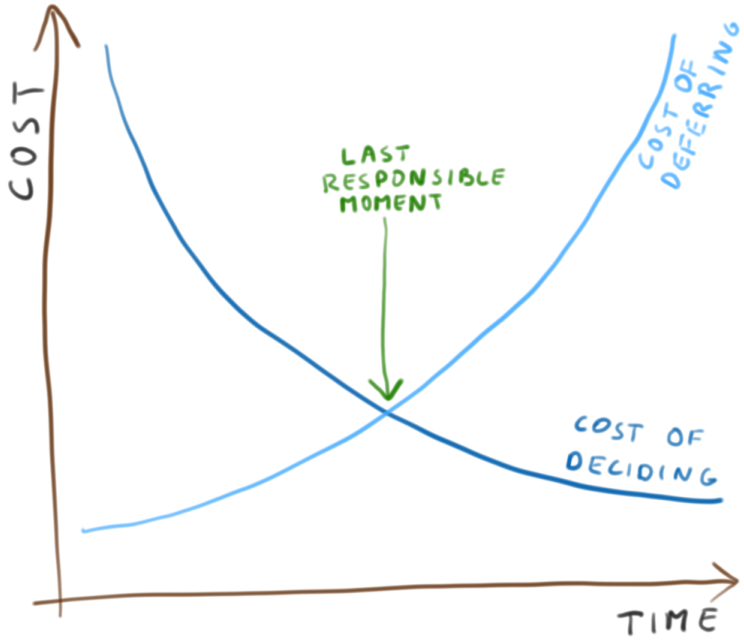


Figure 2.4: Cost of decision

2.5.1 Just in time approach

Instead of making a whole plan at the start we make only the necessary requirements and we will fill the rest up later. This way we can avoid making wrong decisions, caused by our lack of knowledge of the product. As an other benefit from this approach, we can more easily adapt to change in requirements later during the development of the product, because even our customer usually doesn't know exactly what he wants at the start.

Where plan driven development tries to predict what is not known, scrum uses exploration, prototype building and trial and error approach to buy more information so we can make better decisions. The cost of such exploration has significantly lowered in the past decades. In the past, when the plan driven development was most used, the cost of exploration was big, there weren't things like Internet for example to help you, computers were slower and technology was less accessible. But now we have the possibility to explore relatively cheaper, so why not use it.

In scrum we are making decisions just in time when they are needed, accepting change is easier for us. This approach significantly reduces the cost of change in time (figure 2.5). **[esc]**

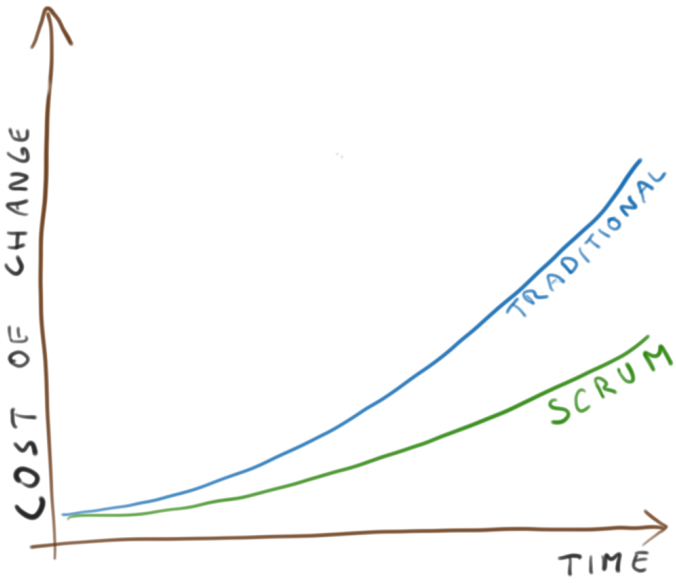


Figure 2.5: Cost of change

2.6 Scrum overview

Scrum is not a sequential step by step process, it's a framework designed for organizing work. There are core rules and principles, but everyone is free to add his own rules and create a unique implementation that suits his organizations needs.

2.6.1 Scrum principles

On figure 2.6 is an overview of core scrum principles, which will be discussed further.

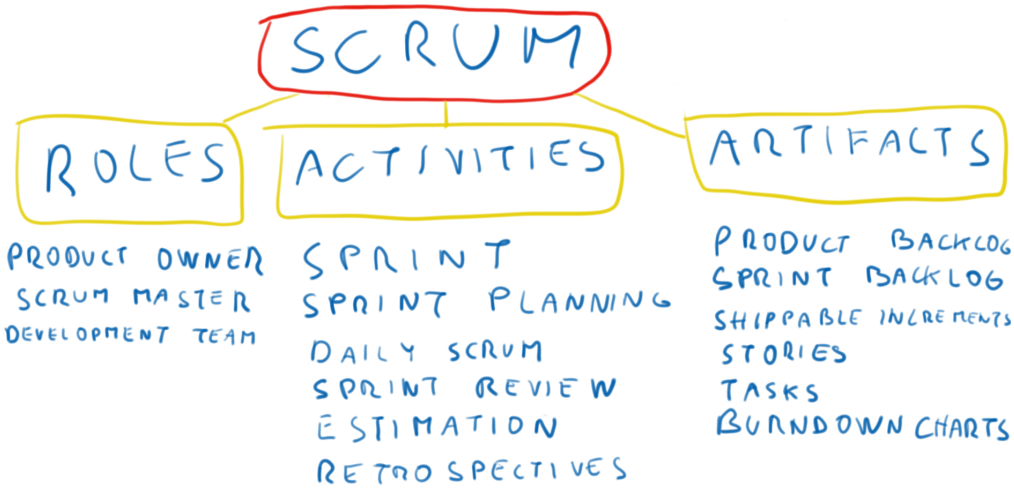


Figure 2.6: Scrum principles

2.7 Scrum roles

In every scrum based development there is one or more scrum teams. Each team consists of three basic roles: one scrum master, one product owner and development team (with several team members), additional roles can be added, but scrum requires only these three. For easy orientation in all future diagrams the **product owner** will be **blue**, **scrum master purple** and members of the **development team** will be **green**.

2.7.1 Product owner

The product owner is the leading authority of the product, he decides which features will be implemented and in which order. He is responsible for the success of the developed product. He acts like a bridge between the development team and the stakeholders (both internal and external), customers and users (figure 2.7). **[esc]**

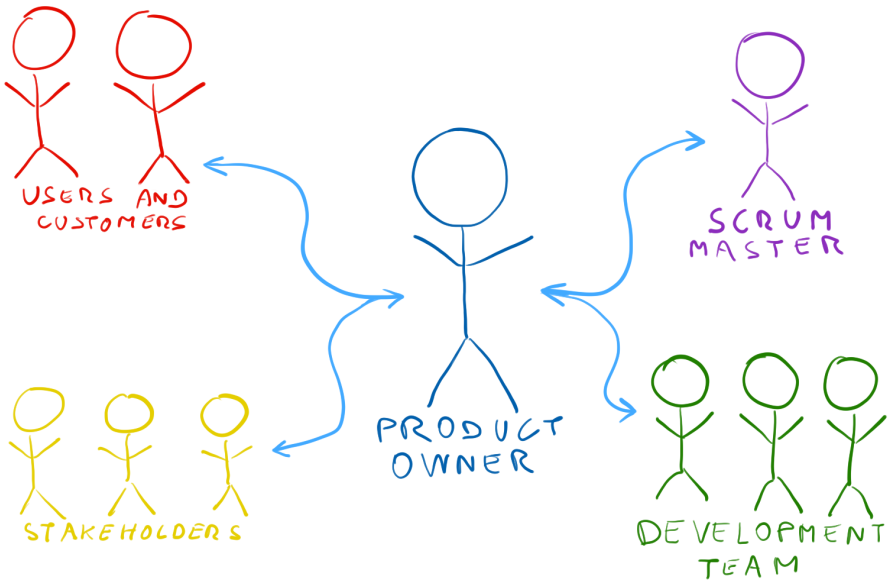


Figure 2.7: Product owner and his relations to others

As we can see above he has to divide his attention to two sides: **[esc]**

- He has to communicate with the customers, stakeholders, users etc. and then he has to understand their needs well enough, so he can act as their voice. He has the responsibility that the solution is developed in the right direction.

- Secondly he has to communicate with the development team, tell them what to build and in what order. He also has to define the acceptance criteria for the features being developed. He is responsible for ultimately deciding if the criteria is met. He has to be available to the team so he can clarify their questions about the features.

Economics

He has to make budget oriented decisions, decide if the added value of a feature is worth its cost. During the planning of the next sprint he decides if to fund the next sprint or not, he can predict the price of developing the planned features in the sprint, he knows how long the sprint is, which people are involved and how much their work time costs. With this he has to decide whether the upcoming less important features are worth the cost or not. **[esc]**

Backlog

Product owner is also responsible for managing the product backlog. He negotiates the customers needs and adds it into the product backlog, the priority of the features in the backlog is shown by its order (features on top of the list are the most important). More on how backlog management works will be told further in the text. **[esc]**

2.7.2 Scrum master

While the product owner is focused on building the right product, the scrum master is a coach, his responsibilities are helping everyone understand and embrace the scrum. He is there for both the development team and the product owner. On figure 2.8 are core responsibilities of the scrum master. **[esc]**

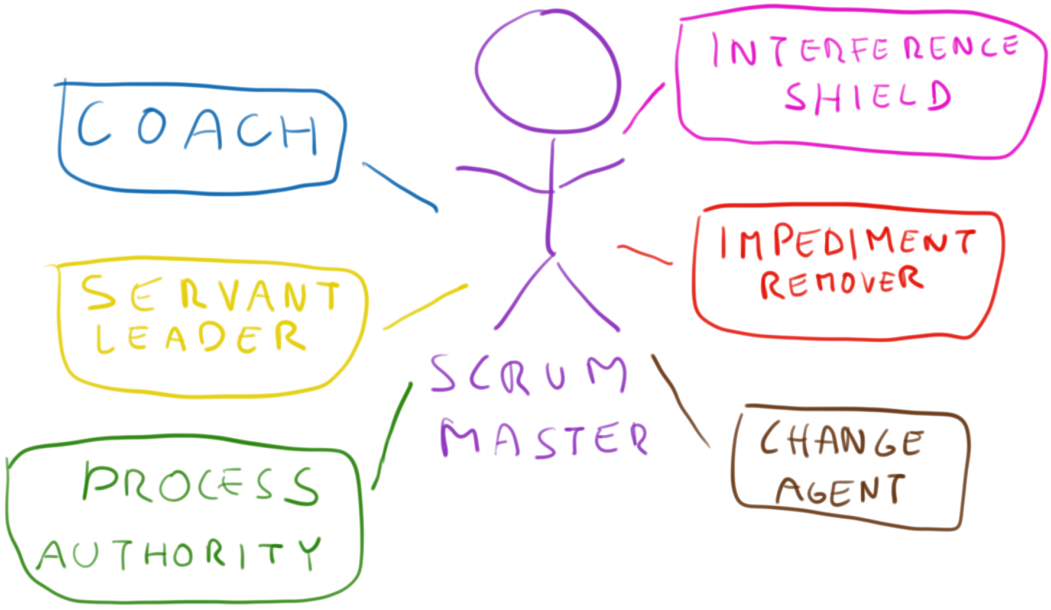


Figure 2.8: Scrum masters responsibilities

Coach

The scrum master acts as a coach or a trainer for the team and the product owner. He observes how they work with the scrum and helps them use it better. He is not there to solve their problem, he is there to help them, to solve them on their own. Only when the team can't solve it on their own, it becomes an impediment and the scrum master proceeds to solve it. **[esc]**

Servant leader "Facilitator"

Scrum master is foremost a coach and not a boss to command the team or the product owner. He is there to help the team and guide them to be more effective. His role does not give him the power to hire or fire team members, he does not dictate what task will obe done next, nor is he responsible if the work does not get done. **[esc]**

Process authority

Scrum masters job is to ensure that everyone understand the scrum process and uses it correctly. He also helps the team to customize and improve the scrum process for the teams and products needs. **[esc]**

Interference shield

Scrum master protects the team from outside interference, allowing developers to focus on the sprint and work. The interference can be a task from other team or a manager who wants something done outside of the sprints scope or change the sprint in the middle. Scrum master tries to solve those problem without bothering the team. **[esc]**

Impediment remover

If an impediment that the team cannot resolve by themselves (could be a hardware is required, another department cooperation is needed or the coffee machine is broken), the scrum master then tries to work out a solution, so the team can remain focused on their work. **[esc]**

Change agent

Since adapting to a chance is a core part of scrum, scrum master helps the team and the whole organization to change so it can embrace the scrum principles. This change can be difficult and requires cooperation of many people on many levels in the organization. **[esc]**

2.7.3 Development team

In traditional development teams usually consist of individuals with same role. There are teams of designers, database specialists, data analytics, programmers, testers and so on. One team passes work to another and there is often a problem in the communication between teams and often one team does not even know what the other teams are doing.

In scrum on the other hand we try to make cross functional teams, which consist of people with different roles. The composition of the team is often suited for the features or work the team will be doing. For example a team can be composed of one designer, one architect, two programmers, one database specialist and one tester. This enhances the communication between the different roles, a web front-end developer can directly discuss with the designer the look of the feature, instead of receiving a finished design from the design team and then having to send it back because it can't be implemented, since the designer did not know some technical limitations. The team size is typically between 5 and 9 people. It's better to have several smaller teams than one big team, since it's much easier to manage smaller teams.

Also important is that the team which implements the new feature is responsible for its testing. But it can still be viable in some situations to maintain a separate QA team (quality assurance, testing), for example it can be in the requirements from the customer. **[esc]**

2.8 Scrum activities and artifacts

Next figure 2.9 shows most important scrum artifacts and activities, which will now be discussed.

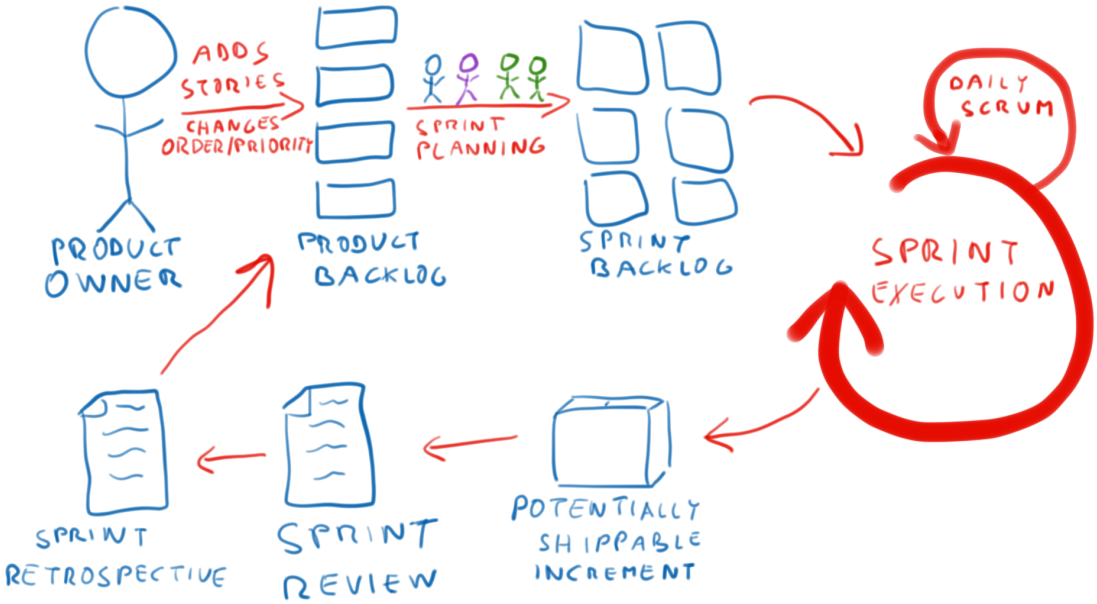


Figure 2.9: Scrum activities and artifacts

The product owner breaks down his idea of what he wants to create to a smaller set of features (stories) and he orders them by priority in a list that is called product backlog. At the start of each sprint the team and product owner agree on what set of features will be done in the sprint, this process is called sprint planning. Than comes the sprint execution, this is the time when actual work on the product is done by the development team. Each day of the sprint the team holds a small meeting, where they discuss what they did and plan to do next. At the end of the sprint the team should have a potentially shippable product increment to show. Next are two small meetings, the sprint review and the sprint retrospective, which will be discussed further. After the sprint ends a new can start, again by sprint planning. **[esc]**

2.8.1 Product backlog

The product owner is responsible for managing the work, what needs to be done and when. To help him he has the product backlog. Basically it's an ordered list of features (stories) with the most important ones on the top (figure 2.10). He fills the

backlog based on input he gets from customers, stakeholders and the scrum team itself. Product backlog can contain new features, changes to current ones, bug reports (defects) and others.

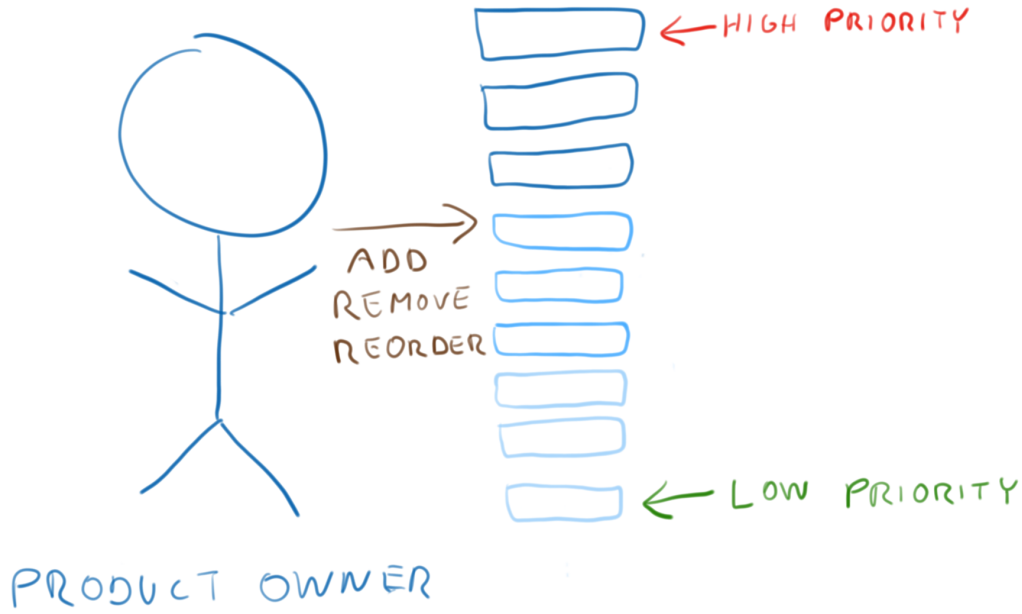


Figure 2.10: Product backlog

From the collaboration with stakeholders and other interested parties the product owner has to determine the correct order of items in the backlog, so the most important work is done first. The product backlog is a dynamic artifact, the product owner can add, remove or reorder the items in the backlog at any time. **[esc]**

2.8.2 Sprints

The work in scrum is done in timeboxed iterations called sprints (figure 2.11). Each sprint has a starting and an ending date. They typically are one to four weeks long and it is a good practice to have all sprints the same length (but it's not necessary). Each sprint is immediately followed by the next sprint. It's not good to change the scope or goal of a sprint when it's in progress. **[esc]**

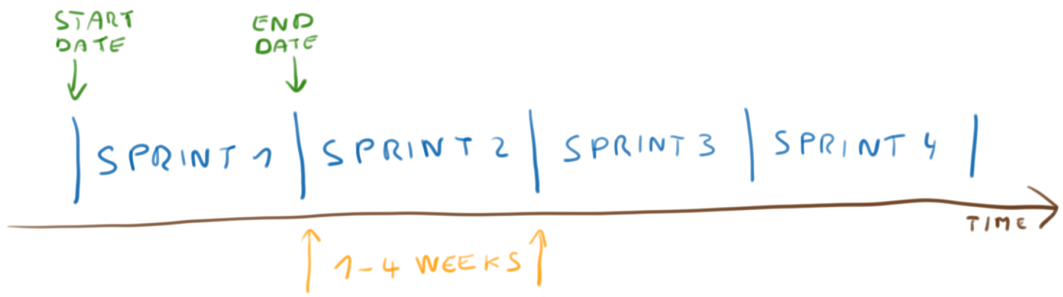


Figure 2.11: Sprints

2.8.3 Sprint backlog

The sprint backlog is very similar in function to the product backlog, but it's related only to a sprint. The sprint backlog is a result of sprint planning and contains items (stories) selected to be worked on in the next sprint.

2.8.4 Stories and tasks

The term story is very often used for one backlog item, it's a short version of user story and it can be easily explained on an example: "As a user I want to be able to view the list of my orders in the application". It can be a request for a feature, a bug report, change of some feature etc. The stories should be understandable even to people without the technical knowledge. This way it's possible to show list of stories to managers, stakeholders or customers without confusing them.

Tasks on the other hand are the necessary steps needed to finish one of the stories. Tasks are not created by product owner, they are created by the development team during the sprint planning process by breaking down the stories. **[esc]**

Shown on the example above tasks could be:

- Design the look of the order list
- Create web frontend
- Implement backend server functionality
- Add or change the database tables
- Test the functionality of the order list

2.8.5 Sprint planning

Product backlog usually contains stories for much more than what can be done in one sprint. At the start of each sprint we need to decide what is the most important work from the backlog (this is easy, since the backlog is ordered by priority).

During sprint planning all three roles should be involved, the product owner has to choose the goal of the sprint, the development team is needed to realistically determine the how much work they can accomplish in the sprint and help to estimate the stories (more on this later). The scrum master acts as a coach and process authority during the planning.

Stories are also broken into tasks (figure 2.12). This also helps with estimating how much work will be needed for the story.

When some work is not done on schedule we have to replan it for the next sprint, which can cause difficulties. The next sprint we have to take less work, so we can finish the unfinished work from the last sprint. On the other hand if we run out of work before the end of the sprint, we have to add something, which is also not ideal state. This is the reason why we estimate stories and calculate velocity, to be able to more accurately predict how much work can be done in the sprint. **[esc]**

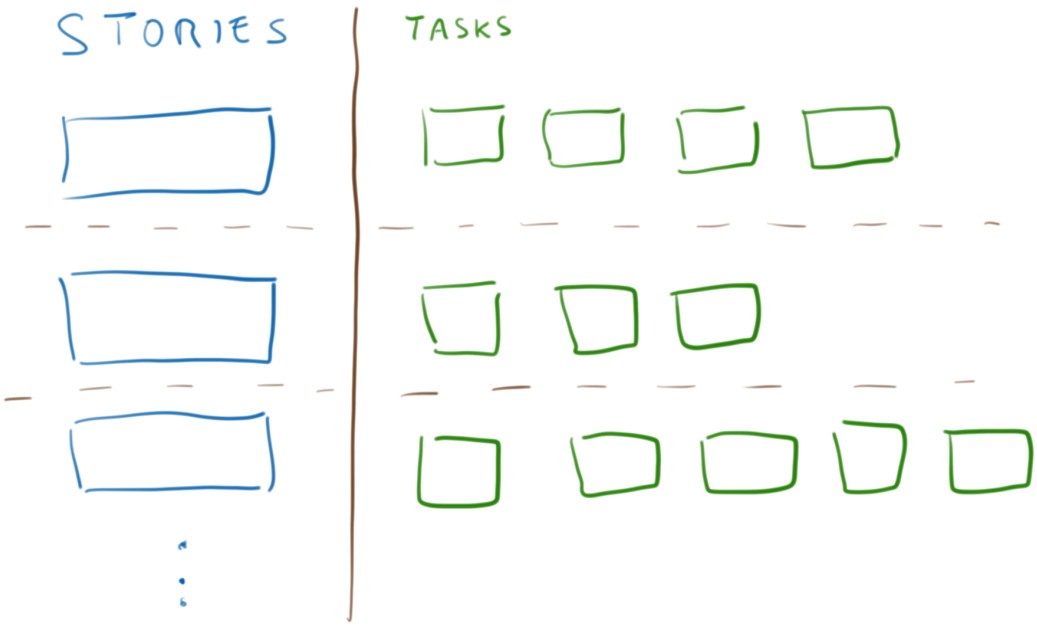


Figure 2.12: Sprint planning

2.8.6 Sprint execution

After the sprint planning is finished, the team starts working on the tasks. The team is free to choose the order in which they will perform the tasks and who will be working on which task. They can also define new tasks on the go as needed. The scrum master is always available to the team to help them with organizing the process. **[esc]**

2.8.7 Daily scrum

Or also called **daily stand-up** or just **stand-up**, is an event when members of the team hold a small meeting (usually up to 15 minutes), preferably at the same time each day.

During the daily scrum members of the team take turns under the scrum masters supervision and answer three questions:

- What did I accomplish since the last meeting
- What I plan to do until the next
- What impediments are stopping me from my work

During the daily scrum everyone from the team can get overview on who is doing what and what problems they have. The daily scrum is not a time to solve those problems, the team members usually try solve them afterwards.

An important rule is that only one person should speak at a time, and those who are not members of the team and came to observe should not speak during the stand-up. To help with this a token is often used and is passed between the attendants. Only the one with the token is allowed to speak. **[esc]**

2.8.8 Estimation

Story or task estimation is not an easy process and there are several ways of estimating how much work will be needed. Besides hours or days (which are hard to guess), the most common way is to estimate the complexity of a story relatively to another story or stories. The relative estimation is usually more accurate than absolute estimation based on days or hours.

For this an abstract unit called **story points** is introduced. Usually altered Fibonacci sequence is used to get valid values (figure 2.13). This way we have only small number of estimations and have to decide if a story is 5 or 8 for example. It also ensures significant differences in possible estimation values. Sometimes clothes sizes are used as estimations instead of Fibonacci sequence. To help us deciding we can have some estimated reference stories, which everyone can easily imagine how complex they are. Then we can say its 5 because it will take very

similar time to finish as another story with 5. When we encounter a story with high estimation, it may be good idea to try to split it to more stories if possible, because its much easier to estimate small stories then big ones.

Tasks should be usually small enough, so their estimate should not be more than few hours. If they are not, then we should split them if possible.

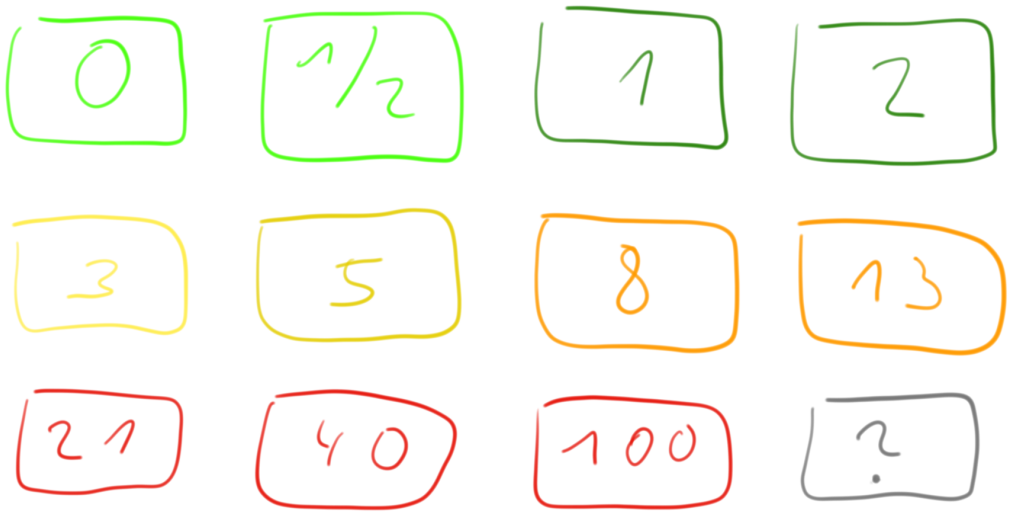


Figure 2.13: Estimation values

Planning poker

To make the process of estimation more interesting and accurate a game called planning poker (or scrum poker sometimes) can be played for each story. To begin we need cards with the above values (or we can use normal cards with some imagination, a queen can be 40 a king can be 100 for example). Very often infinity (meaning it's too big and needs to be split) or question mark (I have no idea how complex is this) are part of the cards. Sometimes a card with a coffee or some food is included to show “I am hungry and need a pause” for when the estimations are taking a long time. **[aep]**

Each development team member gets his own set of cards and they play by the following rules:

1. A story from the backlog is selected and presented to the team.
2. The development team discuses the story and asks the product owner questions if needed.
3. Each team member then privately selects an estimate (a card).

4. When all team members have their estimation, they show it to others, all at the same time.
5. If all estimates are the same we have a valid estimate for the story.
6. If there are differences in the estimates a discussion follows, often those with the lowest or highest estimate start explaining why they chose it.
7. After the discussion we can either chose the estimate based on the choices or start the estimation again (return to step 3).

We should be able to get to a consensus after at most two or three votings, because the ongoing discussion increases understanding of the story by the team. **[aep]**

2.8.9 Velocity

Velocity is the amount of work completed in each sprint (sum of story points of all finished stories in the sprint). Partially finished stories are not included, because product owner gets no value from them. This way velocity is based on size of what was done in the sprint.

We can then use velocity to predict how much work the team is able to finish in one sprint. This can help us during the sprint planning. We can store velocity from past sprints and use it to calculate average or range of minimal/maximal velocity the team was able to deliver. The product owner can for example use all this the data to predict costs. **[esc]**

2.8.10 Sprint review

Sprint review is an activity at the end of the sprint, where the development team, product owner, stakeholders and other interested parties meet and review the completed work in the sprint. Everyone can get an overview on how the development continues and influence its direction. The scrum team gets a valuable feedback on their work. The sprint review is an opportunity to adapt the product. **[esc]**

2.8.11 Sprint retrospective

This is a second activity performed at the end of a sprint. It usually takes part after the sprint review and before the next sprint planning. Opposed to the sprint review, the sprint retrospective is aimed at adapting and inspecting the process. Scrum master, product owner and the development team come together and discuss what is not working regarding the scrum process and how to improve it to make it work better. **[esc]**

2.8.12 Scrum board

Scrum board is a tool that helps the team organize work. There are many different variations on the scrum board, some are programs, some can be a white board with markers or a wall covered with colorful stickers. For example (figure 2.14) tasks are written on stickers (green) and moved on the table from open to in progress to done as work on them continues.

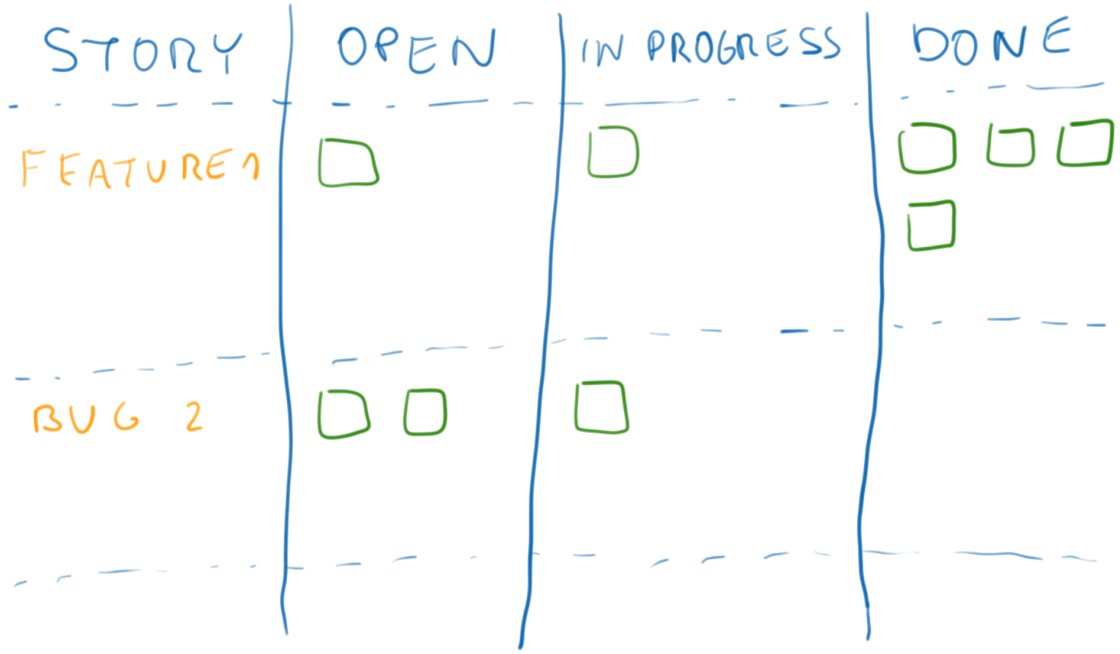


Figure 2.14: Scrum board

2.8.13 Burndown charts

The burndown chart is a representation of remaining work in time (figure 2.15). On the horizontal axis is time of the sprint in days and on the vertical axis is the remaining work (here in story points, task count or only stories count can also be used, if needed). **[ftb]**

The blue line represents ideal work progress for the sprint, the red one the actual values. Numbers represent remaining work in story points. When the red line (actual) is above the blue (ideal) it means that we are behind schedule (day 0 to 6), we have done less work then was expected. On day 6 we are on schedule, actual equals remaining work. From day 6 to 9 we worked harder and finished more work than was expected and we are ahead of schedule. On day 9 and 10 we are on schedule.

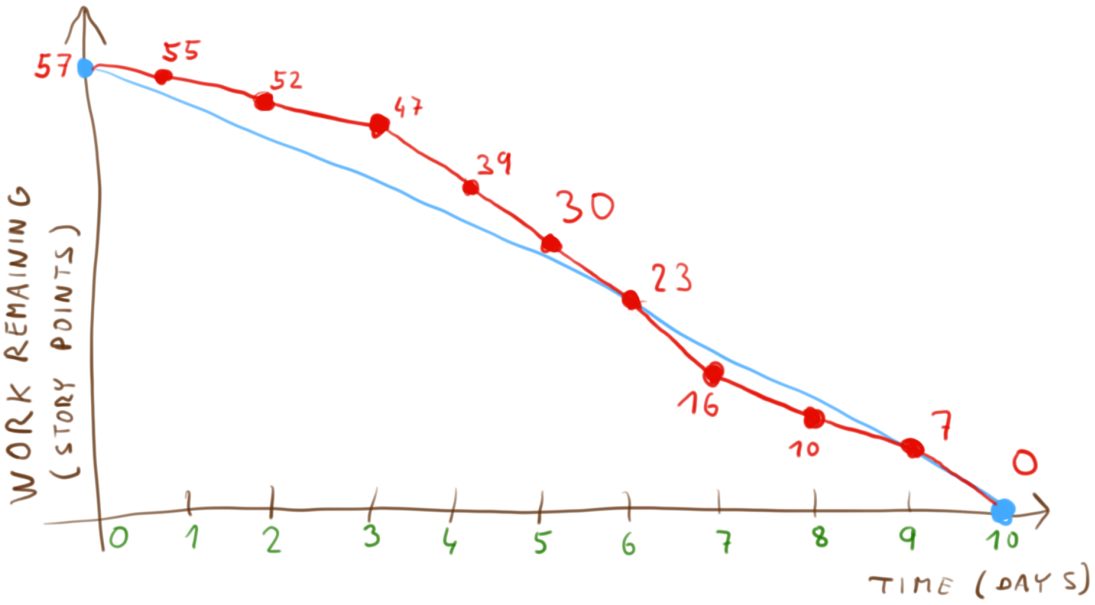


Figure 2.15: Burndown chart

In this particular example we managed to finish all the expected work on schedule. And we also did not get too far from the ideal work progress. But we still have some opportunity to improve.

On the next picture (figure 2.16) we can see three bad examples of burndown charts.

1. In first example we can see that the team was working slowly at the start (probably thinking they had enough time) and as they were closer to the deadline they realized they are far behind schedule and tried to get the job done fast. This can lead to bad coding, as you are working under pressure and try to finish as fast as possible, usually forgetting about testing.
2. On example two we took more work than we actually could do. Either we estimated the stories badly, or we just thought we are better than we really are.
3. On third example we took less work than we can do and had to add some more work in the middle of the sprint. This is also not good, we had to do a sprint planning during the sprint. We should try to estimate the stories better next time.

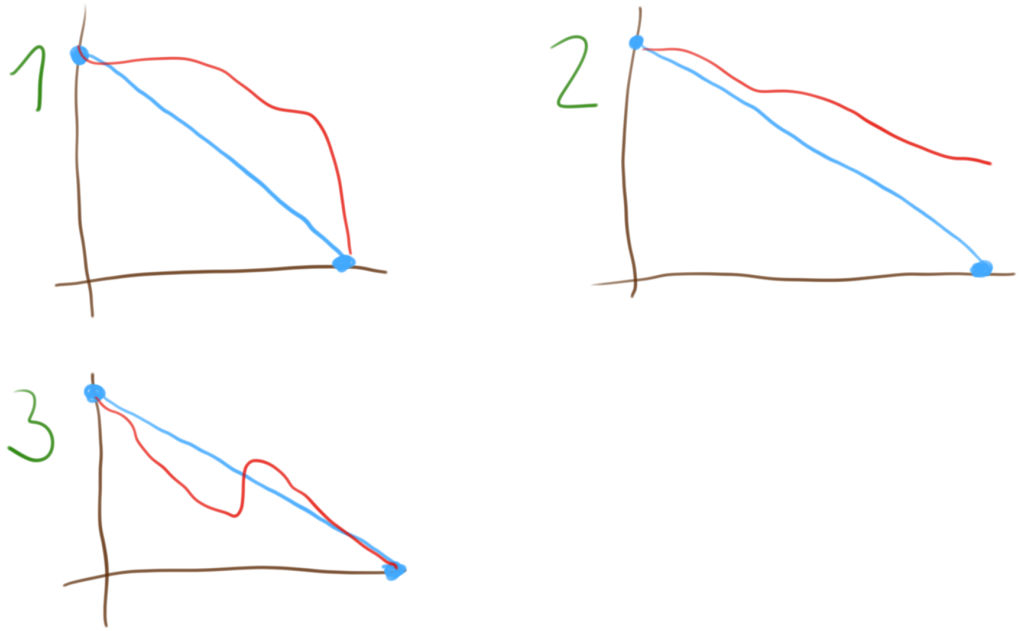


Figure 2.16: Burndown charts - bad examples

3

Multiplatform mobile development

In present time, there is an urge to have a mobile app for almost everything, this creates a demand for rapid and cheap development of such apps. Problem for fast deployment is incompatibility and big differences between major operating systems. For start I will show some graphs of market shares of major systems, then I will describe the most important systems and some of their specifics (from development point). I will follow with comparison between native and multiplatform development and lastly I will compare some of the most used multiplatform mobile development frameworks and point out their pros and cons.

3.1 Mobile operating systems

There are currently three major players on the mobile operating system market (both tablet and smartphone). The biggest number of devices are powered by Google Android, followed by Apple iOS, last place is occupied by Microsoft Windows Phone (called just Windows from version 10 and up). Other operating systems have so small market share that they are usually not taken into account by most mobile developers. On the following table 3.1 we can see market shares of sold devices by their operating system at the end of years 2014 and 2013.

Operating system	3Q 2014 Units sold (thousands)	3Q 2014 Market share %	3Q 2013 Units sold (thousands)	3Q 2013 Market share %
Android	250 060.2	83.1	205 243	82.0
iOS	38 186.6	12.7	30 330	12.1
Windows	9 033.4	3.0	8 916	3.6
Blackberry	2 419.5	0.8	4 401	1.8
Other OS	1 310.2	0.4	1 407	0.6
Total	**301 009.9**	**100.0**	**250 296.8**	**100.0**

Table 3.1: Market share by units sold **[grt]**

Such statistics differ a lot sometimes, because different data are used. There are differences in tablet and smartphone usage, on tablets iOS has a bigger share than it has on smartphones for example. Also there are big differences in market share in different regions. In Western Europe and North America iOS has bigger share than it has in other parts of world (mostly caused by lack of cheap devices with iOS).

On the next table 3.2 is mobile operating system market share measured by web access. As we can see the numbers differ from sales. They also differ a lot between two different sources.

Operating system	Net Applications March 2014 %	Net Applications July 2014 %	StatCounter February 2014 %	StatCounter August 2014 %
Android	36.6	45.2	47.6	54.9
iOS	53.3	44.2	23.0	23.6
Windows	0.7	2.5	2.2	2.4
Symbian	3.9	2.6	14.9	9.7
Bada	0.0	0.0	3.9	0.1
Blackberry	1.1	1.2	2.6	7.7
Other OS	4.4	0.1	2.2	0.6

Table 3.2: Market share by web access **[nap, scg]**

Also very important statistics for developers is the amount of money they can make on different platforms or how much are users willing to pay for applications. On the following table 3.3 is a statistics regarding mobile application stores (by platform).

Statistics	iOS	Android	Blackberry	Windows
Total app downloads	29 000 000 000	31 000 000 000	3 400 000 000	5 100 000 000
Percent users who have never paid more than $1 for an app	43 %	60 %	61 %	56 %
Average number of downloaded apps per phone	88	68	49	57
Total number of apps in store	1 205 000	950 000	230 000	320 000
Total app store revenue in 2014	$6 900 000 000	$1 800 000,000	$750 000 000	$1 250 000 000

Table 3.3: Various application stores statistics **[sbr]**

3.1.1 Apple iOS

Apple iOS (or formerly known as iPhone OS) is an operating system developed by Apple Inc. It's a closed source system, only the Darwin kernel shared with OS X (Apple's desktop and laptop operating system) is open sourced and based on Unix

(mostly NeXTSTEP, and BSD). iOS is used exclusively in Apple's own products and Apple does not license it other companies. It's used in iPhone (smartphone), iPad (tablet), iPod touch (music player) and Apple TV (digital media player). On the following picture (figure 3.1) is iOS 8 (current version, April 2015) home screen with applications icons. **[apd]**

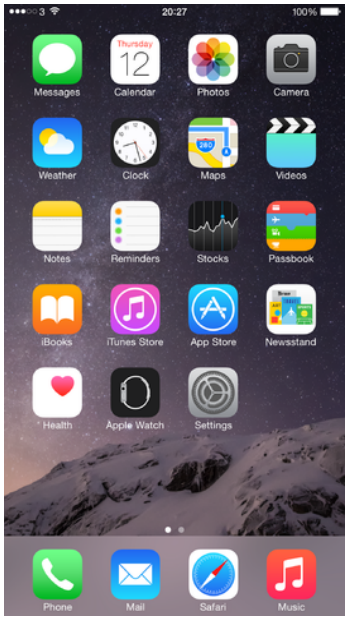


Figure 3.1: iOS 8

The system is very limited in any form of user customization or system changes. Although it's a Unix based system, no shell or shared file system is accessible to user. Applications can be installed only from official Apple's App Store and must be checked by Apple before they can be distributed among users. Application development is only possible from a Mac computer with OS X using Apple's own SDK (Software development kit) and IDE (Integrated development environment) called Xcode. There is also an annual fee for developers to allow them publish their applications and test them on real devices ($99 per year). Multitasking is available, but only one application can be running on foreground, all other applications are suspended and not running. Limited number of exceptions is allowed to run in the background - media players, downloads, voice communication (see **[apd]** for more). **[apd]**

Apple promotes two programing languages for native application development:

- **Objective-C** – which is an objective oriented compiled general purpose language based on C with Smalltalk messaging. It is the main language used in iOS and OS X. All core system libraries like Cocoa and Cocoa touch are written in Objective-C. Standard C and C++ code can be used alongside with Objective-C. **[apd]**

- **Swift** – which is a new programming language designed by Apple to replace Objective-C in all their products. It was announced at Apple's WWDC (Worldwide Developers Conference) in 2014. It is influenced by many current languages and can be compiled alongside with Objective-C and C++ in a single program. **[apd]**

Interpreted code

"An Application may not itself install or launch other executable code by any means, including without limitation through the use of a plug-in architecture, calling other frameworks, other APIs or otherwise. No interpreted code may be downloaded or used in an Application except for code that is interpreted and run by Apple's Documented APIs and built-in interpreter(s). " **[apd]**

The most commonly used interpreter is Apple's Safari Webview (a Webview is basically a full screen running web browser without any visible user controls or address bar) which allows to write applications in JavaScript, HTML and CSS. Some of the native functions provided by system (images from gallery, camera etc.) are accessible through the Webview, but you don't have access to native UI (user interface) elements (buttons, sliders etc.).

3.1.2 Google Android

Android is an open sourced operating system developed by Google based on Linux kernel. System is free to use and modify by others, but Google also provides his proprietary services and applications (including Google play store – an application and media store) for licensing, these are included in almost every android device. Android can be found in a large variety of devices including smartphones, tablets, consoles, media players, small laptops, ebook readers, mini PCs, smart watches, cars and even a fridge. Current version of Android is 5 *Lollipop* (April 2015). On figure 3.2 is the default home screen. **[and]**

Because of its open sourced nature Android can be customized nearly without limits. There are several alternative distributions based on Google's releases (CyanogenMod, MIUI, Amazon's android based OS, etc.). Almost all manufacturers customize system look and applications on their devices to differ from others and to gain advantage over competition. Applications can be installed either from official store (Google play store), from unofficial stores (Amazon, Yandex) or from installation package copied into the device. Multitasking works on the same basis as in iOS, but there is no restriction on what applications can run in background and how long it can run. **[and]**

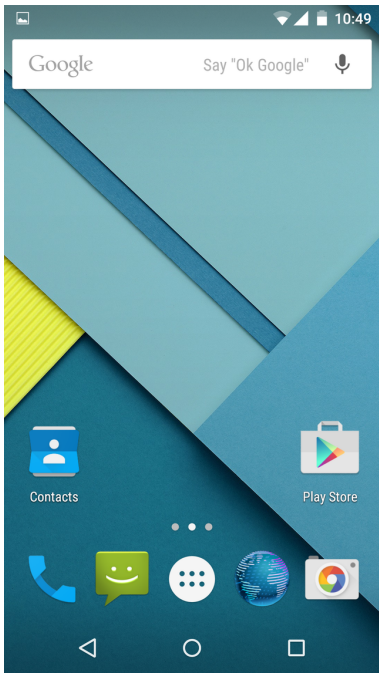


Figure 3.2: Android

Applications are mainly developed in Java and run on the device in Google's own virtual machine. An SDK and an IDE (Eclipse with plugin or newer Android Studio based on IntelliJ Idea) is available from Google. Also NDK (native development kit) is available for development in C or C++. Development tools work on Windows, Linux and OS X. **[and]**

Like on iOS there is a Webview available (based on chrome), which allows running applications written using HTML, CSS and JavaScript, this Webview also provides access to some system functions. There are no restrictions on using language interprets on Android. Like on iOS only native code has direct access to system functions, but reflection can be used to connect to the native API (application programming interface). **[and]**

3.1.3 Microsoft Windows

Windows is a closed source operating system developed by Microsoft. The mobile version (home screen on figure 3.3) is currently known as Windows Phone (current version is 8), but this name will be changed to just Windows in the next release (version 10), as the differences between the mobile and desktop version are getting smaller with each release. Microsoft is aiming to ultimately have only one version, installable on all supported devices, including smarphones, tablets, laptops, PCs, servers and more. Important note is that Windows phone is **not** evolution of the older Windows mobile (it's a little confusing, because the latest version of Windows mobile was 6 and Windows phone started with 7), but a

completely new system, which replaces Windows mobile (development and support of Windows mobile was already discontinued). There is no backwards compatibility with Windows mobile applications. Multitasking works the same way like on other platforms, but running background tasks is like on iOS very limited (see **[ms]** for more details). **[ms]**

Figure 3.3: Windows

SDK and IDE (Microsoft Visual Studio 2013) are available and development for Windows Phone 8 requires Windows 8.1 Professional or better (could be possible to use lesser version, but device emulation will not be available). Windows 10 is not yet released so requirements for development are not yet known (April 2015). **[ms]**

There are two main platforms used to develop applications for Windows phone:

- **Silverlight** – which is an application framework created by Microsoft, early versions aimed for web based applications as alternative to Adobe's Flash. Applications can be developed in any language supported by .NET framework, most commonly used is C#. **[ms]**
- **Windows runtime** – which is shared with desktop version of Windows (it's a successor to Win32 API) and allows deploying an application on both mobile and desktop with minimal changes. Development is possible for example in C#, VB.NET, and also in C++/CX. **[ms]**

Like in iOS and Android, Microsoft supports developing applications in JavaScript, HTML and CSS and deploying them using a Webview based on Internet explorer. **[ms]**

3.2 Native vs multiplatform development

Here is a difference between native and multiplatform (also known as cross platform) development from a general point of view, different multiplatform frameworks will be described further in this chapter.

3.2.1 Native

Native development offers better performance since systems are better optimized for running their native code. Applications can interact more easily and have access to all system functions and APIs. It is also much easier to make application design fit the system, because native UI elements can be used. On the other hand code cannot be usually reused on other platforms, basically the only native code usable on all three above mentioned platforms can be C/C++ code, without any system specific libraries used. Meaning developers will have to write code multiple times for each platform and learn the specifics and language of each platform they want to target.

Pros

+ Better performance
+ Access to all system functions
+ Native UI elements and look

Cons

– Not portable code
– Have to write and maintain code for all targeted platforms
– Developers have to learn platform specifics

3.2.2 Multiplatform

Multiplatform development aims on better portability of code, trying to increase the amount of code that can be reused. Some of the code can be also useful elsewhere, not only on the mobile devices. The idea behind a lot of mobile applications is, that we already have a mobile website for our service, so why don't we reuse the same code. As a result most of the mobile development frameworks focus on using a Webview to display the application in form of JavaScript, HTML and CSS. For many companies it is easier to find web developers, who know above mentioned technologies (or they most probably already have some in the company), than finding mobile developers. Another benefit of using one code on more platforms is easier maintainability of the code.

The biggest drawback of multiplatform approach is increased hardware requirements for such applications. Launching a Webview and displaying a rich JavaScript website in it consumes significantly more system resources than a native application using native UI elements. From my experience a Webview on Android or iOS can eat up more than 50 MB of RAM memory, while displaying only an empty HTML page, on the other hand same native UI needs about 10 times less memory. This can be a problem especially on cheaper Android phones, where system resources are very limited. Same problem is with drawing speed and performance on drawing web pages, which is more resource hungry and drawing is slower than native UI which is better optimized and draws faster. There are frameworks that try to remove this problem by instead of using a Webview they create the UI from the platforms native UI elements (such as Xamarin), but this may not work well since some of the UI elements can work differently on different platforms.

Other notable obstacle is access to system APIs and functions. Some of them are a lot different for different platforms and some may not even exists on some platforms. Most of the frameworks try to solve this by wrapping the native functionality by their own API to provide unified access. Some allow developers to implement their own wrappers (PhoneGap for example).

Pros

+ Reusability of code
+ Easier maintenance of code
+ Developers can use already known technologies

Cons

– More resource consuming applications (both memory and computing power)
– Harder access to system functions
– It can be a problem to create native looking application

3.3 Multiplatform development frameworks

Here I will describe and compare some of the most used multiplatform frameworks for mobile application development. There are so many companies selling multiplatform mobile frameworks, that it is impossible to describe them all. Most of these solutions stand on using Webviews and JavaScript, but there are some other interesting technologies.

3.3.1 Apache Cordova (PhoneGap)

Apache Cordova is an open source framework for creating multiplatform mobile applications using web technologies (HTML, CSS, JavaScript). It is also known by the name PhoneGap. PhoneGap was originally developed by Adobe/Nitobi, but Adobe donated the codebase to Apache Software Foundation (ASF) and thus the name change. Adobe still uses the PhoneGap name for its own distribution, but it is based on the same code. It can include some Adobe specific tools (they also have paid version with various services). Apache Cordova and PhoneGap are free to use for both personal or commercial use and currently the only difference between the two is the name. Cordova can be combined with most Web UI frameworks like *jQuery Mobile*, *Dojo Mobile* or *Sencha Touch*. **[cor]**

Cordova uses a Webview to interpret the JavaScript and display the UI to the user. It provides a unified JavaScript API for accessing several native functions of the device like camera, storage, notifications and so on (see table 3.4 for details). Applications can connect to servers using Internet (XmlHTTPRequest, Web Sockets, etc.) to communicate with backend services. **[cor]**

Feature / OS	iPhone < 3GS	iPhone > 3GS	Android	Blackberry 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	×	✓	✓	×	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
Files	✓	✓	✓	✓	✓	✓	✓	×
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	×	✓	✓	✓	×
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (vibrate)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.4: Features available in Cordova **[cor]**

Additional native functionality can be added through plugins. Many plugins can be found online or if we need something special, we can write our own plugins using the plugin API. We have to create the unified interface so we can access it from JavaScript and than we have to implement the desired native functionality for each needed platform. **[cor]**

Pros

+ Free to use (personal and commercial)
+ Open source
+ Development using web technologies (HTML, CSS, JavaScript)
+ Reusability
+ Supports many platforms
+ Access to several native APIs
+ Plugins

Cons

– Webview is more resource hungry than native UI
– Missing native features without existing plugins require native development of own plugins
– No easy way to have native UI look (application is a web page)

3.3.2 Qt

Qt is one of the most popular multiplatform application development frameworks. Applications written in Qt can be deployed on many platforms, including all major desktop operating systems and also mobile ones. Support for mobile platforms have been added only recently (iOS and Android in December 2013, Windows phone 8 in December 2014) and features are still being added. Qt is owned by Qt company owned by Digia, which owns rights and trademarks to Qt. Qt is developed as open source project and available under GPL and LGPL license. Digia also provides a commercial license for Qt. **[qt]**

Licensing is possible in following packages: (see **[qt]** for more detailed info)

- **Community** – Open source or application must be compatible with GPL or LGPL license (does not mean it has to be distributed under LGPL license)
- **Indie mobile** – Commercial license for mobile development only, costs $25 per month per developer
- **Professional** – Commercial license for mobile and desktop development, costs $174 per month per developer, includes some advanced UI elements and Qt quick compiler
- **Enterprise** – For big companies, individual pricing

Documentation states that development of closed source applications for iOS using the community edition is not possible and violates the Apple Store terms and conditions, since no dynamic linking is allowed and static linking requires sharing of source code by LGPL (there exists a disagreement in this and whole page on Qt wiki is dedicated to this issue, but there is no clear conclusion). **[qt]**

Official IDE called Qt creator is available for application development and works on Windows, Linux and OS X. Main development languages used in Qt are C++ and QML (with optional JavaScript use). QML (see figure 3.4 for example) is a declarative language used for UI creation and can be combined with JavaScript to provide application logic. Qt allows deploying applications in native code or using a Webview. QML code can be compiled either Just-in-time (JIT) on the application startup or using Qt Quick Compiler during the application compile, this allows faster application load and better performance (and is required on iOS, because of Apple's limitations of code interpreting on devices, see iOS section in chapter 3.1 for more). Also it is possible to write the whole UI in C++ only using the Qt widgets. **[qt]**

```
import QtQuick 2.3

Rectangle {
    width: 200
    height: 100
    color: "red"

    Text {
        anchors.centerIn: parent
        text: "Hello, World!"
    }

    MouseArea {
        anchors.fill: parent
        onClicked: parent.color = "blue"
    }
}
```

Figure 3.4: QML example

Unified communication with many device native APIs is provided in Qt and missing functions can be also implemented by developers, but the development is more complicated than in Cordova. There is also a possibility to mix the Qt C++ code with native compiled code on some platforms (Objective-C on iOS for example), including the native UI elements, but such application is no longer multiplatform. **[qt]**

Many low market share mobile operating systems are using Qt for native application development including Jolla Sailfish OS, Ubuntu touch, Blackberry, Symbian and MeeGo. **[qt]**

Pros

+ Open source
+ Targets nearly all available platforms including desktop and embedded devices
+ Good IDE for all major desktop platforms is available
+ Apps can be compiled to platform native code or Webview can be used
+ Compiled code offers better performance than Webview
+ Connection to many native APIs
+ Nearly native UI look for some platforms
+ Can be mixed with native platform code

Cons

– Commercial license required for closed source iOS development
– Some features like Qt quick compiler or advanced UI elements are available only in paid versions
– More complicated development of missing native API functions
– All used Qt libraries have to be shipped with the app, even if another app with them is already present on the device (there is solution which allows library sharing on Android, but it can be confusing to normal user and not many apps use it)

3.3.3 Xamarin

Like Qt, Xamarin focuses on multiplatform applications using native code and native UI elements. Xamarin offers their own IDE called Xamarin studio, which works on OS X and Windows and also integration with Microsoft Visual Studio on Windows. The main development language is C# and applications can be deployed on Android, iOS and Windows Phone. The technology behind is based on the Mono project (open source framework and runtime for C#). On iOS applications are compiled to native ARM assembly code, on Android code is compiled to intermediate language (IL) and IL runtime is included, which then compiles the code on the device on application startup, Windows phone has C# as native

language so runtime is already present in the system. For more details about C# you can see **[ms]**. Xamarin offers limited free license and commercial licenses for application development. **[xam]**

License options: (see **[xam]** for more detailed info)

- **Starter** – only free option, limited application size, no native API connection and some features are not available
- **Indie** – Commercial license for individual developers or small companies (up to 5 developers), no Visual studio integration, $25 per developer per month
- **Business** – Commercial license for companies, all features, limited support from Xamarin $999 per developer per year
- **Enterprise** – Commercial license for big companies with extended support and assistance from Xamarin, $1899 per developer per year

Xamarin offers two ways of UI development, one is using different UI for each platform build from platform native UI elements and with direct access to all platform specific APIs (but this way only some code can be shared and some code is platform specific). Second option is *Xamarin.Forms* that allows for single UI for all platforms, it is also build using native UI elements (not available in starter edition), *Xamarin.Forms* aims at maximum code sharing. Both ways can be combined in single application. On figure 3.5 is example of one app made with *Xamarin.Forms* on three different devices. **[xam]**

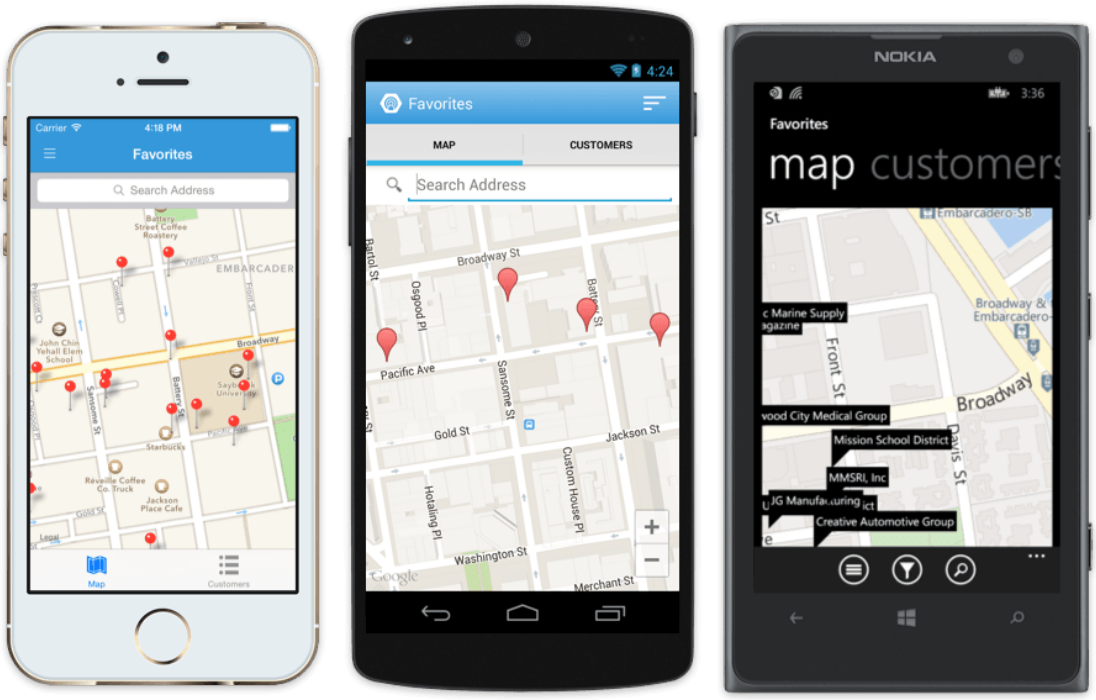


Figure 3.5: Xamarin.Forms

Pros

+ Most advanced multiplatform mobile development framework
+ Very good documentation and development guide
+ Supports all three major platforms
+ Good own IDE and MS Visual studio integration
+ Compiles to native code, better performance than Webview
+ Native look on all supported platforms
+ Connection to many native APIs
+ Binding to existing native code possible (Java, C++, Objective-C)
+ Special licenses for students, universities and open source projects

Cons

– Free version is very limited and unusable for any more serious development
– Not open source (except for Mono)

3.3.4 Appcelerator

Appcelerator is business oriented multiplatform mobile cloud platform based on web technologies (HTML, JavaScript and CSS). It focuses on integrating and connecting various existing business cloud services. They offer both a Webview apps and native ones. It is closed source solution aimed on large companies. There is no free version and pricing starts on $39 for single developer per month. Also additional functionality and plugins can be bought on their marketplace. **[apc]**

Licensing options: (see **[apc]** for more detailed info)

- **Indie** – only for one developer, $39 per month, limited functionality and no support
- **Team** – for small teams and businesses, $259 per developer per month, limited support
- **Enterprise** – for large companies with premium support, individual pricing

Pros

+ Integration with many cloud services
+ Huge number of plugins with various functionality
+ Own IDE

Cons

– No free version, expensive
– Too complicated for small applications development

3.3.5 Sencha touch

Sencha touch is a JavaScript multiplatform web application development library. It focuses on both mobile websites and mobile applications. Sencha touch offers a variety of tools to help with development including IDE plugins, Sencha architect (advanced HTML5 UI editor) and more. Sencha offers native looking themes for applications and a web components that can be integrated into applications (graphs drawing tools for example). Application packaging for mobile devices is done using PhoneGap / Cordova, which means it is possible to use existing Cordova plugins or own plugins with Sencha touch. **[sen]**

Licensing options include various products individually or a complete bundle for 5 developers ($4825 per year) or 20 developers ($18895 per year), see **[sen]** for more detailed info.

Pros

+ Complex JavaScript library for mobile web and application development
+ Huge number of plugins with various functionality
+ Development using web technologies (HTML, CSS, JavaScript)
+ Reusability
+ Supports many platforms
+ Access to several native APIs
+ Based on Cordova (can use Cordova plugins)

Cons

– No free version, expensive
– Webview is more resource hungry than native UI
– Missing native features without existing plugins require native development of own plugins

3.3.6 Online app builders

Services that have seen a big boom in the past few years are various application builders or generators. They very often offer simple development environment in browser, mostly WYSIWYG editor with pre-made components. (see figure 3.6 for example). Typically applications are developed using HTML5, JavaScript and CSS and deployed using Cordova or Xamarin. Good side is that applications can be made very easily and even managers can do it. This makes this approach very good for fast creation of various application prototypes to show to customers. This prototype can be then used as a template for the final application made by real developers. Some of these services even offer remote testing on real devices.

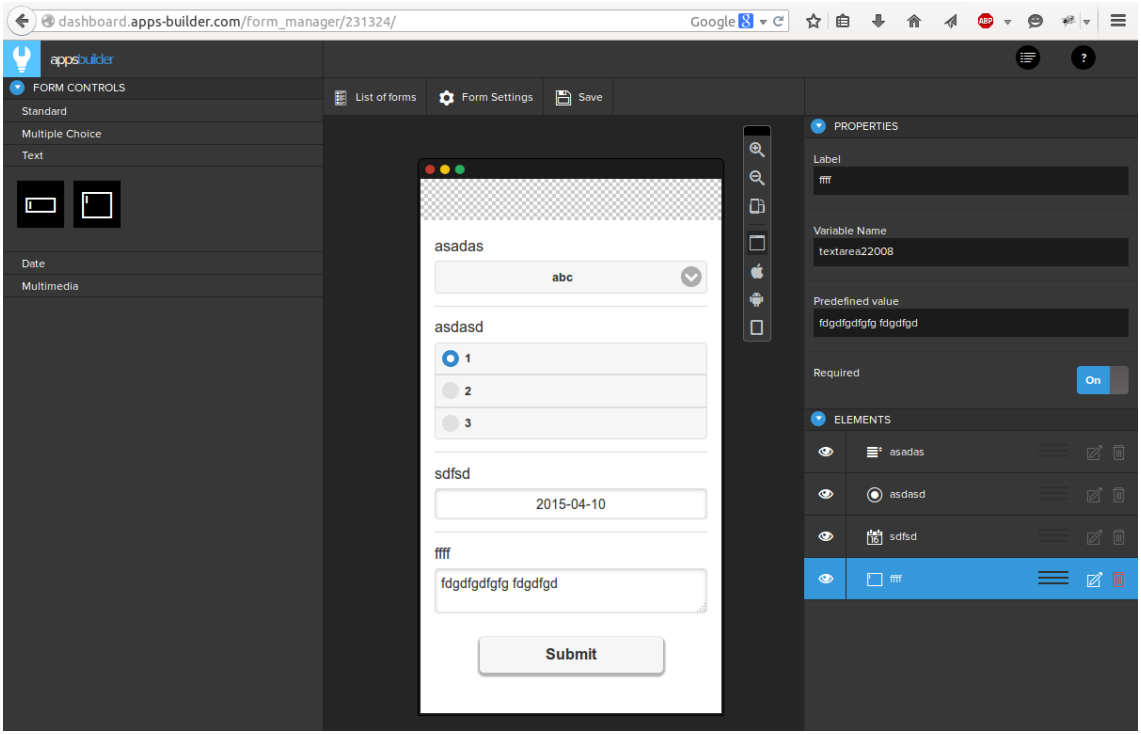


Figure 3.6: AppsBilder editor in browser

Notable application builders are for example AppsBuilder, Altova MobileTogether, Appery.io, Appy Pie, Mobincube, TheAppBuilder, Good Barber, AppMachine, Como and many more exists. Most of these solutions are paid or the free version offers very limited functionality.

Pros

+ Fast and simple development
+ Good for prototyping
+ WYSIWYG editors
+ Can create applications in browser

Cons

– Usually paid or very limited functionality
– Not good for complex applications
– Webview is more resource hungry than native UI

3.3.7 Pure Webview application

Sometimes there is no need to use any framework at all, when we have no use for their added functionality, or we want something none of these frameworks can do. We can use the Webview each platform provides on our own. There will be some small overhead, because we have to create several native applications with Webview component in it, but as a result we will have full control over our application. All platforms offer a way how to interact with native code from the Webview (Cordova uses it too for example). We can then develop our application using HTML, CSS and JavaScript.

3.3.8 Games

There also exists a huge amount of multiplatform game engines. The situation with games is a little different than with traditional applications, games usually don't require any system specific API, UI is created using *OpenGL* or *DirectX* so native UI elements are not needed either. This way most of the code can be shared between different platforms more easily.

The most known one is Unity, like Xamarin it is build on the Mono project, so main programing language is C#. Games written with unity can target almost any possible platform available including mobile devices, desktop (Windows, Linux, OS X), game consoles or even virtual reality devices. Other notable multiplatform game engines are Unreal engine, Marmalade or Corona.

Apple Game center and Google Play services

Both Apple and Google offer a SDK with API that can be used to create multiplayer games with ability to connect two or more devices over the Internet without the need of a server, but this designed to be used with games and does not integrate well with normal applications. The most notable problem is both of these solutions are locked to their platform and don't offer cross platform connections.

3.4 Conclusion

In this chapter I described the difference between multiplatform and native development, their ups and downs and major mobile operating systems. I compared several most used multiplatform development frameworks and approaches. For my application I have chosen to use Cordova, since it's completely free, can be easily enhanced with missing functionality and lot of material exist to help with development. Also in late summer 2013, when I made this decision, the other frameworks were not as mature as they are now in April 2015, when I am writing this. I will describe Cordova more closely from a developers point of view in one of the further chapters. Main targeted platforms will be Android and iOS, because they share the vast majority of market. I will leave the Windows phone version out, but there shouldn't be any problem with adding it in the future.

4

Existing solutions

There are plenty of mobile clients for existing cloud project management services, but mostly they rely on connecting to a server to work. Also not many mobile applications are oriented on scrum and its activities. There is a big number of applications aiming solely on the scrum poker, other big group consists of interactive scrum guides, but those are more of packed books, than actual applications. I will try to compare some of the more interesting scrum applications in this chapter, I will not include project management applications which do not have anything in common with scrum, as I am interested only in scrum. There are also several applications, which aim on helping the scrum master, but they don't implement the whole scrum framework. I will try focus on applications, that are available and work on multiple platforms, rather than single platform ones. As there are many similar applications, I will write only about those that are interesting for a specific reason.

4.1 Scrum poker applications

There are many scrum poker applications available on Google Play or Apple App Store. Many of them just work as a replacement for cards and don't offer any form of communication. There exist few applications that offer some form of communication mostly WLAN/zeroconf, Bluetooth, Game center, or a server (see next chapter for more about communication). I was able to find one multiplatform scrum poker game called Estimated (Android, iOS or web page), which uses a server to synchronize. It allows to pick a card and others don't see your pick until all have voted or owner can force the voting end (see figure 4.1 for screenshot **[est]**).

Other applications can be found with names like Scrum poker, Scrum, Planning poker etc.

Figure 4.1: Estimated

Pros

+ Easy to use
+ Many free applications
+ Multiplayer applications

Cons

– Only scrum poker

4.2 JIRA Connect Enterprise

JIRA is a popular issue tracker and project management tool. It can be deployed on own server or used as a cloud service. It supports agile and scrum planning. Mobile applications are available for Android and iOS. Those applications are just clients and require connection to the server, also the features are limited, comparing to the web application. Use of the mobile application requires to have "Mobility for JIRA" add-on installed on the server, which is paid. See figures 4.2 and 4.3 for some screenshots. **[jira]**

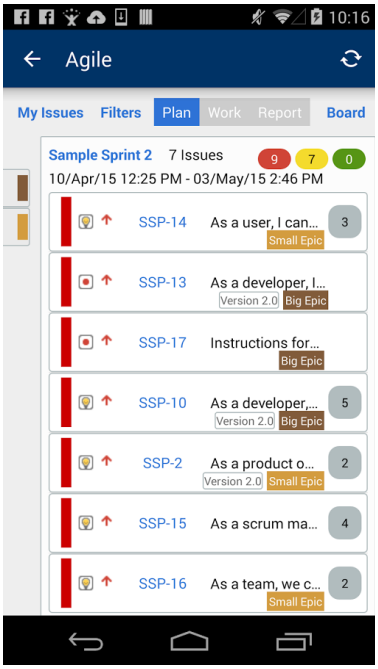


Figure 4.2: JIRA android

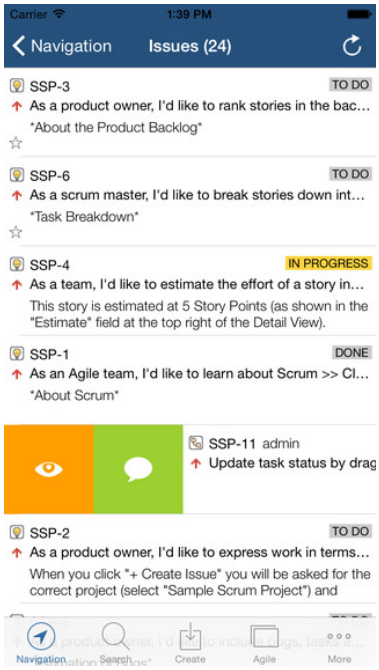


Figure 4.3: JIRA iOS

Pros

+ Powerful tool
+ Many features, can be configured for scrum
+ Add-ons

Cons

– Paid solution
– Mobile applications have limited features
– Requires connection to server to work

4.3 AgileScrum Pro

AgileScrum Pro is iOS only app for managing sprints and scrum board. It is interesting because it uses email or box.net for sharing project data with other people. I wasn't able to test the sharing in any depth, because I have only one iOS device, but it looks like it is mostly meant to send out the project data from one central device, so others can read it. Screenshot on figure 4.4. **[asp]**

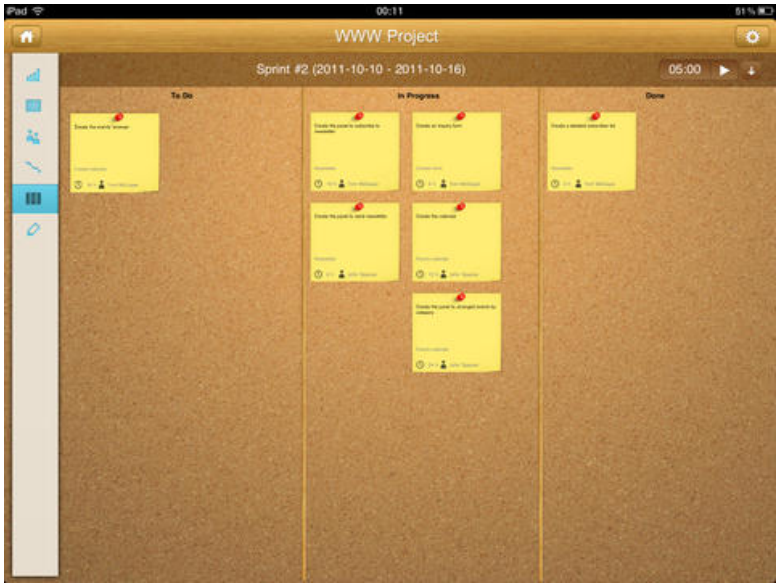


Figure 4.4: AgileScrum Pro

Pros

+ Easy to use
+ Sharing over email or box.net

Cons

– Limited features (no scrum poker for example)
– iOS only
– No phone UI (iPad only)

4.4 Lion Monkey Scrum

Lion monkey is very similar application to AgileScrum Pro, it has more features, also has only iOS version (figure 4.5), but sharing is done only using iTunes by exporting/importing data as CVS files or SQLite database. **[lms]**

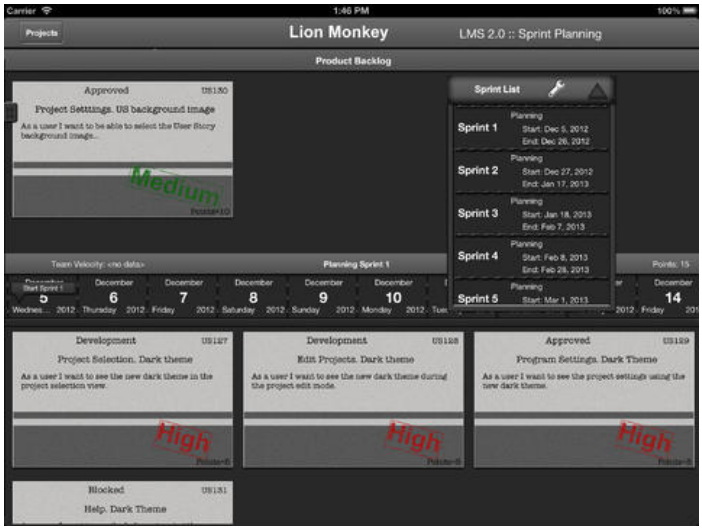


Figure 4.5: Lion Monkey Scrum

Pros

+ More features, but also no scrum poker
+ Sharing using CVS files or SQLite database

Cons

– No scrum poker
– Complicated UI
– iOS only

4.5 Pivotal tracker

Pivotal tracker is like JIRA another big project management cloud application. It focuses on agile development. Like JIRA, connection to the server is required and Pivotal tracker is paid solution. Official mobile application is only for iOS (figure 4.6). **[pvt]**

Figure 4.6: Pivotal tracker

Pros

+ Powerful tool
+ Many features, focuses on agile development
+ Add-ons

Cons

– Paid solution
– Mobile application have limited features
– Requires connection to server to work
– iOS only mobile application

4.6 Conclusion

There are many scrum oriented mobile applications, most of them are simple apps for scrum poker. There are also clients for big project management cloud applications, which have limited set of features and require access to server. There are several local scrum management apps, most of them are for iOS only and their ability to synchronize data between devices is very limited.

5

Communication

In this chapter I will describe possible methods of communication between mobile devices. Since my intention is to avoid having a dedicated server running somewhere, I will focus mainly on technologies that allow communication directly between devices without any public server. Two most promising candidates are Wi-Fi and Bluetooth.

5.1 SMS

SMS messages could be used to share data between devices, but without an unlimited messaging plan this could lead to a very high bill. Another notable problem is, that on some platforms it's not allowed to read SMS messages from an application (iOS **[apd]**, Windows Phone 8 **[ms]**). Also not all devices are capable of sending or receiving SMS messages (many tablets don't have GSM module).

5.2 Communication over Internet

Connecting from one device to second over Internet is a problem. Mobile devices in GSM network don't have public IP address or other easy way of communicating with them. When we want to communicate over Internet, we need some third public point which both devices know. Also we don't want to have opened listening socket on mobile connection all the time, this would drain the device's battery really fast.

5.2.1 Push notifications

To overcome some of these issues Apple, Google or Microsoft have special service called push notifications. General principle is that a server called push server exists and it's known to all the devices. When an application on the device wants to receive push notification from some service it asks the push server to create a token for it, then it gives the token to the service it wants to receive the notifications from. The token is application and device specific. When the service

wants to contact the device it sends the message to the push server instead. The device maintains an optimized communication channel with the push server, the push server delivers the notifications over this channel only. This way notifications for all applications on the device are delivered over only one connection, which is optimized specifically for this task.

The sender of the notification has to prove himself by a certificate, a pair of SSL certificates is used, one is uploaded by application developer to the push server (public key), second is used by the sender of the notification (private key). Normally we use a server to send notifications, each mobile device requests a push token from the push server and then sends it to our server, when our server wants to send a notification, it signs it with the certificate and including the device token it sends it to the push server.

Since I want to avoid having a server, this approach can't be used. Technically it could be possible to send the push notification from a mobile device, but it would be insecure. Also it violates the terms and conditions of most of the push services and I would still have to distribute the device token first, before push notifications can be used.

5.2.2 Email, File sharing services

One possibility for sharing data over the Internet would be to use emails or some file sharing service like DropBox or Box.net, where we could upload project data. This way we could use existing infrastructure without the need to create any dedicated server. As a drawback it would be very difficult to manage concurrent updates from different devices, because we can't run any code on those servers.

5.3 Local communication

Local communication is limited to only some group of devices, which are able to reach each other. This limit could be distance between devices or a access to shared network for example. Two most used technologies for this are Bluetooth and Wi-Fi.

5.3.1 Bluetooth

Bluetooth is a short range wireless communication technology present in almost all phones and tablets. It is a WPAN (Wireless Personal Area Network) an works in the 2.4 GHz ISM band. This frequency band is 2400 - 2483.5 MHz. Bluetooth focuses on low power consumption, low cost and robustness. Bluetooth is standardized by IEEE as IEEE 802.15.1, but is maintained by Bluetooth SIG (Bluetooth Special Interest Group), they oversee and develop the standard. Current version of Bluetooth is 4.2. Bluetooth allows device independent communication, it is used

for communication between computers, phones and tablets, but also for communication with various other devices, including keyboards, mouses, cameras, watches and other wearable electronics. **[bt]**

Bluetooth operates on a master – slave ad-hoc architecture. One master can communicate with up to 7 slaves. It is also possible for one device to act as both master and slave and form a more complex network with several masters called scatternet. Bluetooth specifies set of profiles, which describe possible ways and forms of communication like A2DP (Advanced Audio Distribution Profile), OBEX (Object exchange) and many others (see **[bt]** for more details). **[bt]**

Bluetooth defines three power classes, most mobile devices are in the second class (max power – 2.5 mW) with maximal range around 10 meters. Theoretical speed is 25 Mb/s. **[bt]**

Devices must pair first to be able to communicate, this sometimes causes problems, when one device can't see the other (this problem mostly occurs with older devices).

Pros

+ No need for additional hardware
+ Present in almost every device
+ Low energy consumption

Cons

– Small range
– Need for device pairing
– Low speed (with older versions)

5.3.2 Wi-Fi

Wi-Fi is a WLAN (Wireless Local Area Network), it is defined in IEEE standard 802.11 and is maintained by the Wi-Fi Alliance. Wi-Fi works in the 2.4 and 5 GHz ISM radio bands. Wi-Fi is an implementation of the first and second ISO/OSI layers. Wi-Fi is present in almost all mobile devices, including phones, tablets or laptops. Wi-Fi is very often used to allow mobile devices to access Internet. An access point (or hotspot) is used as a central point to which all the devices connect. This can be either some router or another phone can make access point from itself and allow other devices to connect and use its Internet connection for example. Transfered data are organized in packets and usually TCP/IP protocol is used for communication. The access point can be secured by encryption and require

a password to allow connection. When a device connects to an access point it receives a local IP address from the access point, that identifies the device in the network. **[802.11]**

Wi-Fi has higher range than Bluetooth, over 20 meters in buildings and over 100 meters outside, and also much bigger speeds can be achieved, the 802.11ac can theoretically go over 1 Gb/s. Range and speed varies by used frequency (2.4 GHz has slightly better range than 5 GHz) and also newer standards like 802.11ac have better speeds than the older ones.

Zero configuration networking (zeroconf)

When we want to send data to some device we need to know its IP address. The most simple way to exchange IP addresses between mobile devices on the network is to use device discovery technology. One such technology is called *zeroconf*, it is a mechanism that uses multicast messages to advertise and discover devices and services in the network. It does not require any special server or manual configuration of the devices (hence the name zeroconf). It is based on the TCP/IP protocol. **[RFC-6762, RFC-6763]**

System called mDNS (multicast domain name system) is used to resolve IP addresses in small local networks without DNS server. It evolved from Apple's Apple talk and Microsoft's UPnP SSDP technology. It's now standardized and it's specified in **[RFC-6762, RFC-6763].**

The most used implementation is Apple's Bonjour, which is included in all Apple products. Libraries for other platforms/languages are available, including implementations in C, Java (compatible with Android) or C# for Windows and Windows phone.

Pros

+ Present in almost every device
+ Fast
+ Good range
+ Network devices discovery (zeroconf)

Cons

– Some form of access point is needed (router, hotspot)

5.4 Conclusion

I have chosen to use local area Wi-Fi as a communication technology for my application, because it is available on almost all mobile devices, it is easy to use and with zeroconf user will not have to bother themselves with IP addresses or other networking. The need of access point is not a big drawback today, when Wi-Fi is available almost everywhere. Bigger drawback is the need for the devices to be on local network, not allowing any way of communication over Internet, but no simple solution allowing such communication without a dedicated server is available.

6

Data storage

For my application I will need some form of persistent data storage on the device to preserve data and its structure even if the application is shut down. All the major mobile platforms offer very similar data storage options. I will shortly describe them in this chapter.

6.1 Files

On Android, iOS and Windows Phone it is possible for an application to store files in a application private folder, where they are hidden for other applications. On Android and Windows Phone 8.1 it is also possible to store files in shared file system accessible to other applications and file managers. On iOS there is no shared file system available to applications nor any file manager. **[apd, and, ms]**

Pros

+ Available on all platforms
+ Can store any data format

Cons

– Complex data structure will require complex parser to write and read the data

6.2 Key value storage

One easy data storage option available on three above mentioned systems is a key value storage. This storage is private for each application. Data are stored in a form of key – value pair. The key is unique and the value is specified as a string, boolean or number. It is very easy to use this storage, but it is not good for complex data, because there is no hierarchy or relations in the data. **[apd, and, ms]**

The following examples show how simple the manipulation with the key – value storage in C# in Windows Phone works. Figure 6.1 is an example of saving a value and on the figure 6.2 there is an example of loading the before stored value. **[ms]**

```
private void btnSave_Click(object sender, RoutedEventArgs e)
{
    IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;
    // txtInput is a TextBox defined in XAML.
    if (!settings.Contains("userData"))
    {
        settings.Add("userData", txtInput.Text);
    }
    else
    {
        settings["userData"] = txtInput.Text;
    }
    settings.Save();
}
```

Figure 6.1: Saving value

```
private void btnDisplay_Click(object sender, RoutedEventArgs e)
{
    // txtDisplay is a TextBlock defined in XAML.
    txtDisplay.Text = "USER DATA: ";
    if (IsolatedStorageSettings.ApplicationSettings.Contains("userData"))
    {
        txtDisplay.Text +=
        IsolatedStorageSettings.ApplicationSettings["userData"] as string;
    }
}
```

Figure 6.2: Loading value

Pros

+ Available on all platforms
+ Very easy to use

Cons

– Can't create any more complex data structure

6.3 SQLite database

The most complex data storage available on mobile devices is the SQLite database. It is a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is in the Public Domain and can be freely used for any purpose, including commercial use. **[sqlite]**

The database does not run in a separate process, data is written directly to disk and the whole database is stored in a single file. The database format and file is platform independent and can be easily moved between different architectures and operating systems. SQLite is good for any embedded systems where memory or resources are limited, because its requirements are minimal. It can be embedded in application as local data storage or it can be also used for server use (creating one database for each user for example). SQLite database supports any number of concurrent readers, but only one writer at any time (others have to wait in queue). **[sqlite]**

SQLite is compatible with most of the standard SQL language, but some features are not available. See **[sqlite]** for more details.

SQLite database is available in Android and iOS, for Windows Phone a library exists and can be included in an application. Another positive aspect is that a Cordova plugin exists to allow unified access to device SQLite database from JavaScript code running in a Webview.

Pros

+ Available on most platforms
+ Uses SQL to manipulate with data
+ Fast and minimal resource requirements
+ Cordova plugin for unified access from a Webview application

Cons

– Not all SQL features are available

6.4 Conclusion

I decided to use SQLite database for data storage in my application, it allows me to to store complex data with structure and work them using SQL language. I will also use files to export and import data from the database to allow their sharing.

7

ScrumApp specification

The mobile app shall support distributed software development teams, who use Scrum in planning, staying synchronized and having clear overview over the project's status. The application should offer Scrum board as it is one of the most important project overview features, backlog management of stories and tasks, scrum poker game and a burndown chart for sprints and project.

The mobile app should be multiplatform and it should work on most used mobile operating systems (Android, iOS), as companies often have fragmented portfolio of devices. Native look and feel is not required.

Distributed communication should be used, without the need to use and maintain any dedicated server. One device will act as a master, store the project data and will keep other devices synchronized. The preferred way of communication is Wi-Fi. It would be good to have some way to synchronize, when no connection is available.

7.1 Basic features

This is a list of core features which should be implemented in the application.

- Distributed communication over Wi-Fi
- Product and sprint backlog
- Story and task planning
- Burndown charts for sprints and project
- Scrum board
- Android version
- iOS version
- Phone user interface

7.2 Optional features

This is a list of optional features, not everything on this list will be implemented. This list is ordered by priority, with most important features on top.

- Scrum poker game
- Project data export and import
- Communication encryption
- Some form of backup way of synchronization if no Wi-Fi is available
- Database encryption
- Windows Phone version
- Tablet user interface
- Changes history (tasks, stories)
- Saving logs
- Localization to more languages

7.3 Technologies

I have decided to use multiplatform framework Cordova to implement the ScrumApp. When I made this decision (summer 2013) other frameworks were not as mature as they are now. Also Cordova is well documented, free, open source and more functionality can be added in form of plugins. Custom plugins can be relatively easy made, thanks to the plugin API. I will have to implement some part of the application in native code, as not every functionality I need can be made from JavaScript and suitable plugins are not available. I will be making Android and iOS version only, because other platforms have only small market share (It should be possible to add more platforms, that are supported by Cordova latter). For the multiplatform JavaScript part I chose to use Google Web Toolkit (GWT) framework with mobile GWT (mGWT) plugin for UI, that allows me to write the code in Java and then compile it using GWT compiler to JavaScript, as I know Java better than JavaScript (I will describe GWT further in the text).

I will use Wi-Fi with zero-conf for distributed communication between mobile devices. This will allow users to search for devices on local network without the need to know their IP addresses. For data storage I picked the SQLite database, it can store complex data structures and has low resources requirements. For backup of data I will allow project export and import using files.

7.4 Basic architecture overview

Basically the architecture can be divided into three main parts (figure 7.1).

- Native part containing Cordova library, Cordova plugins and other native code. This part is unique for each platform and uses Webview to display and interpret the rest of the application. This part is also responsible for handling communication and data storage.
- JavaScript part serves as a bridge between GWT and native code. It contains Cordova JavaScript library and plugins, allowing access to native code from GWT.
- GWT part contains application logic written in Java with mGWT plugin to draw mobile and touch friendly UI for the user. Code created using GWT is then compiled using GWT compiler into JavaScript, which is then deployed on the device as a part of the application and is run in the Webview with other JavaScript.

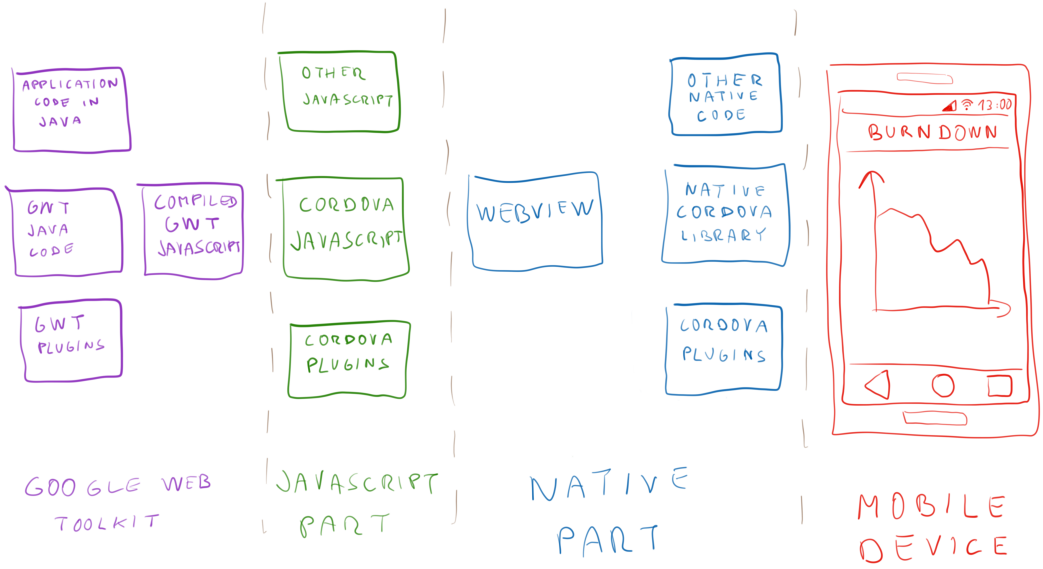


Figure 7.1: Basic architecture overview

7.5 Use cases

There are three types of users in the application: Scrum master, Product owner and Team members. Scrum master has a special role of running the master device, which takes care of synchronizing other devices. I also want to allow other users to view and modify some data, even when they can't connect to the master device. This puts some limitation on what different roles are allowed to do compared to normal client-server application, where the server is always available. I had to define and restrict some features, that would be normally possible in scrum because of this distributed communication model. Some actions like taking an unassigned task by team member requires a working connection to the master device to prevent possibility of two users taking the same task. Scrum poker is usually organized by the product owner, but to make the synchronization easier I decided, that it will be controlled by the scrum master instead (both the scrum master and the product owner are present at the estimation meeting). Below is the list of use cases available to each role, with specified if they need to be connected to master [online] or not [offline].

Scrum master (only 1 per project) – runs the server [SM]

- Create new project
- Add or edit users in project
- Estimate story - only in backlog, can be result of Scrum poker game – it should be game like (playing cards)
- Create task – only in not accepted stories
- Edit task details
- Remove task – only free (not taken tasks)
- Estimate task
- Force free task – for purpose when some problem happened with user who has taken it
- Start and end scrum poker
- Create issues

Product owner (only 1 per project) [PO]

- Edit his own user profile – [online]
- Create story in backlog – [offline]

- Accept story – when all tasks are done its marked finished and he can accept it – [offline]
- Edit story – only in backlog – [offline]
- Delete story – only in backlog – [offline]
- Create story from issue – accept the issue – [offline]
- Create issues – [offline]
- Move story from backlog to sprint – [offline]
- Create and edit next sprint – [offline]
- Remove story from sprint to backlog – story must not contain taken tasks, will remove all tasks from story – [online]
- Set next sprint as current one (automatically moves unfinished tasks and stories to next sprint) – [online]

Team members (many) [TM]

- Edit his own user profile – [online]
- Create tasks – [offline]
- Edit free or assigned tasks to him which are created by him (if edited also by SM then the SM version is used) – [offline]
- Take task – must be online to prevent more users from taking same task [online]
- Free task – only taken by him [offline]
- Close task – only taken by him, when it's finished – [offline]
- Set time worked on task – only taken by him – [offline]
- Create issues – [offline]
- Join scrum poker and vote on stories – [online]

7.6 Testing

There is possibility to use unit tests for parts of the application logic. Problem with integration testing would be the fact that huge part of functionality is provided by the device and running integration tests on the mobile device is not easy and creating a device mock is a lot of work. As a result the most important testing will be manual testing of the final application.

8

Implementation

This and several following chapters will contain description of the ScrumApp implementation. The application consist of three separate projects and Cordova JavaScript files:

1. Android native part (references in text will be **green**)
2. iOS native part (references in text will be **orange**)
3. Application logic and GUI in GWT (references in text will be **purple**)
4. Cordova plugins

The third part needs to be compiled first and the result is then embedded into both native parts. I will start with database model for storing data and distributed data synchronization, then I will follow with native parts, next will be communication protocol and last will be the actual application logic in GWT.

8.1 Multiplatform and native part

Not every functionality can be done in multiplatform way. Some things work differently on different platforms or cannot be performed from JavaScript running in Webview. For example background tasks, that have to run when application is not in foreground cannot be done using JavaScript, because Webview pauses its interpretation when it is minimalised or not focused.

Native parts

- Communication server and client
- Database (plugin for Cordova is available and works on Android and iOS and offers unified, platform independent JavaScript API)
- File export and import

Multiplatform parts

- Application GUI
- Application logic

8.2 Technology stack

As was written in the previous chapter (see 7.3) the chosen technology stack is:

- Native part for Android (Java) and iOS (Objective-C)
- Wi-Fi for communication
- MDNS (Apple bonjour) for device discovery on network
- SQLite database for data storage
- File sharing for project data export and import
- Cordova as multiplatform framework (uses HTML, CSS and JavaScript)
- Cordova plugin for unified database access from JavaScript
- GWT for application logic
- mGWT plugin for mobile touch GUI
- GWT plugin for working with database

9

Database

This chapter will be about the ScrumApp database and data synchronization. First is the overview on the database data model. Then will be described all the tables in the database and last part will be about data synchronization between the mobile devices.

9.1 Database model

On the figure 9.1 is a diagram of used database model. This database is present on all devices and project data for multiple projects can be stored in it. Database contains special table and columns for synchronization (will be described more closely in chapter 9.3).

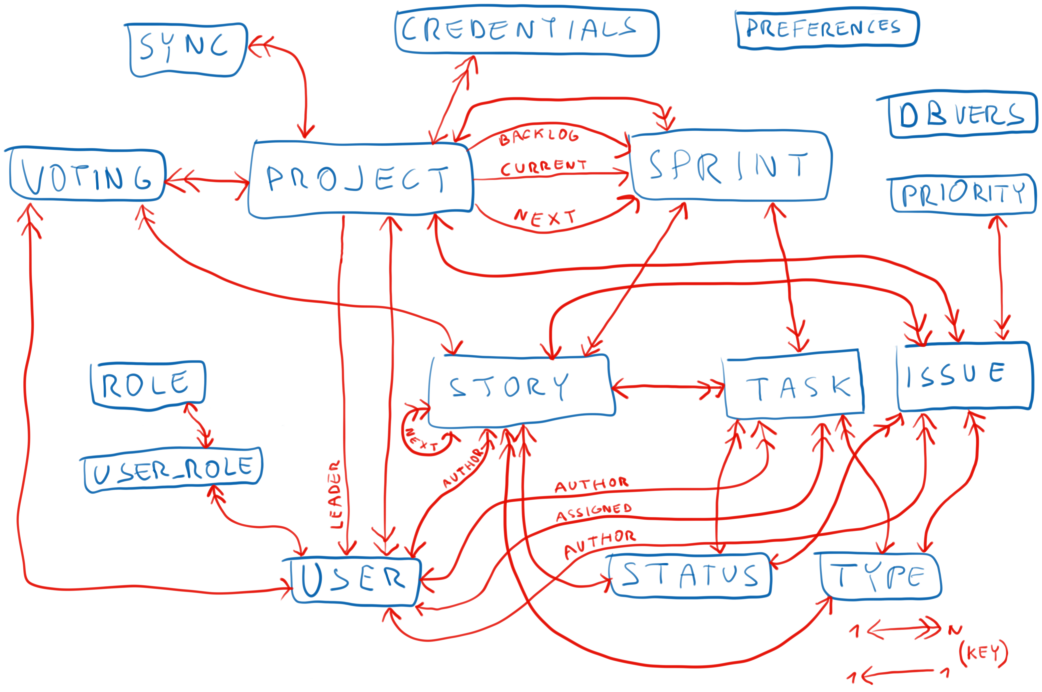


Figure 9.1: Database model

The model contains many relations and is complex, so I will try to describe each table, its content and its purpose. The database doesn't contain any triggers or procedures, because SQLite database has limited features.

9.2 Database tables

The list of all database tables and their structure can be found in **attachment 1**. Data types in SQLite are not forced, they work as a suggestion for the database and it is possible to to store a string in a integer column. Basic data types are null, integer, real, text and blob. There is no date-time like data type, dates can be stored as integers, reals or text (I store them as big unsigned integers in milliseconds). Text values are not limited by size (except global size limit). SQLite database will try to convert inserted value to defined column type automatically, if possible (this can be overridden). See **[sqlite]** for more details on data types in SQLite. If not specified, the column is a text type. All primary and foreign keys in my database are stored as big integers (64-bit).

9.2.1 Database update

Dbvers is a simple table for storing current database version. It has only one column called value and only one row with an integer number. Because the database is distributed on all devices, it also needs to be updated on all devices and the upgrade script has to be distributed with application updates. When application code in `ScrumAppDAO` class (invoked on each startup) finds that database version stored in *Dbvers* is lower than in code, it starts the update method which updates the database and increases the DB version in *Dbvers* table (figure 9.2).

```
if (old < 2)
{
  tx.executeSql("CREATE TABLE IF NOT EXISTS voting(...);", null);
  tx.executeSql("UPDATE dbvers SET value=?;",  new Object[] { 2 });
}
```

Figure 9.2: Database update

9.3 Data synchronization

Data synchronization works on one master and several slaves architecture. SQLite database doesn't support any form of distributed synchronization, so I have to implement it on my own. Because I want to allow all devices to do changes and create data, even when they are not connected to master I have already created limits on who, what and when can do database changes to prevent conflicts (see

use cases in chapter 7.5 for more details). Data exchange format between slave and master is JSON. The communication protocol and format will be closely described in chapter 12.

9.3.1 Distributed keys

In distributed environment I can't use auto-incremental keys, because they would lead to conflicts when two devices create database row with same primary key. This can be easily overcome with using GUID (Global unique identifier) also known as UUID (Universal unique identifier) specified in **[RFC-4122]**. These are randomly generated large enough numbers to minimalise chance of collision. Usually 128bit numbers are used, which provide about $5.3x10^{36}$ possible unique keys. SQLite support storing only 64bit numbers **[sqlite]**, there is possibility to store keys as strings or use only 64bit numbers with less possible unique keys (about 1.8×10^{19}), which is still enough for my use. So I decided to use only 64bit UUIDs, but it would be possible in case of problems to switch to the string version of 128bit keys, even with maintaining backwards compatibility with already generated 64bit keys. **[RFC-4122]**

9.3.2 Database synchronization

There are several scenarios in the ScrumApp:

- Slave pulls new or modified data from master since his last synchronization
- Slave pushes his offline changes to the master
- Slave asks master to update some data for him
- Master makes direct changes to data

Also not all tables require synchronization, some are only enumerations filled once and don't need to be synchronized and some tables are local only.

9.3.3 Slave pulls master for updates

This allows slave to get updated and new data from the master. A special column called **lastchange** is used. It is used on the master device to store time stamp of last change occurred in this record. This way when slave requests synchronization, he tells master last time he got update and master can send him only those records from each table that had change after the last synchronization. Slave requesting update with 0 time stamp will receive back full database.

Tables that are synchronized using this mechanism:

- Project

- Sprint
- Story
- Task
- Issue
- User
- User_role

See `UpdateProjecHandler` class, which takes care of extracting all new and updated data from database on the master. On the slave the data are stored in slaves database in `ProjectUpdateService` class.

9.3.4 Slave pushing his changes to master

Slaves use the modified column to mark records they made new or changed, they also use the Sync table to mark tables they modified so they don't have to go through whole database before sending. After the data are successfully sent to master, they set the modified column to false to mark the data as synchronized.

Five tables are synchronized using this mechanism:

- Project
- Sprint
- Story
- Task
- Issue

On slave `SendIssuesService`, `SendProjectService`, `SendStoriesService`, `SendTasksService`, `SendSprintsService` classes are responsible for sending the modified data to master.

On master see classes `IssueSyncHandler`, `ProjectSyncHandler`, `StorySyncHandler`, `TaskSyncHandler`, `SprintHandler`, as they take care of adding the slave's changes to the master's database.

9.3.5 Slave asks master to update the database for him

These are the data changes that require the slave to be connected to the master to prevent conflict in data changes. Also scrum poker game requires the users to be online to attend to the voting.

These actions use online synchronization:

- Accepting stories
- Scrum poker game
- Profile editation
- Project list of projects running on the master
- Claiming tasks by team members
- Authentication data testing (if login/password is correct)

Responsible classes on slave are `AcceptStoryService`, `ClaimTaskService`, `PokerService`, `ProfileEditService`, `ConnectionService` and classes on master `AcceptStoryHandler`, `ClaimTaskHandler`, `PokerHandler`, `ProfileEditHandler`, `ProjectListHandler`, `TestAuthHandler`.

10

Native android part

The native Android part contains both a client and server part, only Android devices can currently act as masters. The server part cannot run from JavaScript or GWT code, because JavaScript cannot be executed in background. For this reason the server code is written in native code (Java) for Android. For further references the project root package is `at.ontec.scrumapp`.

The Android project contains following parts:

- Cordova library
- SQLite Cordova plugin
- JmDNS library for Apple bonjour
- Database layer (DAO) for DB access from Android code (for running server)
- Server ScrumApp part
- Client ScrumApp part
- File import activity for importing project data
- My Cordova plugins for JMDNS, Client, Server and File export
- Multiplatform HTML, CSS and JavaScript code

The application entry point is the `ScrumApp` class, which is an Android activity. It is responsible for creating the user interface and user interaction.

"Android 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 UI and communication with the user." **[and]**

Another important class is the `MainApp` that handles application specific events and is responsible for establishing the database connection to the local SQLite database (this database is private to the ScrumApp and is stored in its private

application directory). It is important to note, that for the database correct working it is required to have only one open database handler for the SQLite database in Android, otherwise it can lead to concurrency problems. There is no problem in accessing the database concurrently from different threads, but it has to be done using the same handle across the whole application, for the locks to work properly. As I need to work with the database in the background service and also from the JavaScript using the SQLite plugin I had to store the handle in the application object so I can access it in all parts of the application. For this reason I also had to modify the SQLite plugin to use my handle from the application object instead of creating it's own.

10.1 Cordova library

I have already described the Cordova framework in chapter 3.3.1, and in 10.8 I will describe the plugins API. I recommend reading the official Cordova documentation and tutorials **[cor]** for detailed information as they are good and well maintained. Cordova offers tools for generating empty Cordova based Android project, which can then be used to create multiplatform application. I used such project as a base for ScrumApp.

Cordova library offers a **`CordovaActivity`** class, which can be subclassed to create Android activity with preconfigured WebView component in it.

My subclass is **`ScrumApp`** in project root. Other important file is **`config.xml`** located in XML resources, which contains Cordova settings and is used to register plugins for Cordova, see figure 10.1 for plugin registration example. Each referenced plugin is a subclass of **`CordovaPlugin`** class provided by Cordova.

```
<feature name="SQLitePlugin">
    <param name="android-package" value="org.pgsqlite.SQLitePlugin"/>
</feature>
```

Figure 10.1: Plugin registration

The folder for placing HTML, CSS and JavaScript files is located in **`assets/www`** directory inside the Android project. The Webview is then told what file it is supposed to load on start (figure 10.2).

```
loadUrl("file:///android_asset/www/index.html");
```

Figure 10.2: WebView entry file

10.2 SQLite Cordova plugin

I used special SQLite Cordova plugin for unified access to SQLite database from iOS and Android. See **[SQLitePlg]** for more information.

10.3 JmDNS library

JmDNS is a Java implementation of Apple bonjour (see chapter 5.3.2 for more). It is a networking library using multicast messages on local network for device discovery and advertisement. Library and documentation can be obtained at **[JmDNS]**.

I have created two small wrappers for the JmDNS library in the `jmdns` package. `JmDNSDiscovey` is for discovery and `JmDNSService` is for registration.

The service advertises itself in the network with a remote type to identify type of the service. I use remote type: `_scrum._tcp.local.` to create type that doesn't conflict with already commonly used types like `_http._tcp.local.` The service also advertises more information like IP address, Application name, operating system etc.

The discovery scans the network for such services and generates three basic events:

1. discover – new service was found on the network
2. resolve – resolves additional information about the found service
3. remove – a service sends message that it will be no longer available

In the resolve event I am able to get IP address of the master device and store it for further use.

10.4 Database DAO

The package `db.dao` contains database access objects (DAO) for manipulation with data and database transfer objects (DTO) for storing the data in memory. This mostly copies the database structure described in 9.1 and 9.2. See the actual classes for more details.

10.5 ScrumApp server

The server part is unique for the Android version of the application. The server works as very simple HTTP server with very limited features. The server runs in a separate part of the application called service.

10.5.1 Background server service

Services are Android components designed for running long term background tasks. There are several types of background services each for different purpose. Examples of services use are music playback, long downloads, file operations, VOIP calls or some network communication. A service can run in the background even if the component or activity that started it was stopped. **[and]**

Android offers predefined services for various tasks (downloads, media playback), but as I need a special one I have to implement it by myself. Foreground part of the application or even another application (if allowed) can bind to a running service and communicate with it. Like other application components as activities or content providers the service has to be declared in application `AndroidManifest.xml` file, this file contains information about application, required privileges, defined activities, content providers (data sharing with other applications) and various settings (see **[and]** for more details). On following figure 10.3 is a simple service definition, referencing the class that provides the service. The exported flag indicates, whether the service can be accessed by other applications (not allowed here).

```
<service android:name="at.ontec.scrumapp.tcpserver.ServerService" android:exported="false" />
```

Figure 10.3: Service declaration

The service is defined in `ServerService` class, important methods are:

- `onStartCommand` – this method is invoked when service is started or restarted by the system, it takes care of starting the server thread and creates a persistent notification in the notification area, so user knows it is running
- `bind` – this method allows the foreground part of application to bind to this service and communicate with it
- `getAddresses`, `getPort`, `getServerName` – allows those who are bound to this service to get some data form it, like on what IP address and port is the server running
- `onCreate`, `onDestroy` – are called by the system when the service object is created/will be destroyed to perform some initialization/cleanup

The service is started in special mode called foreground service, this tells the system, that the service is important to the user and it should not be stopped or destroyed, even if memory is low. In case the service gets killed, it is set to tell the

system to start it again when it is possible. For this kind of service it is required to display notification in the notification area, which cannot be removed, unless the service is stopped or moved from the foreground mode.

10.5.2 TCPServer

The `tcpserver` package apart from `ServerService` contains the `ServerThread` class with simple HTTP server implementation. The communication protocol will be described in the chapter 12. The server thread has two main responsibilities, it advertises its IP address and port using zeroconf (JmDNS) and it listens on a TCP socket for incoming connections. The server uses a random port assigned by operating system.

Only small amount of clients is expected to connect to the server (about 10 at max), so I did not implement any form of thread pooling for the server. For each request a new thread is created, because the request is handled in a short time, there are no cumulating long running tasks.

The communication is encrypted using AES 256bit cypher with a hard-coded key, I originally wanted to implement a possibility for users to use their own keys instead of the default one, but this feature was not implemented because there was not enough time. The encryption is handled by JNCryptor **[jnc]** (Java implentation of the RNCryptor library **[rnc]**). It uses a cross-language AES encryption/decryption format, allowing easy encrypted communication between devices with different operating systems. The hard-coded key was generated from a strong password with PBKDF2 tool (password based key derivation tool 2). See the `ServerThread` class for the actual implementation of the server loop.

There are several other files in the `tcpserver` package:

- `TCPRequest` – this is a wrapper class for the client's request
- `TCPResponse` – this is a wrapper class for the server's response
- `TCPServiceBinder` – this class allows communication with the background server service
- `TCPServiceConnection` – this class is used by the main activity to communicate with the background server service

10.5.3 Scrum server

The `scrumserver` package contains handlers for synchronization requests described in the 9.3 chapter. These classes are responsible for parsing the request, handling it and generating a response for the client.

10.6 ScrumApp Client

The client takes care of discovering the master devices in the network using the above described JmDNS library and sending requests to the server. In the `tcpclient` package simple HTTP client is implemented as the `ClientThread` class. It uses the same JNCryptor library as the server for AES encryption and decryption. It too uses the same hard-coded key.

10.7 File import and export

File import and export serves as a way to backup project data. Project data can be exported and imported as JSON files. These files can be send to someone over the Internet, using email or stored on the device's shared file system. The file import and export functionality is controlled through a Cordova plugin, that will be described later and the GUI for it is created using mGWT.

10.7.1 File import

The `FileImport` activity is a native Android activity made with native UI elements, because it takes long time to initialize a WebView and this activity is launched from other applications (like file managers, DropBox, email), when the rest of the ScrumApp is not running. File import activity is responsible for importing project from a file. This activity is registered in `AndroidManifest.xml` to accept share requests for JSON files (see the figure 10.4 below). This allows other applications to share JSON files with ScrumApp. If they contain valid project data, ScrumApp will be able to create project from them.

```
<activity android:label="@string/app_name" android:name="at.ontec.scrumapp.FileImportActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.ACTION_VIEW" />
        <action android:name="android.intent.action.ACTION_EDIT" />
        <action android:name="android.intent.action.ACTION_PICK" />
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:mimeType="text/plain"/>
        <data android:mimeType="application/json"/>
    </intent-filter>
</activity>
```

Figure 10.4: FileImport activity definition in AndroidManifest.xml

It saves the imported project file to the private application folder, where it can be accessed from the ScrumApp application. On the next figure 10.5 is the Activity UI.

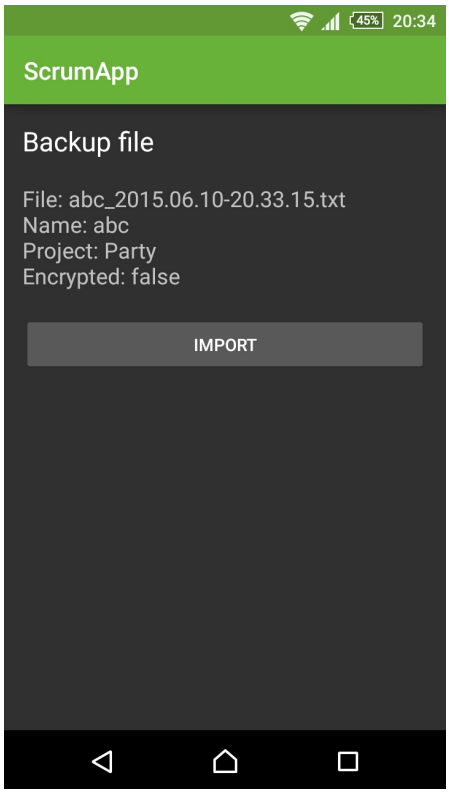


Figure 10.5: FileImport activity

10.7.2 File export

For application to be able to share files with another application, it has to implement a Content provider. Android offers some providers for common tasks (downloads, media etc.), I used the File provider (see **[and]** for more on Content providers). The File provider is defined in the `AndroidManifest.xml`. An Android Intent has to be created with information about the shared data. An Intent is a description of operation we want to perform, it can target specific Activity in this or different application. It can be general, like any Activity that can send email. Here (figure 10.6) I am using a generic request for any application that can handle the *application/json* file format. Many applications like file managers respond to any file share request and will offer saving the file on file system.

```
File file = new File(cordova.getActivity().getApplicationContext().getFilesDir(),  filename);
Uri uri = FileProvider.getUriForFile(cordova.getActivity().getApplicationContext(),
   "at.ontec.fileprovider", file);

Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
shareIntent.putExtra(Intent.EXTRA_SUBJECT, name);
shareIntent.setType("application/json");
cordova.getActivity().startActivity(Intent.createChooser(shareIntent, "Share"));
```

Figure 10.6: Starting a file sharing intent

10.8 Cordova plugins

These plugins serve as a bridge between the JavaScript Web based application part and the native code. Cordova offers a simple API for plugin creation. I have created four Android Cordova plugins for the ScrumApp, they are located in the `pg.plugins` package:

- `Files` – plugin takes care of file operations, exporting, importing and sharing project files
- `JmDNS` – plugin servers as controller for JmDNS discovery
- `TCPClient` – plugin controls sending and receiving messages using the HTTP client
- `TCPServer` – plugin controls starting and stopping the background server services

10.8.1 Plugin principle

Each plugin class inherits from **`CordovaPlugin`** class and overrides the execute method, which is called every time the plugin is used from JavaScript. On the following figure 10.7 is a part of the implementation of the execute method in `TCPServer` plugin.

```
@Override
public boolean execute(String action, JSONArray args, final CallbackContext callbackContext)
throws JSONException
{
    if (action.equals("start"))
    {
        //Starts the service
        ...
        JSONObject result = new JSONObject();
        result.put("status", "start");
        callbackContext.success(result);
        return true;
    }
    else if (action.equals("stop"))
    {
        //Stops the service
        ...
        callbackContext.success(result);
        ...
        return true;
    }
    else if (action.equals("status"))
    {
        //Reports service status.
        ...
        callbackContext.success(result);
        return true;
    }
    else if (action.equals("available"))
    {
        //Returns if server is available on this platform
        JSONObject result = new JSONObject();
        result.put("status", "available");
        result.put("message", true);
        callbackContext.success(result);
    }
    return false;
}
```

Figure 10.7: JmDNS plugin Java part

The action specifies what operation will be executed, **`args`** contain parameters passed from JavaScript in JSON Array, the callback context serves as means of passing return value back to the JavaScript code. The plugin call is asynchronous. To tell Cordova which plugins are installed, plugins are registered in **`config.xml`** file (figure 10.8).

```
<feature name="TCPServer">
    <param name="android-package" value="at.ontec.scrumapp.pg.plugins.TCPServer"/>
</feature>
```

Figure 10.8: TCPServer plugin registration

On the JavaScript side, plugin can be called with following call (figure 10.9). We have to supply two callback functions, one for success and second for error, then we specify plugin name from the **config.xml** (*TCPServer*), the required operation (*start*) and the last are arguments in an array, here only one with server name.

```
cordova.exec(successCallback, errorCallback, "TCPServer", "start", [ serverName ]);
```

Figure 10.9: Calling Cordova plugin from JavaScript

Many Cordova plugins for various use cases exists and can be downloaded. As shown above it is very simple to implement own plugins.

10.9 Multiplatform HTML, CSS and JavaScript code

The **assets/www** folder contains all the web resources used by the WebView to render the application. As I created the application using GWT, there are not many interesting files here. A simple **index.html** file used to load the GWT JavaScript, CSS file with some default style settings and simple JavaScript wrappers for the Cordova plugins. The **scrumappgwt** folder contains all the generated GWT files.

11

Native iOS part

The iOS application contains only the client part, running the server on iOS would be technically possible, but only certain tasks are allowed by Apple to run persistently in the background as described in 3.1.1.

The iOS project contains following parts:

- Cordova library
- SQLite Cordova plugin
- Client ScrumApp part
- File export and import
- My Cordova plugins for mDNS, Client and File export
- Multiplatform HTML, CSS and JavaScript code

The iOS native part is written in Objective-C (Swift was not released, when I started implementing ScrumApp). The iOS application is much simpler than the Android version, because of the missing server part.

The ScrumApp iOS application has one View called `MainViewController`, which contains the WebView used for rendering the GWT code. The `AppDelegate` class handles application events and performs initial setup. Other various application settings are stored in the `ScrumApp-Info.plist`.

11.1 Cordova library

The Cordova library works very similar on iOS and Android. Both above mentioned classes `MainViewController` and `AppDelegate` inherit most of their function from Cordova. Plugins too are registered in the `config.xml` file. The Cordova tools can generate an empty Cordova based iOS project, which I used as a base for ScrumApp.

11.2 SQLite plugin

I used special SQLite Cordova plugin for unified access to SQLite database from iOS and Android. See **[SQLitePlg]** for more information.

11.3 ScrumApp Client

The client takes care of sending requests to the server and discovering the master devices in the network using the mDNS (Apple bonjour), which is integrated into the iOS SDK and no third party library is needed.

The `ServerBrowser` class takes care of the device discovery and it offers similar functionality as the `JmDNSDiscovey` on Android, it too searches the network for the `_scrum._tcp.local.` remote service type and handles the following events:

1. discover – new service was found on the network
2. resolve – resolves additional information about the found service
3. remove – a service sends message that it will be no longer available

The `Connection` class is a simple HTTP client implementation. The client uses the *RNCryptor* library, which is fully compatible with the *JNCryptor* Java implementation in the Android application, allowing easy encrypted communication between the iOS and Android ScrumApp applications. Like in the Android version, 256bit AES encryption with the same hard-coded key is used.

11.4 File import and export

The file export on iOS is implemented using a native component for sending emails. File import is currently only possible using *iTunes*, by copying the project file to the application documents directory, where it can be loaded. Exported projects can backed up by this way too. Document sharing via *iTunes* can be enabled easily by adding a key `UIFileSharingEnabled` with value `true` to the `ScrumApp-Info.plist` file.

The email composer component is launched from the Files plugin. The following code (figure 11.1) is responsible for starting the composer as a modal View.

```
MFMailComposeViewController* controller = [[MFMailComposeViewController alloc] init];
  controller.mailComposeDelegate = self;
  [controller setSubject:name];
  [controller addAttachmentData:exportFileData mimeType:@"application/json" fileName:filename];
  if (controller) {
    [ ((AppDelegate*) self.appDelegate).viewController
    presentModalViewController:controller
    animated:YES];
  }
```

Figure 11.1: Starting mail composer in Objective-C

An email template will be created for the user with subject and attachment.

11.5 Cordova plugins

These plugins serve as a bridge between the JavaScript Web based application part and the native code. Cordova offers a simple API for plugin creation. I have created four iOS Cordova plugins for the ScrumApp.

- **`FilesPlugin`** – plugin takes care of file operations, exporting, importing and sharing project files
- **`MDNSPlugin`** – plugin serves as controller for MDNS discovery
- **`TCPClientPlugin`** – plugin controls sending and receiving messages using the HTTP client
- **`TCPServerPlugin`** – TCP server is not implemented on iOS and the only purpose of this plugin is to tell the JavaScript part, that it is not available.

11.5.1 Plugin principle

Plugins on iOS work very similar as on Android. Each plugin is a subclass of the CDVPlugin class. The plugin defines available operations in it's interface (figure 11.2)

```
#import <Cordova/CDVPlugin.h>
#import <MessageUI/MFMailComposeViewController.h>
#import "AppDelegate.h"

@interface FilesPlugin : CDVPlugin <MFMailComposeViewControllerDelegate>
- (void) filesExport:(CDVInvokedUrlCommand*)command;
- (void) filesShare:(CDVInvokedUrlCommand*)command;
- (void) filesFiles:(CDVInvokedUrlCommand*)command;
- (void) filesFile:(CDVInvokedUrlCommand*)command;
- (void) filesDelete:(CDVInvokedUrlCommand*)command;
- (void) filesImport:(CDVInvokedUrlCommand*)command;
@end
```

Figure 11.2: Plugin interface in header file

On the figure 11.3 below is a shortened implementation of one of the operation from the FilesPlugin class. As on Android, the plugin receives an array of arguments from the calling JavaScript and then uses the callback object to pass back results. Calling plugins in iOS is asynchronous too.

```
- (void) filesDelete:(CDVInvokedUrlCommand*)command
{
    @try
    {
        NSArray* arg = command.arguments;
        NSString* filename = arg[0];
        ...
        NSDictionary* result = @{
                                  @"status" : @"delete",
                                  @"delete" : success ? @"true" : @"false"
                                  };
        NSLog(@"Delete file %@", result);
        CDVPluginResult *pluginResult = [ CDVPluginResult
                                          resultWithStatus    : CDVCommandStatus_OK
                                          messageAsDictionary : result
                                          ];
        [self.commandDelegate sendPluginResult:pluginResult callbackId:command.callbackId];
    }
    @catch (NSException *exception)
    {
        NSLog(@"Delete file error %@", exception);
        NSDictionary *jsonObj = @{
                                   @"status" : @"error",
                                   @"message" : @"Delete file failed"
                                   };
        CDVPluginResult *pluginResult = [ CDVPluginResult
                                          resultWithStatus    : CDVCommandStatus_ERROR
                                          messageAsDictionary : jsonObj
                                          ];
        [self.commandDelegate sendPluginResult:pluginResult callbackId:command.callbackId];
    }
}
```

Figure 11.3: Plugin operation implementation

Plugins are registered in the **config.xml** file like on android (figure 11.4).

```
<feature name="SQLitePlugin">
    <param name="ios-package" value="SQLitePlugin"/>
</feature>
```

Figure 11.4: Plugin registration on iOS

The JavaScript part of plugins is exactly same like on android and is shared by both implementations.

11.6 Multiplatform HTML, CSS and JavaScript code

The **www** folder contains all the web resources used by the WebView to render the application. As I created the application using GWT, there are not many interesting files here. A simple **index.html** file used to load the GWT JavaScript, CSS file with some default style settings and simple JavaScript wrappers for the Cordova plugins. The **scrumapp** folder contains all the generated GWT files.

12

Communication protocol

The communication between the client (slave device) and server (master device) is done using REST architecture. A simple HTTP server runs on the master and the slaves use a simple HTTP client to communicate with the server. Communication is always initiated by the client. MDNS is used to obtain IP address of the server device by clients.

HTTP request

All of the requests from clients are using the HTTP post method and message payload is sent in JSON format. The HTTP header contains an authorization field with user login and password. Both the payload and the authorization string are encrypted. On the following figure 12.1 is the request format.

```
POST / HTTP/1.1
Authorization: login:password     //encrypted
Content-Type: application/octet-stream
Content-Length: payload size
Host: address:port

{ encrypted JSON payload }
```

Figure 12.1: HTTP request format

HTTP response

The response format from server is very similar (figure 12.2). The server does not communicate in unencrypted format and is not designed to be used by any third party clients.

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: payload size

{ encrypted JSON payload }
```

Figure 12.2: HTTP response format

12.1 Server method list

As can be seen in the `ScrumServerHandler` class, there are several methods available on the ScrumApp server. Each request to the server contains a JSON object as payload with field **`cmd`**, which identifies the requested operation. The response is usually a JSON object or a JSON array.

List of methods:

- **`get projects`** – returns list of available projects on the server
- **`test auth`** – tests if login and password are correct
- **`update project`** – pulls new project data since last synchronization or whole project
- **`sync issues`** – pushes client offline changes of issues to the server
- **`sync stories`** – pushes client offline changes of stories to the server
- **`sync project`** – pushes client offline changes of project to the server
- **`sync tasks`** – pushes client offline changes of tasks to the server
- **`sync sprints`** – pushes client offline changes of sprints to the server
- **`edit profile`** – sends changes after client profile editation
- **`claim task`** – lets the team member take an open task from scrum board
- **`accept story`** – lets product owner accept a finished story
- **`vote`** – sends a client vote in scrum poker
- **`get votes`** – returns list of current votes

12.2 Message content examples

Here I will show few communication examples. I will show the payload decrypted, as it wouldn't be interesting to show the encrypted bytes. The content length stands for the encrypted version of the message.

12.2.1 Project list

First example shows client requesting list of available project from a server.

Request

```
POST / HTTP/1.1
Authorization: login:password
Content-Type: application/octet-stream
Content-Length: 82
Host: 192.168.0.19:41219

{"cmd":"get projects"}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 274

[
  {"id":"6815474812","name":"Party","description":"Test project"},
  {"id":"2550179721","name":"Test scrum","description":"abc"},
  {"id":"6891572548","name":"New project","description":"My new project"}
]
```

12.2.2 Vote

This example shows voting in scrum poker game.

Request

```
POST / HTTP/1.1
Authorization: login:password
Content-Type: application/octet-stream
Content-Length: 146
Host: 192.168.0.19:41219

{"cmd":"vote","project":"2550179721","story":"9021323674","user":"125958003","vote":"13"}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 66

{"vote":"13.0"}
```

12.2.3 Sync issues

Last example shows client sending a new issue to the server.

Request

```
POST / HTTP/1.1
Authorization: login:password
Content-Type: application/octet-stream
Content-Length: 386
Host: 192.168.0.19:41219

{
"cmd":"sync issues",
"project":"2550179721",
"issues":
 [{
  "id":"6469405778",
  "name":"New issue",
  "description":"This is very important",
  "created":"1434213464642",
  "closed":null,
  "priority":"2",
  "lastchange":"1434213464644",
  "author":"125958003",
  "type":"11",
  "status":"21",
  "project":"2550179721",
  "story":null,
  "comment":null,
  "modified":"true"
 }]
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 66

{"synced":1}
```

13

GWT multiplatform part

GWT (Google Web toolkit) is a set of open source (Apache license) tools for web front-end development in Java.

From the official documentation:

"GWT is a development toolkit for building and optimizing complex browser-based applications. Its goal is to enable productive development of high-performance web applications without the developer having to be an expert in browser quirks, XMLHttpRequest, and JavaScript. GWT is used by many products at Google, including AdWords, AdSense, Flights, Hotel Finder, Offers, Wallet, Blogger. It's open source, completely free, and used by thousands of developers around the world." **[GWT]**

13.1 GWT overview

The GWT allows developers to write, test and debug the application in Java and then using the GWT Java to JavaScript compiler, the Java code is translated to optionally obfuscated and optimized JavaScript. GWT contains JavaScript implementation of many standard Java classes (for example, **`java.lang`** and a part of **`java.util`** packages). **[GWT]**

For more detailed information on how GWT framework works see **[GWT]**, the official documentation is good and well maintained.

13.1.1 Developing using GWT

GWT code can be debugged in the Chrome browser using a GWT development plugin. The application code is run on local Java virtual machine, instead of compiling it to JavaScript. This allows easy development without having to compile the code after every change, the debugger supports code hot swap. **[GWT]**

13.1.2 JSNI

GWT Java code is translated to JavaScript, this allows for combining the Java code with JavaScript and connect the application to existing JavaScript code. The JSNI (JavaScript native interface) allows this. It even allows using Java objects from GWT in the JavaScript code. The JSNI methods use a keyword native and the JavaScript implementation of the method is written in Java comment (figure 13.1). **[GWT]**

```
public static native void alert(String msg) /*-{
  $wnd.alert(msg);
}-*/;
```

Figure 13.1: GWT native method

Calling Java methods and using Java objects from JavaScript (figure 13.2) use similar syntax as calling them from C in the JNI (Java native interface). **[GWT]**

```
[instance-expr.]@class-name::method-name(param-signature)(arguments)
```

Figure 13.2: Java call

The figure 13.3 below shows real example of calling Java code from JavaScript. The callback object is a Java class and its **`onSuccess`** and **`onFailure`** methods are used. The **`jsonObject`** created in the JavaScript is a Java object too. The **`setCallback`** method serves for setting a callback on the MDNS Cordova plugin I created, which is invoked when MDNS fires an event.

```
public static native void setCallback(Callback callback)
/*-{
  if (typeof $wnd.mdns === 'undefined') return;
  var successCallbackfunc = function(data)
  {
    var jsonObject = @com.google.gwt.json.client.JSONObject::new(
      Lcom/google/gwt/core/client/JavaScriptObject;)(data);
    callback.@com.ontec.scrumapp.client.jsni.Callback::onSuccess(
      Lcom/google/gwt/json/client/JSONObject;)(jsonObject);
  };
  $wnd.mdns.setSuccessCallback(successCallbackfunc);

  var errorCallbackfunc = function(data)
  {
    var jsonObject = @com.google.gwt.json.client.JSONObject::new(
      Lcom/google/gwt/core/client/JavaScriptObject;)(data);
    callback.@com.ontec.scrumapp.client.jsni.Callback::onFailure(
      Lcom/google/gwt/json/client/JSONObject;)(jsonObject);
  };
  $wnd.mdns.setErrorCallback(errorCallbackfunc);
}-*/;
```

Figure 13.3: JSNI example

13.1.3 mGWT

I am using a GWT plugin called mGWT (mobile GWT) for building the application GUI. This plugin contains classes, tools, CSS and JavaScript resources aimed at creating mobile, touch friendly, animated user interfaces. The plugin is open source, distributed under the Apache license. It offers universal theme and platform themes for iOS and Android (figure 13.4). It is developed by Daniel Kurka and the current version is 2.0. **[mGWT]**

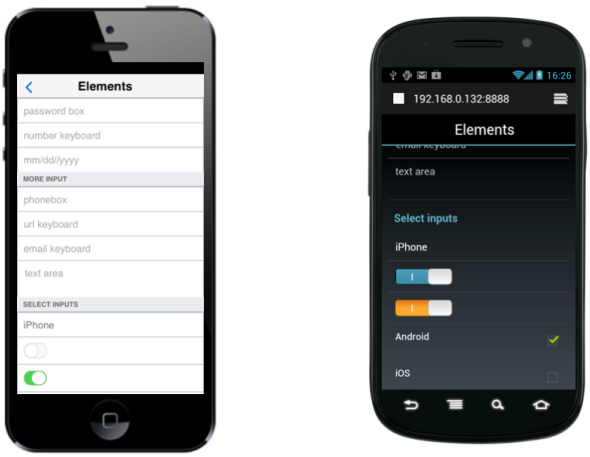


Figure 13.4: mGWT platform themes

I am using older version of mGWT (1.1.2). I started development in the late summer 2013 and the mGWT project did not get any updates for a long time (over a year) and it looked abandoned, there were bugs, that seemed no one is going to fix, so I made several workarounds for them. When the new 2.0 version was released, I tried updating my project, but many things were completely reworked and were not backwards compatible and I would have to rewrite too many things. I did not have time to do so and decided to leave the project with the patched old version. In the old version the universal and Android theme was not so good and did not work correctly, so I decided to use the iOS 6 theme for the whole application, as it was the best looking and least buggy option I had (and I did not had to fix two themes). The mGWT contains common UI elements like text fields, sliders, buttons etc. which are all rendered using HTML, JavaScript and CSS.

13.1.4 GWT PhoneGap

GWT PhoneGap is a GWT plugin allowing easy use of Cordova JavaScript APIs from GWT. It is implemented as JSNI wrapper around the JavaScript API. By using this plugin I did not had to write so many JSNI wrappers, only for my own Cordova plugins. **[mGWT, gwt-pg]**

13.2 ScrumApp GWT

The ScrumApp contains many classes, I will not try to write about everything, I will rather describe some functional units. Default package with my sources is `com.ontec.scrumapp`. The application sources consists of following parts:

- `com.google.code.database` package with database plugin
- `com.googlecode.mgwt` package with patched mGWT sources
- `client` package with entry points and various helper classes
- `client.activities` package with application activities and views
- `client.connection` package with classes taking care of parsing requests and creating responses
- `client.css` package with some custom css settings
- `client.dao` package with database layer
- `client.files` package with classes handling project export and import
- `client.jsni` package with JSNI wrappers
- `client.resources` package with application resources
- `client.ui` package with custom UI components
- `client.util` package with utility classes

And two additional libraries:

- `gwtphonegap` is a Cordova JavaScript API wrapper
- `gwt-crypto` contains encryption utilities

The base package contains `ScrumAppMGWT.gwt.xml` file, where are defined used GWT modules (plugins) and the application entry point class.

13.2.1 Database plugin

I used a GWT plugin from the *gwt-mobile-webkit* project **[gwt-db]**, that allows using SQLite database embedded in the browser. I modified the sources a bit so the plugin connected to the SQLite Cordova API instead (when available) on the device using the SQLite database embedded in the mobile device (it has better performance and is not limited in size by the browser settings). For testing, development and debugging in browser I used the Chrome's embedded database.

13.2.2 Client

The client package contains application entry point and various helper classes:

- `MgwtAppEntryPoint` is an application entry point class, it takes care of initializing the application and performing startup setup. Additionally it initializes connection to Cordova API.
- `ClientFactory` and it's implementation is responsible for creating Views (page/screen UI) when they are first needed and then stores them for further need, so they are not created again (the operation is resource heavy). Any other class that needs to obtain a View do it through this factory class.
- `AppHistoryObserver` class takes care of rebuilding the View history hierarchy, when application is started on other than the root View. It pushes all the needed Views to the history, so when user navigates back, he gets to correct previous pages. It also registers to the WebView back action (browser back button), for example when Android back button is triggered.
- `AppPlaceHistoryMapper` contains registrations of *tokenizers*, they take care of converting string URL parameters to Place objects and vice versa (more about places later).
- `PhoneActivityMapper` creates a new Activity (page controller/logic) from a Place object and passes the correct arguments to it from the Place object.
- `PhoneAnimationMapper` defines transition animations between pages.
- Tablet mappers are not implemented and could be used to create a tablet layout (two columns layout in landscape for example).

13.2.3 Connection

The `client.connection` package contains classes for communication with the master device, pulling data from server and pushing changes back. The implementation tries to contact last known IP address and port of the master device (stored in project table for each project), if the communication fails it starts the mDNS discovery and searches for the master on the network, if master is found on new address, it is stored to the database, otherwise the communication attempt fails (there are few retries and a timeout).

Important classes are:

- `DeviceDiscovery` is a class controlling the mDNS discovery, anyone can register a callback and receive notifications on discovered or removed services from network, it uses my Cordova plugin for mDNS. It automatically starts the mDNS when first callback is added and stops it, when last callback is removed.
- `MessageDispatcher` is a class that handles sending and receiving messages over TCP using my Cordova **`TCPClient`** plugin.
- `ServerStatus` is a class that allows reading status of the local TCP server service if it is running and on what address and port.
- `services.ConnectionService` is a class that adds a convenience layer on the communication. It uses the `MessageDispatcher` and `DeviceDiscovery` classes to handle sending messages. The sender does not have to know IP and port of the master, only the project, message and credentials have to be provided. The `ConnectionService` automatically tries to find the master on last known address or starts discovery to search the network. Sender does not have to care about what needs to be done to deliver the message. If max retry count is reached, sender is notified via callback about delivery failure, otherwise he receives a server response.

Synchronization

In the `client.connection.services.sync` package are classes for using the available methods of the server REST API. Those classes provide client side of the API and copy the structure of the methods of **`TCPServer`** handlers described in 12.1.

13.2.4 DAO

The package `client.dao` contains database access objects (DAO) for manipulation with data and database transfer objects (DTO) for storing the data in memory. This mostly copies the database structure described in 9.1 and 9.2. See the actual classes for more details.

The most interesting part is the `ScrumAppDAO` class, this class creates the database, if it does not exists, handles database structure updates and provides database handle for the rest of the application.

13.2.5 Files

The files package contains a `Backup` class that handles exporting project data to JSON and importing the from JSON. It communicates with my Cordova Files plugin to store and load the data from the mobile device.

13.2.6 JSNI

The package `client.jsni` contains JSNI wrappers for my Cordova plugins.

13.2.7 Resources

The resources package `client.resources` contains application resources, localized strings (only English version exists), image resources and some CSS resources.

13.2.8 UI

The `client.ui` package contains a custom *TabBar* buttons for the scrum board and a `ValidationForm`, which is a convenience class for easy creation of forms with user input validation, as forms are widely used in the application.

13.2.9 Util

The `client.util` package contains various utility classes:

- `Crypto` contains methods for calculating Hashes (SHA1 and MD5).
- `Estimation` contains constants for the estimation and scrum poker game.
- `IDPool` contains methods for generating random UUIDs.
- `SyncTimer` is a class with synchronization timer, that takes care of starting synchronization with master periodically. The synchronization can be scheduled with interval in milliseconds and is paused, when application is not in foreground.
- `Vector` is a class representing a 2D vector and operations with vectors. It is used for graph painting.

13.2.10 Activities

The `client.activities` package contains individual pages (screens) of the ScrumApp and some common functionality for them.

- `BasicActivity` is base class for activities, contains shared functionality of all activities. Registers browser actions (history events) and refresh, resume and pause events.

- **`BasicView`** and **`BasicViewImpl`** contain shared elements: Header, footer, back button, action button, scroll panel with layout panel for content.

Each page consists of three classes and one interface:

- **`View`** is an interface for the **`ViewImpl`** class, there can be more View implementations, for example one with GUI for phones and second for tablets.
- **`ViewImpl`** class defines the user interface for the page and allows the activity to display content in it or retrieve user's input.
- **`Place`** allows an activity to be accessed via URL, it provides *PlaceTokenizer* which allows serialization and deserialization of places to and from URL .
- **`Activity`** contains the page's functionality.

13.2.11 Pages

Here is a list of individual pages in the ScrumApp (in the **`client.activities`** package).

authentication

This page is for setting and updating credentials for a project. It is started, when user starts connecting to a new remote project. It tests provided credentials using the server's **`test auth`** method. If provided project credentials it starts the project data update afterwards.

backlog

This page is for displaying the product backlog as a list of stories. It allows product owner to add, edit and remove stories, reorder the backlog and move stories to sprint backlog. Scrum master can start the scrum poker for a story from here.

backlog.edit

This page contains a form for creating and editing product backlog stories.

editprofile

This page lets user to change his profile picture, it updates the profile settings on the server immediately (or fails if connection to server is not available).

estimation.picker

This page serves for setting or changing the estimation for a story (without using the scrum poker).

export

This page handles exporting a project from a project list to a file using Cordova files plugin.

files

This page shows a list of available project files in application private folder using Cordova files plugin.

fileimport

This page is for importing a project from a file using Cordova files plugin.

issues

This page is for displaying a list of issues and allows the product owner to manipulate them. Anyone can create an issue on this page.

issues.close

This page allows the product owner to close an issue, refuse or accept it (and create a story from it).

issues.create

This page contains a form for creating issues.

menu

This page servers as an entry point of the ScrumApp, it is a dashboard containing overview of relevant information about currently selected project and links to other parts of application. ScrumApp server can be started or stopped.

project

This page shows list of local (running from this device), remembered (was connected to, has local copy) and available projects (never connected to, no local copy). A local or remote project can be set as current project. Also allows creating new projects.

project.create

This page contains form for new project creation.

project.edit

This page contains form for editing project settings.

project.edit.users

This page shows list of users of a project. Scrum master can add, edit or deactivate users here.

project.edit.users.edit

This page contains form for creating or editing user by scrum master.

projectgraph

This page contains HTML canvas and draws project burndown graph on it.

scrumboard

This activity shows project scrum board. It has a tabbar component with three tabs (open, work and done tasks) and allows manipulation with tasks.

Scrumboard.edittask

This page contains form for creating or editing tasks.

scrumpoker

This page is for the scrum poker game, it shows a story that is being estimated and a list of users (from project) and their votes (hidden until voting is finished), each user can vote for the story. The scrum master can end the voting and show the voting results to everyone.

sprint.current

This page contains form for creating or editing current sprint.

sprint.graph

This page contains HTML canvas and draws sprint burndown graph on it.

sprint.next

This page contains form for creating or editing next sprint.

sprintbacklog

This page is for displaying the sprint backlog as a list of stories. It allows product owner to add, edit and remove stories, reorder the backlog and move stories back to product backlog, he can also accept stories when they are finished. Scrum master can start the scrum poker for a story from here.

sprintbacklog.accept

This page contains form, where product owner can comment on story when he accepts it.

sprintbacklog.edit

This page contains a form for creating and editing sprint backlog stories.

sprinthistory

This page shows a list of finished sprints, for each finished sprint details can be viewed (list of stories, burndown chart).

sprinthistory.sprintdetail

This page shows a list of stories for a finished sprint in the sprint history.

transferstory

This page allows product owner to select a new position for a story, when he moves it between sprints.

14

Tests

From the possible methods of testing I decided to use manual test scenarios as they suit my application best. A lot of application functionality rely on access to database or networking, for unit test it would be required to create mock of the database and networking, this would be very complex and time consuming. Maybe if the application was designed with this in mind from start, but as I was using and exploring the technology used in this application for first time, I had to made a lot changes in the code. Maintaining tests with heavily changing code is time consuming. Using any form of integration tests to test the distributed communication is a problem, as I did not find any suitable testing framework for this. There is a possibility to test Webview based applications using Selenium framework on real devices, but it is hard to make it work and it does not offer any easy way to test synchronization of multiple devices at the same time. In the **attachment 2** are described used testing scenarios. The ScrumApp did not get any form of extensive user testing yet. Application was tested on following devices:

- **Nvidia Shield Tablet** (Android 5.0 and 5.1) – everything works, UI is fast and smooth
- **Sony Xperia Z3 Compact** (Android 4.4 and 5.0) – everything works, UI is fast and smooth
- **Xiaomi MiPad** (Android 4.4) – everything works, but UI lags and is slow (this tablet has issues with all WebView based applications, despite having powerful hardware)
- **Samsung Galaxy Nexus** (Android 4.3) – everything works, UI is not smooth, as this device is older and has weaker hardware
- **Apple iPad mini 1st generation** (iOS 7) – everything works, UI is fast and smooth (despite this iPad's old hardware, WebView applications are well optimized on iOS)

15

Results

I created a multiplatform mobile application called ScrumApp using the Cordova framework. It has all the required features specified in the specification and several optional features are implemented. The application works on Android and iOS, only the Android version has the server functionality (due to iOS limitations). I haven't published the application on the Google play and Apple AppStore yet, one reason is, that paid account is required (which I don't have) and the second is, it would require me to further develop, maintain, provide bug fixes and support the application and I currently don't have enough time to do it. In the **attachment 3** are some screenshots of the ScrumApp.

15.1 What is done

All the basic features are implemented in some way:

- Distributed communication over Wi-Fi – works correctly, users don't have to deal with IP addresses, as they are automatically resolved using the mDNS.
- Product and sprint backlog – product and sprint backlog are implemented, they support story adding, editing and ordering. Stories can be moved between product and sprint backlog.
- Story and task planning – stories can be planed in the backlog, for estimation is implemented the scrum poker game. For task planning is the scrum board.
- Burndown charts for sprints and project – both are implemented
- Scrum board – scrum board is implemented and three task states are possible (open, work, done)
- Android version – The Android version has all implemented features.

- iOS version – The iOS version has all implemented features except the server part.
- Phone user interface – The phone user interface is created using GWT with mGWT plugin and has iOS 6 visual theme.

These optional features are implemented:

- Scrum poker game – the scrum poker game is fully implemented and can be used to estimate stories, it uses distributed communication allowing several mobile devices to participate in the voting process.
- Project data export and import – projects can be exported and imported from files. These files can be send by email and exported/imported by iTunes on iOS or shared to or from any application that supports JSON files on Android.
- Communication encryption – Network communication is encrypted using 264 bit AES encryption, only drawback is the hard coded encryption key, that cannot be changed.
- Some form of backup way of synchronization if no Wi-Fi is available – the file export/import can be used as a backup way of synchronization, but it is not very convenient way.

15.2 What isn't done

Some of the optional features were not implemented, mostly to lack of time and technical difficulties:

- Database encryption – there are several libraries for SQLite database encryption on iOS and Android, the most promising was SQLCipher **[sql-c]**, which should work with the used Cordova SQLite plugin, but I did not manage to get it working in reasonable amount of time. The use with Cordova is not well documented, and second problem is that I am accessing the database also from native code on Android.
- Windows Phone version – the WP version was not implemented, first problem was, when I started development the WP 7 did not have any support for Apple bonjour (mDNS), second problem is that used mGWT version supports only WebKit based browsers and WebView on WP uses trident from Internet explorer. It would also take a lot of time to create the native WP part.
- Tablet user interface – only lack of time prevented the tablet user interface, there are no technical obstacles for further implementation.

- Changes history (tasks, stories) – not implemented due to lack of time.
- Saving logs – not implemented due to lack of time.
- Localization to more languages – not implemented due to lack of time.

15.3 Encountered problems

The biggest problem was the mGWT plugin for GWT. As mentioned before I am using older version of this plugin, because updating to newer version would take too much effort and many things would have to be rewritten to work with the new API. This older version did not get updated for a long time and contains many bugs. When I was writing this application I had to fix or workaround several problems mostly broken CSS rules (not working with longer texts, enlarging buttons instead of centering them in parent container, and some more), or customizing the TabBar component as it lacked needed features (custom icon buttons) and the iOS 6 themed version did not worked correctly on Android and had to be fixed.

Another small problem was the bureaucracy needed to be able to develop and deploy iOS applications on real device. Launching applications from Xcode in simulator is simple, but to deploy them on real device, the device has to be registered in developer account, certificates for application has to be created and then imported to Xcode. This process is very chaotic for anyone who haven't seen it before (it is actually simpler to hack and unlock the device to accept unsigned applications).

When I was writing the GWT code I encountered a problem called callback hell. This problem came to GWT and it's plugins from JavaScript, where asynchronous calls, that return result in a callback are very common. It is caused by nesting callbacks into another callbacks (see simplified example in GWT Java on figure 15.1).

```
AsyncFunction1(...) {
  @Override
  public void onResult(Result result) {
    asyncFunction2(...) {
      @Override
      public void onResult(Result result) {
        asyncFunction3(...) {
          @Override
          public void onResult(Result result) {
            …
          });
        }
      });
    }
  });
}
```

Figure 15.1: Callback hell

GWT uses Java, but as it is closely connected with JavaScript, this problem occurs here too and the resulting code looks even uglier than in JavaScript (thanks to the anonymous classes wrapping the callbacks). I must confess, that I did not manage to resolve this problem in any way and on many places in the GWT part there are nested callbacks and the resulting code is not very readable, not mentioning that this code is prone to errors and very hard to test. But it is very hard to avoid this problem, as you can't avoid using the callbacks.

15.4 Supported devices

The ScrumApp requires an Android device with version 2.3 and above or an iOS device with iOS 7 and above. Wi-Fi is needed for communication, but it is present in basically all modern smarphones and tablets.

15.5 Performance

WebView application are more resource demanding than native applications, more powerful hardware is needed to run them. Also very important is the optimization and performance of the used WebView, which is provided by the device and is not part of ScrumApp. iOS has had very well optimized WebView for a long time and this application can run smoothly even on older iOS devices. The situation on Android is more complicated, as many device manufacturers provide their own WebView implementations, the performance can vary vastly between different devices. Android devices with version 2.3 and lower don't use hardware acceleration for WebView resulting in very poor performance (but this Android version is old and there are not many devices with it today). From version 4.0 to 4.3 the WebView uses a hardware acceleration, but it's performance and optimization is not so good (varies a lot between different devices). Version 4.4 received many WebView optimizations (WebView is based on Google Chrome code) and runs much better (except on the Xiaomi MiPad I used for testing). From version 5 and above the WebView is an independent system component and is updated like a normal Android application through the Google play, performance was also greatly improved.

Support for newer technologies like HTML5 in WebViews is very good and the situation is much better than on desktop, as mobile devices are more often replaced and there is no need to optimize the code for older browsers (like older versions of Internet explorer), also except Windows phone all mayor platforms use WebKit as a base for WebViews, making optimization easier.

15.6 Battery usage

Using the client side of the application has no special impact on battery compared to other applications, all the networking resources are cleaned and connections are closed when application leaves foreground and is paused.

The server part has a small to moderate impact on battery life as it prevents the device from going into sleep. It depends on the battery size, larger devices like tablets will be affected much less by the background running service. When the device is awake and being used, the added energy consumption is very low, compared for example to the device's screen or cellular connection.

15.7 Software used for development

The iOS part of application was developed using Xcode on OS X. For Android development I used mainly Eclipse with ADT (Android development tools) plugin, later, when it was released I switched to Android studio (based on IntelliJ IDEA). The GWT part was created in Eclipse with GWT plugin and development tools. Base empty Cordova projects for both native parts were generated with Cordova tools.

This text was written in LibreOffice Writer and most of the diagrams are drawn by me on my Android tablet in application ArtFlow.

15.8 Statistic

The ScrumApp consists of three projects:

- **iOS native part** is made of 9 classes made by me and it has one View containing the WebView
- **Android native part** is made of 50 classes (16 are DAO and DTO objects), it has two activities one for the WebView and second for file import
- **GWT multiplatform par**t contains big amount of classes, because every page consists of 3 classes and one interface (there are 29 pages), then there are DAO classes and many callback interfaces, so it very hard to count any relevant number here.

16

Conclusion

In this work I explored existing mobile solutions for project management using scrum. I compared and described available mobile multiplatform development frameworks. I designed a model for distributed communication between mobile devices. I created a specification with basic and optional features for a mobile scrum management application.

I have implemented a multiplatform mobile application **ScrumApp**, all basic features from the specification are implemented and about a half of optional features as well. I have chosen the Cordova framework for the multiplatform part, because when I started implementing this application it worked the best from the available solutions. It is open source and free (even for commercial use). It enables creating multiplatform applications with standard web technologies like HTML, CSS and JavaScript. It is also well documented and supported. For the GUI and application logic I used Google Web Toolkit, which can be used to create web front-end applications in Java. I made this choice because I know Java better than JavaScript and it can be used to create mobile and touch friendly web applications.

The created application uses distributed communication on LAN using Wi-FI and no dedicated server is required. One of the devices acts as a master device (currently only in Android) and other devices will use it to synchronize. The Apple bonjour technology was used for device discovery on LAN. The project data are stored on the master device in SQLite database and all client devices have a local copy for offline access. Reading and even some data modifications are allowed on clients and are synchronized to the master when it is available. Interactive card like scrum poker game is also implemented. The application runs on Android and iOS.

For future improvements, the remaining optional features can be implemented. The application could also use some refactoring, as there is some old unused code and the code structure could be simplified. The application did not undergo any extensive testing and any form of user testing would help greatly to discover bugs and errors.

References

[esc] RUBIN, Kenneth S. *Essential Scrum: a practical guide to the most popular agile process*. Upper Saddle River, NJ: Addison-Wesley, 2012, xliii, 452 p. ISBN 0137043295.

[aep] SCHWABER, Ken a Mike BEEDLE. *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall, 2002, xvi, 158 p. ISBN 0130676349.

[ftb] DINWIDDIE, George. Feel the Burn: Getting the Most Out of Burn Charts. *Better Software Magazine: Software Practices, Technologies, and Tools* [online]. 2009, 2009(5) [accessed 2015-05-22]. Available: http://idiacomputing.com/pub/BetterSoftware-BurnCharts.pdf

[grt] Gartner. *Gartner Says Sales of Smartphones Grew 20 Percent in Third Quarter of 2014* [online]. 2014, 15.12.2014 [accessed 2015-04-17]. Available: http://www.gartner.com/newsroom/id/2944819

[nap] Net applications. *Net applications mobile OS usage statistics* [online]. 2014, August 2014 [accessed 2015-04-17]. Available: http://www.netapplications.com

[scg] StatCounter Global. *StatCounter Global mobile OS usage statistics* [online]. 2014, August 2014 [accessed 2015-04-17]. Available: http://www.netapplications.com

[sbr] Statistic Brain. *Mobile Phone App Store Statistics* [online]. 2015, March 2015 [accessed 2015-04-17]. Available: http://www.statisticbrain.com/mobile-phone-app-store-statistics

[apd] Apple Inc. *Apple developer* [online]. 2015, April 2015 [accessed 2015-04-21]. Available: https://developer.apple.com/

[and] Google Inc. *Android developer* [online]. 2015, April 2015 [accessed 2015-04-03]. Available: http://developer.android.com/guide/index.html

[ms] Microsoft. *MSDN - Microsoft Developer Network* [online]. 2015, April 2015 [accessed 2015-04-05]. Available: https://msdn.microsoft.com/

[cor] Apache Software Foundation. *Apache Cordova* [online]. 2015, April 2015 [accessed 2015-04-08]. Available: https://cordova.apache.org/

[qt] Qt Company Ltd. *Qt Documentation* [online]. 2015, April 2015 [accessed 2015-04-08]. Available: http://doc.qt.io/

[xam] Xamarin Inc. *Xamarin* [online]. 2015, April 2015 [accessed 2015-04-11]. Available: http://xamarin.com/

[apc] Appcelerator Inc. *The Appcelerator Platform* [online]. 2015, April 2015 [accessed 2015-04-13]. Available: http://www.appcelerator.com/

[sen] Sencha Inc. *Sencha Touch* [online]. 2015, April 2015 [accessed 2015-04-13]. Available: http://www.sencha.com/products/touch

[est] Estimated. *Estimated* [online]. 2015, April 2015 [accessed 2015-04-15]. Availablez: http://estimated.mobi/

[jira] Atlassian. *Atlassian JIRA* [online]. 2015, April 2015 [accessed 2015-04-17]. Available: https://www.atlassian.com/software/jira

[asp] Softhis. *AgileScrum Pro* [online]. 2015, February 2015 [accessed 2015-04-17]. Available: https://itunes.apple.com/us/app/agile scrum-pro/id547660657?mt=8

[lms] Vinicius Rochedo. *Lion Monkey Scrum* [online]. 2015, July 2014 [accessed 2015-04-18]. Available: https://itunes.apple.com/us/app/lion-monkey-scrum/id506103466?mt=8

[pvt] PivotalTracker. *PivotalTracker* [online]. 2015, April 2015 [accessd 2015-04-19]. Available: http://www.pivotaltracker.com/

[bt] Bluetooth SIG, Inc. *Bluetooth specification* [online]. 2015, April 2015 [accessed 2015-04-20]. Available: https://www.bluetooth.org/en-us

[802.11] IEEE 802.11. *LOCAL AND METROPOLITAN AREA NETWORK STANDARDS* [online]. 2012, March 2012 [accessed 2015-05-05]. Available: http://standards.ieee.org/getieee802/download / 802.11d-2001.pdf

[RFC-6762] RFC-6762. *Multicast DNS* [online]. 2013, February 2013 [accessed 2015-05-05]. Available: https://tools.ietf.org/html/rfc6762

[RFC-6763] RFC-6763. *DNS-Based Service Discovery* [online]. 2013, February 2013 [accessed 2015-05-05]. Available: https://tools.ietf.org/html/rfc6763

[sqlite] SQLite. *SQLite Documentation* [online]. 2015, 2015 [accessed 2015-04-22]. Available: http://sqlite.org/docs.html

[RFC-4122] RFC-4122. *A Universally Unique IDentifier (UUID) URN Namespace* [online]. 2005, July 2005 [accessed 2015-05-05]. Available: https://www.ietf.org/rfc/rfc4122.txt

[SQLitePlg] SQLite pugin. *Cordova SQLite Storage* [online]. 2015, April 2015 [accessed 2015-04-25]. Available: https://github.com/litehelpers/Cordova-sqlite-storage

[JmDNS] JmDNS. *JmDNS library* [online]. 2011, August 2011 [accessed 2015-04-27]. Available: http://jmdns.sourceforge.net/

[jnc] JNCryptor. *JNCryptor* [online]. 2015, March 2015 [accessed 2015-04-26]. Available: https://github.com/RNCryptor/JNCryptor

[rnc] RNCryptor. *RNCryptor* [online]. 2015, April 2015 [accessed 2015-04-28]. Available: https://github.com/RNCryptor/RNCryptor

[GWT] GWT. *GWT* [online]. 2015, May 2015 [accessed 2015-05-02]. Available: http://www.gwtproject.org/

[mGWT] Mgwt. *Mgwt* [online]. 2014, September 2014 [accessed 2015-05-03]. Available: http://www.m-gwt.com/

[gwt-pg] GWTPhoneGap. *GWTPhoneGap* [online]. 2014, October 2014 [accessed 2015-05-03]. Available: https://code.google.com /p/gwt-phonegap/

[gwt-db] GWT Mobile WebKit. *GWT Mobile WebKit Database* [online]. 2010, April 2010 [accessed 2015-05-03]. Available: https://code.google.com/p/gwt-mobile-webkit/

[sql-c] Zetetic LLC. *SQLCipher* [online]. 2014, December 2014 [accessed 2015-05-05]. Available: https://www.zetetic.net/sqlcipher/

List of figures

Figure 2.1: Scrum overview . . . 3
Figure 2.2: Iterative and incremental development . . . 4
Figure 2.3: Requirements in plan driven development . . . 5
Figure 2.4: Cost of decision . . . 6
Figure 2.5: Cost of change . . . 7
Figure 2.6: Scrum principles . . . 7
Figure 2.7: Product owner and his relations to others . . . 8
Figure 2.8: Scrum masters responsibilities . . . 10
Figure 2.9: Scrum activities and artifacts . . . 12
Figure 2.10: Product backlog . . . 13
Figure 2.11: Sprints . . . 14
Figure 2.12: Sprint planning . . . 15
Figure 2.13: Estimation values . . . 17
Figure 2.14: Scrum board . . . 19
Figure 2.15: Burndown chart . . . 20
Figure 2.16: Burndown charts - bad examples . . . 21
Figure 3.1: iOS 8 . . . 24
Figure 3.2: Android . . . 26
Figure 3.3: Windows . . . 27
Figure 3.4: QML example . . . 32
Figure 3.5: Xamarin.Forms . . . 35
Figure 3.6: AppsBilder editor in browser . . . 38
Figure 4.1: Estimated . . . 42
Figure 4.2: JIRA android . . . 43
Figure 4.3: JIRA iOS . . . 43
Figure 4.4: AgileScrum Pro . . . 44
Figure 4.5: Lion Monkey Scrum . . . 45
Figure 4.6: Pivotal tracker . . . 46
Figure 6.1: Saving value . . . 53
Figure 6.2: Loading value . . . 53
Figure 7.1: Basic architecture overview . . . 57
Figure 9.1: Database model . . . 62
Figure 9.2: Database update . . . 63
Figure 10.1: Plugin registration . . . 68
Figure 10.2: WebView entry file . . . 68
Figure 10.3: Service declaration . . . 70
Figure 10.4: FileImport activity definition in AndroidManifest.xml . . . 72
Figure 10.5: FileImport activity . . . 73
Figure 10.6: Starting a file sharing intent . . . 74
Figure 10.7: JmDNS plugin Java part . . . 75
Figure 10.8: TCPServer plugin registration . . . 75
Figure 10.9: Calling Cordova plugin from JavaScript . . . 76
Figure 11.1: Starting mail composer in Objective-C . . . 78
Figure 11.2: Plugin interface in header file . . . 79
Figure 11.3: Plugin operation implementation . . . 80
Figure 11.4: Plugin registration on iOS . . . 80
Figure 12.1: HTTP request format . . . 81

Figure 12.2: HTTP response format 81
Figure 13.1: GWT native method 86
Figure 13.2: Java call 86
Figure 13.3: JSNI example 86
Figure 13.4: mGWT platform themes 87
Figure 15.1: Callback hell 99

List of abbreviations

A2DP	Advanced Audio Distribution Profile
ADT	Android Developer Tools
AES	Advanced Encryption Standard
API	Application Programming Interface
BSD	Berkeley Software Distribution
BT	Bluetooth
CSS	Cascading Style Sheets
CVS	Comma Separated Values
DAO	Database Access Object
DB	Database
DNS	Domain Name System
DTO	Database Transfer Object
GPL	General Public License
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
GUID	Globally Unique Identifier
GWT	Google Web Toolkit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
IDE	Integrated Development Environment
IP	Internet Protocol
JNI	Java Native Interface
JSNI	JavaScript Native Interface
JSON	JavaScript Object Notation
JmDNS	Java Multicast Domain Name System

LGPL	GNU Lesser General Public License
mDNS	Multicast Domain Name System
mGWT	Mobile Google Web Toolkit
MS	Microsoft
NDK	Native Development Kit
OBEX	OBject EXchange
OS	Operating System
PBKDF2	Password-Based Key Derivation Function 2
PO	Product Owner
QML	Qt Meta Language
RAM	Random Access Memory
REST	Representational State Transfer
SDK	Software Development Kit
SM	Scrum Master
SMS	Short Message Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TM	Team Member
UI	User Interface
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
WLAN	Wireless Local Area Network
WP	Windows Phone
WPAN	Wireless Personal Area Network
WWDC	Apple Worldwide Developers Conference
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

CD content

- **[bin] –** this folder contains distributable compiled files, iOS installation is not present, as it is only possible to install applications from Apple AppStore or using Xcode on iOS devices.
  - **[Android] –** this folder contains installable android APK file.
  - **[GWT] –** this folder contains compiled JavaScript from GWT
- **[src]** – this folder contains sources
  - **[Android]** – Android Studio project for native Android part
  - **[GWT]** – GWT sources in form of eclipse project
  - **[iOS]** – Xcode project with native iOS part
- **[doc]** – this folder contains documentation
  - **[Android]** – JavaDoc for Android project
  - **[GWT]** – JavaDoc for GWT project
  - **[Resources]** – Images and other resources
  - **[Text]** – Thesis text in PDF and ODT files
- **readme.txt** – CD content description

Attachment 1 – Database tables

Here is a list of all tables, their structure and types.

Dbvers

This table serves for database updates and it's purpose was explained in the text.

Preferences

Preferences is another simple table used for storing various key-value data. It has two columns: key and value. The key is unique. The `PreferencesDAO` class ensures that only one row with each key exists. Key and value are by default stored as strings.

Type

Type is an enumeration table containing two sets of values one is for stories and issues and second is for tasks. It also specifies the color used to display type. It has three columns:

- id – primary key
- name – type name (Bug, Todo ...)
- color – HTML color for displaying the type

This table is filed with default values, which cannot be user edited. See method **`createDB`** in `ScrumAppDAO` for actual values.

Role

Role is an enumeration table for user roles, there are two columns:

- id – primary key
- name – role name (Scrum master, Product owner, Team member)

This table is filed with default values, which cannot be user edited. See method **`createDB`** in `ScrumAppDAO` for actual values.

Status

Status is an enumeration table containing three sets of values one is for stories, second for issues and last is for tasks. It has two columns:

- id – primary key
- name – status name (Free, In progress, Done ...)

This table is filed with default values, which cannot be user edited. See method **createDB** in **ScrumAppDAO** for actual values.

Priority

Priority is an enumeration table containing a set of values for issues priority:

- id – primary key
- name – priority name (low, normal ...)
- color – HTML color for displaying the priority

This table is filed with default values, which cannot be user edited. See method **createDB** in **ScrumAppDAO** for actual values.

User

User is a table containing information about one user. There are no cross project users, if a user is in two projects he has to be defined in both and there will be two rows in the user table for him (this is by design to simplify the synchronization, as there is no global user table defined anywhere, each device only knows users in projects it has access to).

Columns:

- id – primary key
- name – user's full name
- login – user's login, he uses it to authenticate himself to the server
- password – user's password, he uses it to authenticate himself to the server, the password is stored in hashed form (SHA1)
- nick – user's nick, shown on most GUI items like tasks and stories
- color – HTML color to allow user to select whatever color he likes
- valid – flag for setting if the user is activated and can connect to the master device, invalid users are refused, if they try to synchronize
- project – reference to project, specifies to which project the user belongs to
- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

User_role

User role is a junction table between user and role table. This allows one user to have more than one role.

Columns:

- id – primary key
- user – reference to user
- role – reference to role
- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

Project

Project table stores information about a project.

Columns:

- id – primary key
- name – project name
- description – project description
- local – flag, if project is local and this device act as a master for this project
- enabled – flag, if a local project is enabled and allows others to synchronize with it
- leader – reference to user, represents the scrum master of this project
- backlog – reference to sprint, represents the product backlog for this project, it is created automatically with a new project
- current – reference to sprint, represents current sprint
- next – reference to sprint, represents next sprint
- address – used by slave devices, represents last known IP address of master device
- start – project start date stored as integer
- end – project end date stored as integer
- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

Credentials

Credentials is a table which stores logins and passwords for for project, so user does not have to write them every time he starts the application.

- id – primary key
- login – login for project

- password – password for project
- project – reference to project

Sprint

Sprint table hold information about a single sprint.

Columns:

- id – primary key
- name – sprint name
- start – sprint start date stored as integer
- end – sprint end date stored as integer
- project – reference to project, which the sprint belongs to
- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

Story

Story table holds information about a story and their order (priority) in the sprint. Each story has a reference on the next one creating a linked list ordered by priority. A special story called TOP serves as a list head and is invisible to the user.

Columns:

- id – primary key
- name – story name
- description – story description
- created – date when was the story created stored as integer
- added – date when was the story added to print backlog stored as integer
- finished – date when was the story finished stored as integer
- estimation – estimation of this story in story points stored as real
- author – reference to user, author of the story
- type – reference to type, type of story
- status – reference to status, status of story
- sprint – reference to sprint, which this story belongs to
- comment – product owner's comment on this story
- next – reference to story, represents the next story in the backlog

- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

Task

Task represents a single task. Each story is broken down into several tasks.

Columns:

- id – primary key
- name – task name
- created – date when was the task created stored as integer
- finished – date when was the task finished stored as integer
- estimation – estimation of this task in hours stored as real
- author – reference to user, author of the task
- assigned – reference to user, user working on the task
- type – reference to type, type of task
- status – reference to status, status of task
- sprint – reference to sprint, which this task belongs to
- story – reference to story, which this task is part of
- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

Issue

Issue represents a bug, todo etc. and can be created by anyone, the product owner then can convert issues to stories or refuse them.

Columns:

- id – primary key
- name – issue name
- description – issue description
- priority – reference to priority, represents this issue priority
- created – date when was the issue created stored as integer
- closed – date when was the issue closed stored as integer
- author – reference to user, author of the issue

- type – reference to type, type of issue
- status – reference to status, status of issue
- project – reference to project, which this issue belongs to
- story – reference to story, which this issue was converted into
- lastchange – used for synchronization, will be described later
- modified – used for synchronization, will be described later

Voting

Voting table is used for storing results of running or last finished scrum poker game.

Columns:

- project – reference to project, which this scrum poker result belongs to
- story – reference to story that is being estimated
- user – reference to user, whose vote this is
- vote – the actual vote of this user

Sync

Sync is a table used only for synchronization, it contains information about tables and if they require synchronization from client to server (have some uncommitted changes).

Columns:

- project – reference to project, which is subject to synchronization
- tablename – table name from the project
- lastchange – date of last change in the table
- required – flag if there have been made changes

Attachment 2 – Test scenarios

Create project

Create an empty project.

Prerequisites: be on an Android device.

	Step	Expected result	Result
1	Go to projects page and press new button	Project creation form shows	Project creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to projects page and the created project should be listed here and set as current project	Project created successfully

Test passed

Add user

Add a new user to local project.

Prerequisites: be on android and local project must be selected as current

	Step	Expected result	Result
1	Go to users page and press add	User creation form shows	User creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to users page and the created user should be listed here with correct name and roles	User created successfully

Test passed

Connect to a remote project

Connect to a remote project and download its data.

Prerequisites: Wi-Fi running and connected, master device with at least one available project must be on the same network.

	Step	Expected result	Result
1	Go to projects page select the remote project and press connect	Login page shows	Page showed
2	Fill in wrong credentials and press OK	Error message shows	Error message showed
3	Fill correct credentials and press OK	Application should go to projects page and show success dialog, project should be listed here and set as current project	Project downloaded successfully

Test passed

Create sprint

Create a new sprint in empty project.

Prerequisites: be connected to a remote project as a product owner.

	Step	Expected result	Result
1	Select edit current sprint on the dashboard	Sprint creation form shows	Sprint creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to dashboard page and the created sprint should be showed here with correct data	Sprint created successfully

Test passed

Create issue

Create a new issue.

Prerequisites: be connected to a remote project.

	Step	Expected result	Result
1	Go to issues page and press new button	Issue creation form shows	Issue creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to issues page and the created issue should be listed in open tab	Issue created successfully
4	Go to issues page on master device	The new issue should appear here in few seconds	New issue appeared

Test passed

Close issue

Close an issue.

Prerequisites: be connected to a remote project as a product owner, at least one open issue must be created.

	Step	Expected result	Result
1	Go to issues page and select an open issue	Issue close form should show	Issue close form showed
2	Fill the required fields correctly accepting the issue	Application should go to issues page and the closed issue should be listed on the closed tab	Issue closed successfully
3	Go to product backlog page	New story with data from the issue should appear	New story appeared
4	Go to issues page on master device	The closed issue should appear in closed in few seconds	Closed issue appeared
5	Go to product backlog page on master device	New story with data from the issue should appear in few seconds	New story appeared

Test passed

Create story

Create a new story.

Prerequisites: be connected to a remote project as a product owner.

	Step	Expected result	Result
1	Go to product backlog page and press new	Story creation form shows	Story creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to product backlog page and the created story should be listed here	Story created successfully
4	Go to product backlog page on master device	New story should appear in few seconds	New story appeared

Test passed

Move story

Move story from product backlog to sprint backlog.

Prerequisites: be connected to a remote project as a product owner.

	Step	Expected result	Result
1	Go to product backlog page, select s story and press pull to sprint	Pull to sprint form shows	Pull to sprint form showed
2	Select a position and pres OK	Application should go to product backlog page and the story shouldn't be here	Story disappeared
3	Go to sprint backlog page	The moved story should be here on selected position	Story moved successfully
4	Go to sprint backlog page on master device	The moved story should be here on selected position in few seconds	Story moved successfully

Test passed

Delete story

Delete story from product backlog.

Prerequisites: be connected to a remote project as a product owner, at least one story must be created in product backlog.

	Step	Expected result	Result
1	Go to product backlog page, select a story and press delete	Application should go to product backlog page and the story shouldn't be here	Story deleted successfully
2	Go to product backlog page on master device	The deleted story shouldn't be here	Story deleted successfully

Test passed

Scrum poker game

Play the scrum poker game.

Prerequisites: One master device and two clients should be connected to the same project, at least one story should be in product backlog.

	Step	Expected result	Result
1	Go to product backlog page on master device, select a story and press start scrum poker	Scrum poker game page should be shown with the selected story	Scrum poker game page showed
2	Pres join scrum poker on client 1 and 2	Scrum poker game page should be shown with the selected story	Scrum poker game page showed
3	Make a vote on all three devices	Devices that voted should be marked in the list, only your vote should be visible	Voting showed, only my vote was visible
4	End voting on master device	All devices should show the voting result	Result showed on all devices correctly

Test passed

Create task

Create a new task.

Prerequisites: be connected to a remote project, at least one story must be created.

	Step	Expected result	Result
1	Go to scrum board page and press new task button	Task creation form shows	Task creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to scrum board page and the created task should be listed in open tab under the story it belongs to	Task created successfully
4	Go to scrum board page on master device	The new task should appear here in few seconds	New task appeared

Test passed

Claim task

Claim a task.

Prerequisites: be connected to a remote project, at least one task must be created.

	Step	Expected result	Result
1	Go to scrum board page, select an open task and press claim task button	Task should move to work tab and should be assigned to you	Task claimed successfully
2	Go to scrum board page on master device	The claimed task should appear here in work tab in few seconds	Claimed task appeared

Test passed

Free task

Free a task.

Prerequisites: be connected to a remote project, at least one task must be created and assigned in work state to you.

	Step	Expected result	Result
1	Go to scrum board page, select a work task and press free task button	Task should move to open tab and should no longer be assigned to you	Task freed successfully
2	Go to scrum board page on master device	The freed task should appear here in open tab in few seconds	Freed task appeared

Test passed

Close task

Close a task.

Prerequisites: be connected to a remote project, at least one task must be created and assigned in work state to you.

	Step	Expected result	Result
1	Go to scrum board page, select a work task and press close task button	Task should move to done tab	Task closed successfully
2	Go to scrum board page on master device	The closed task should appear here in done tab in few seconds	Closed task appeared

Test passed

Create next sprint

Create a next sprint.

Prerequisites: be connected to a remote project as a product owner.

	Step	Expected result	Result
1	Select edit next sprint on the dashboard	Sprint creation form shows	Sprint creation form showed
2	Press OK without filling anything	Error message with required fields should show	Error message showed
3	Fill the required fields correctly	Application should go to dashboard page	Sprint created successfully

Test passed

Switch to next sprint

Switch to next sprint, moving all unfinished stories with it.

Prerequisites: be on an Android device current and next sprint must be created, some open and finished stories should be in the sprint backlog.

	Step	Expected result	Result
1	Select switch to next sprint on the dashboard and pres OK	A new sprint should appear on the dashboard, all unfinished stories should be moved to this new sprint, the old sprint should appear in the sprint history	Switched to next sprint successfully
2	Go to master device	A new sprint should appear on the dashboard, all unfinished stories should be moved to this new sprint, the old sprint should appear in the sprint history	Switched to next sprint successfully

Test passed

Export project

Export a project to a file.

Prerequisites: A project must exist on the device.

	Step	Expected result	Result
1	Go to project list page, select a project and press export	Export form shows	Export form showed
2	Write a name for the backup and pres export	Application should go to the the project list page and a new backup with selected name should appear in the backup page	Project data exported successfully
3	Go to backup page, select the created backup and pres share	Android: Share dialog with compatible applications should show	Share dialog showed
		iOS: Email composer should show with the exported file as attachment	Email composer showed
4	Open the exported file	Exported file contains project data in JSON	File contains correct data

Test passed

Import project

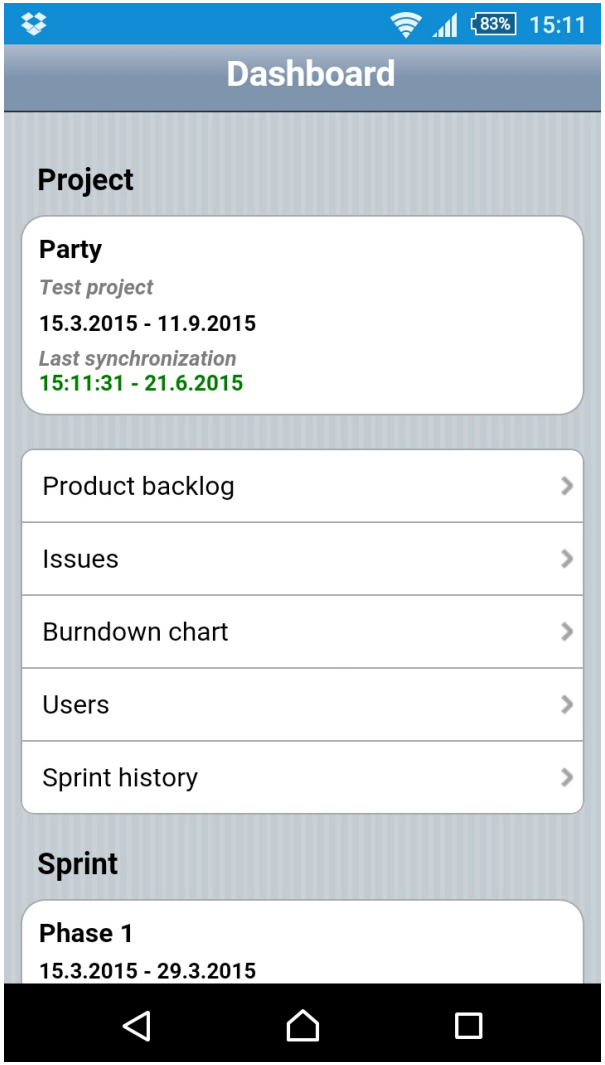
Import a project from a file.

Prerequisites: Project backup file must exist.

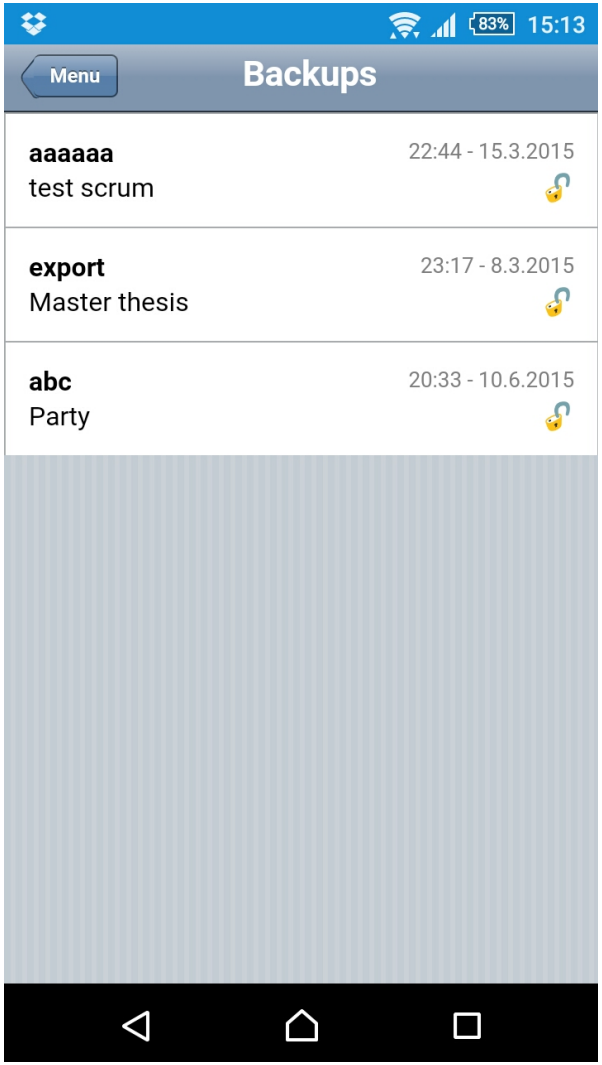
	Step	Expected result	Result
1	iOS: Copy the backup file to application documents using iTunes	The backup should appear in the backup page	Backup showed in the backups list
	Android: Select the application file in some file manager on the device and open it with ScrumApp and press import	The backup should appear in the backup page	Backup showed in the backups list
3	Go to backup page, select the importedbackup and press import	Import form shows	Import form showed
4	Select user, enter correct password and press import	The project should appear in the project list page or be overwritten if it existed and should have all data from the backup	Import successful

Test passed

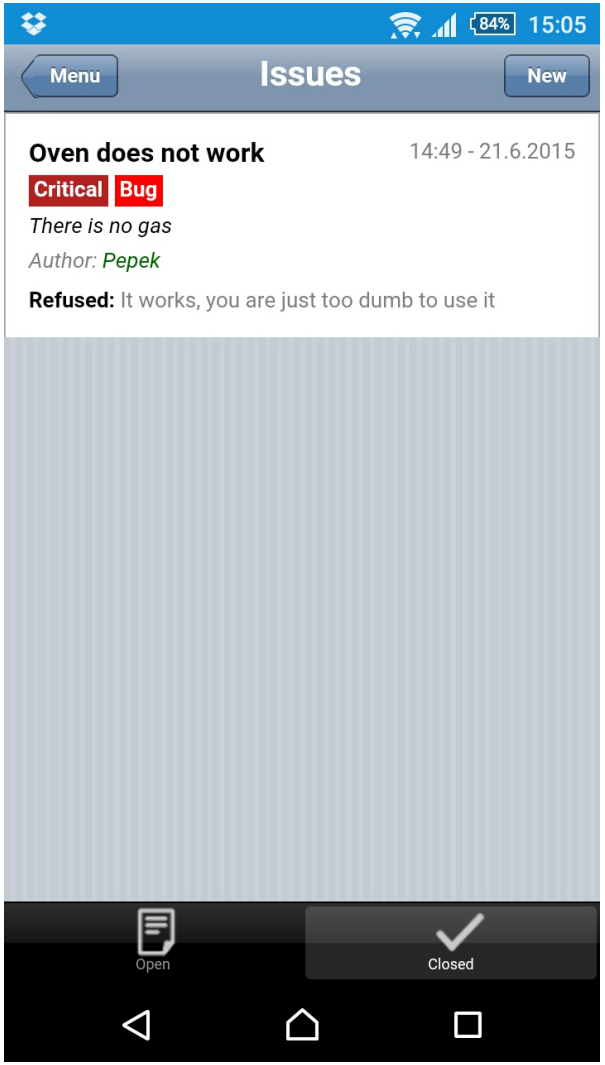
Attachment 3 – Screenshots



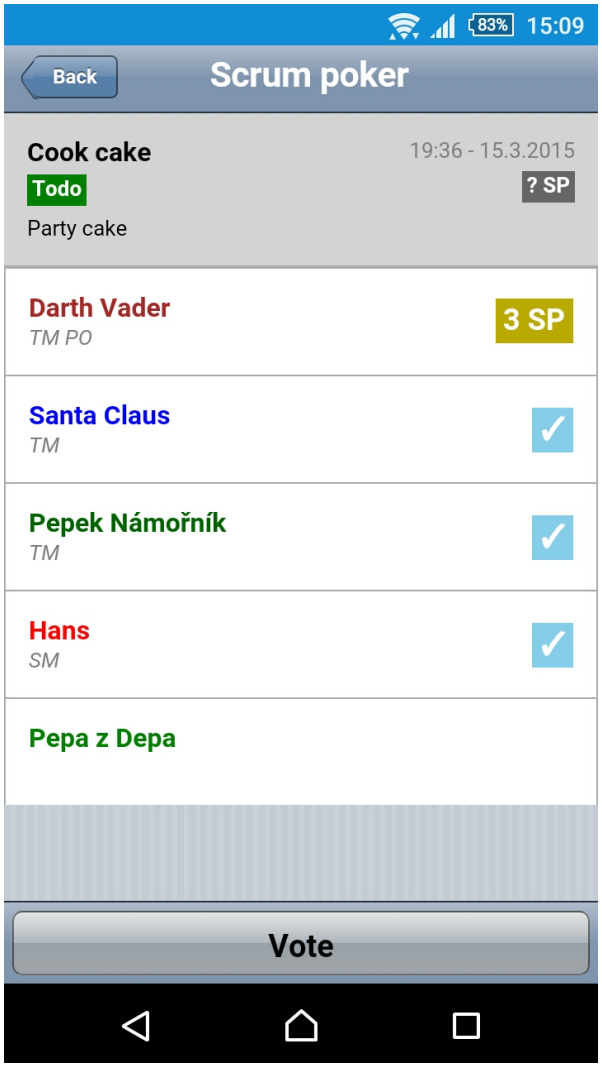
Screenshot 2: Dashboard

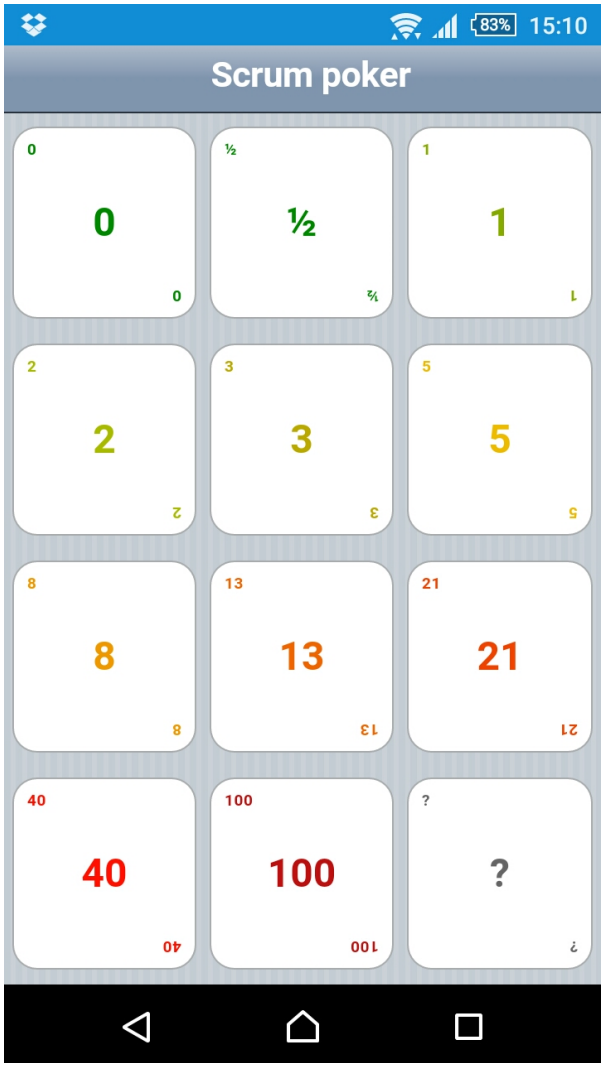


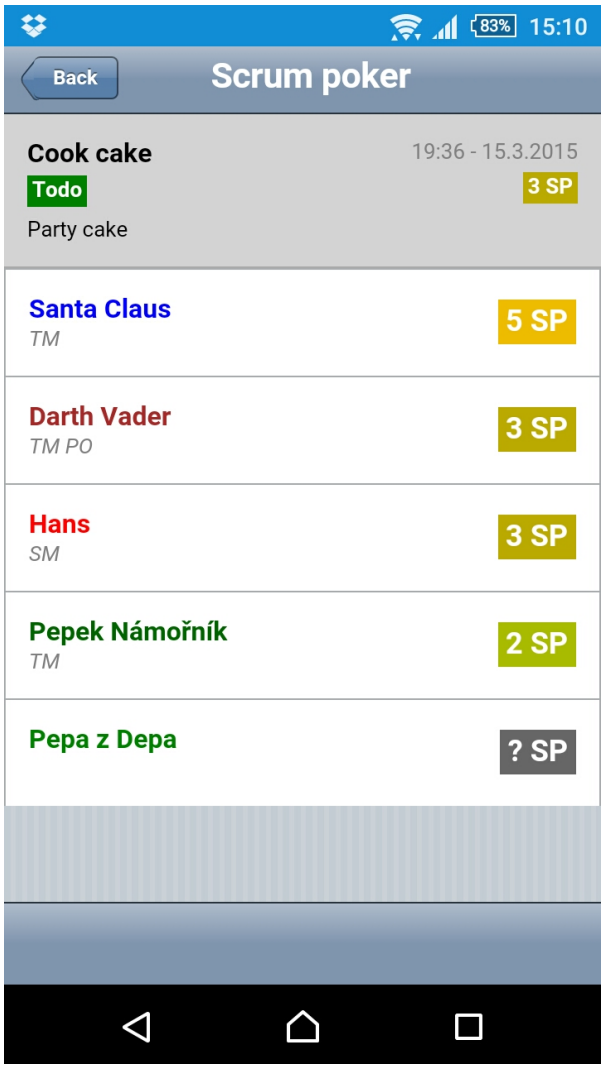
Screenshot 3: Backups

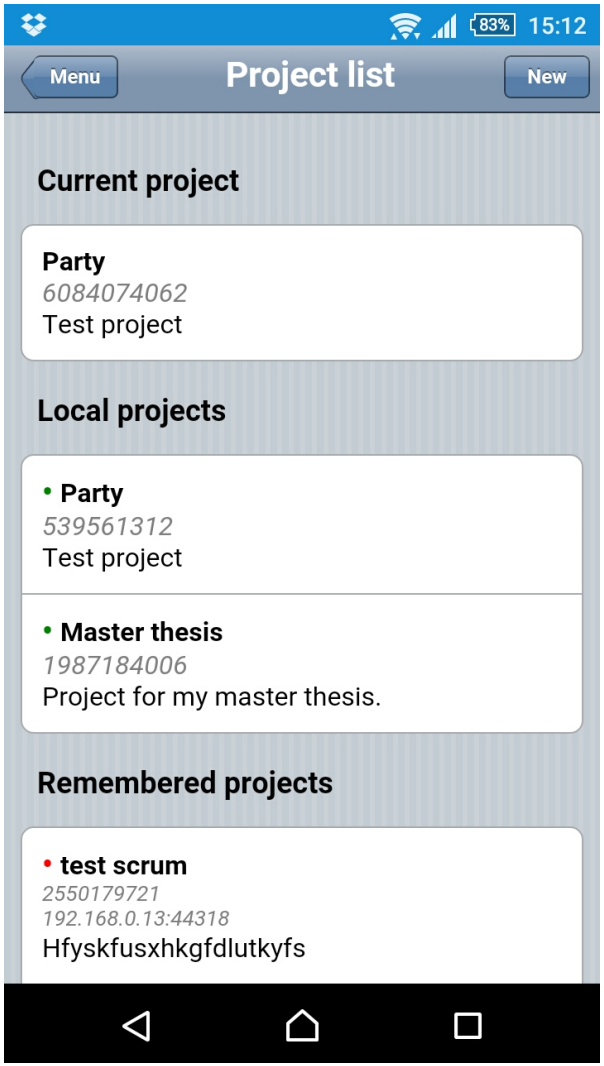


Screenshot 1: Closed issues

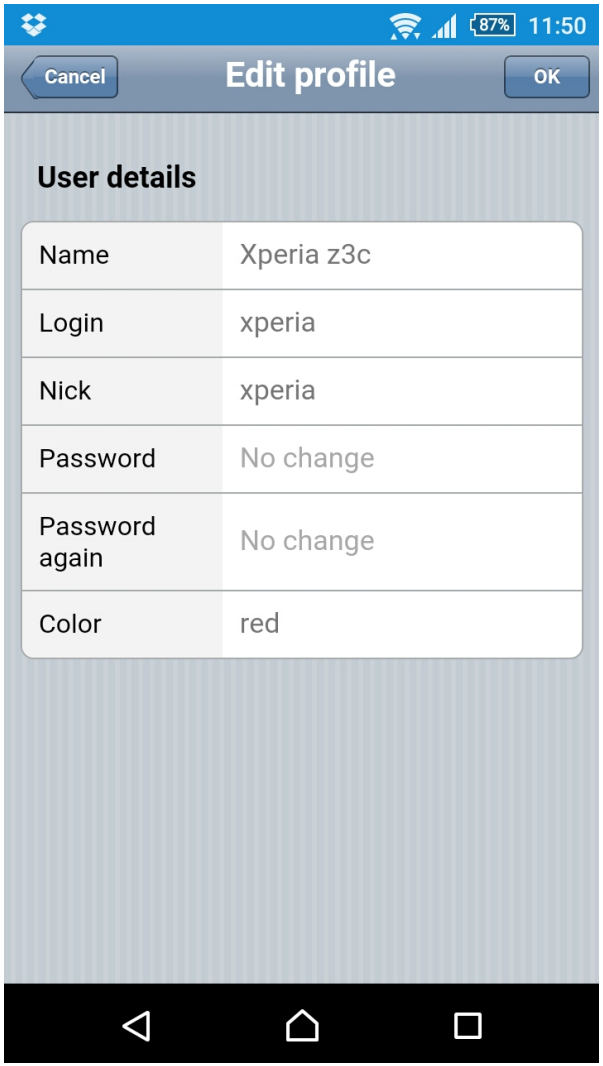

83% 15:09
Back
Scrum poker
Cook cake
19:36 - 15.3.2015
Todo
? SP
Party cake
Darth Vader
TM PO
3 SP
Santa Claus
TM
Pepek Námořník
TM
Hans
SM
Pepa z Depa
Vote


83% 15:10
Scrum poker
0
½
1
2
3
5
8
13
21
40
100
?

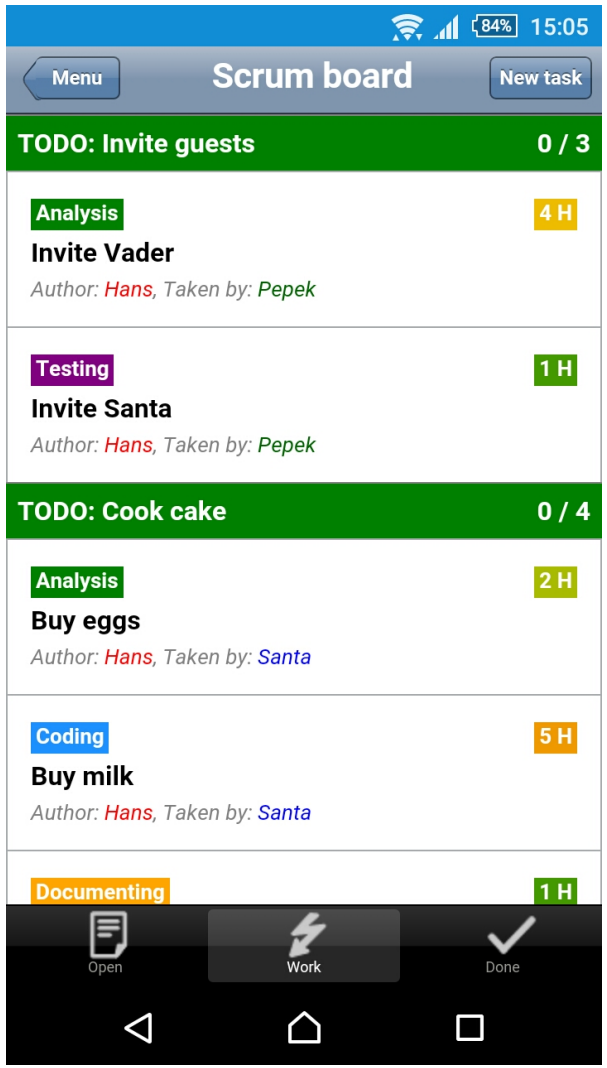

83% 15:10
Back
Scrum poker
Cook cake
19:36 - 15.3.2015
Todo
3 SP
Party cake
Santa Claus
TM
5 SP
Darth Vader
TM PO
3 SP
Hans
SM
3 SP
Pepek Námořník
TM
2 SP
Pepa z Depa
? SP



Screenshot 6: Project list

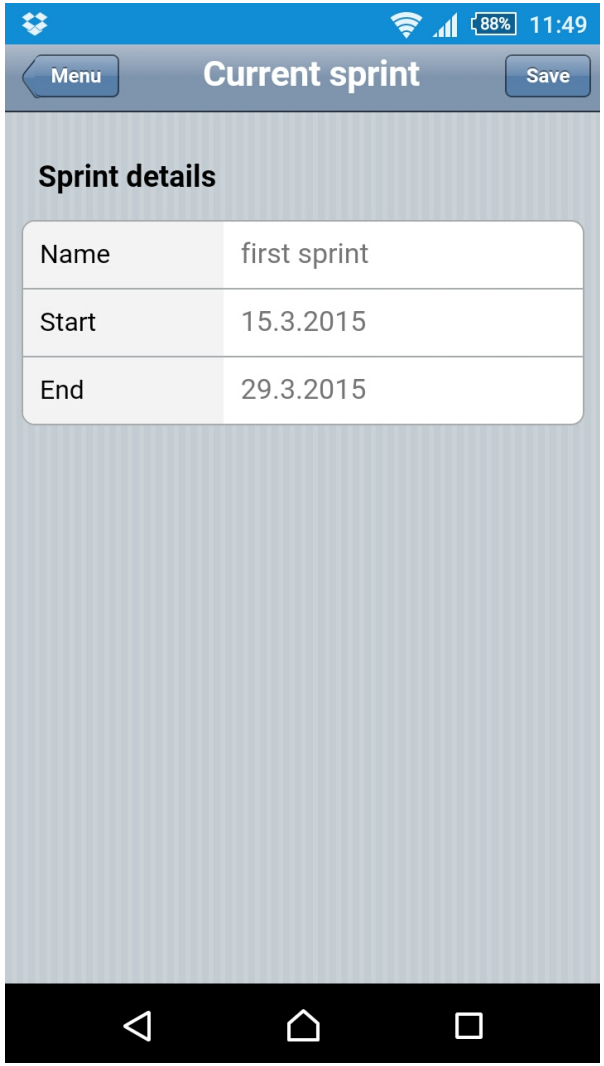


Screenshot 4: Profile editation

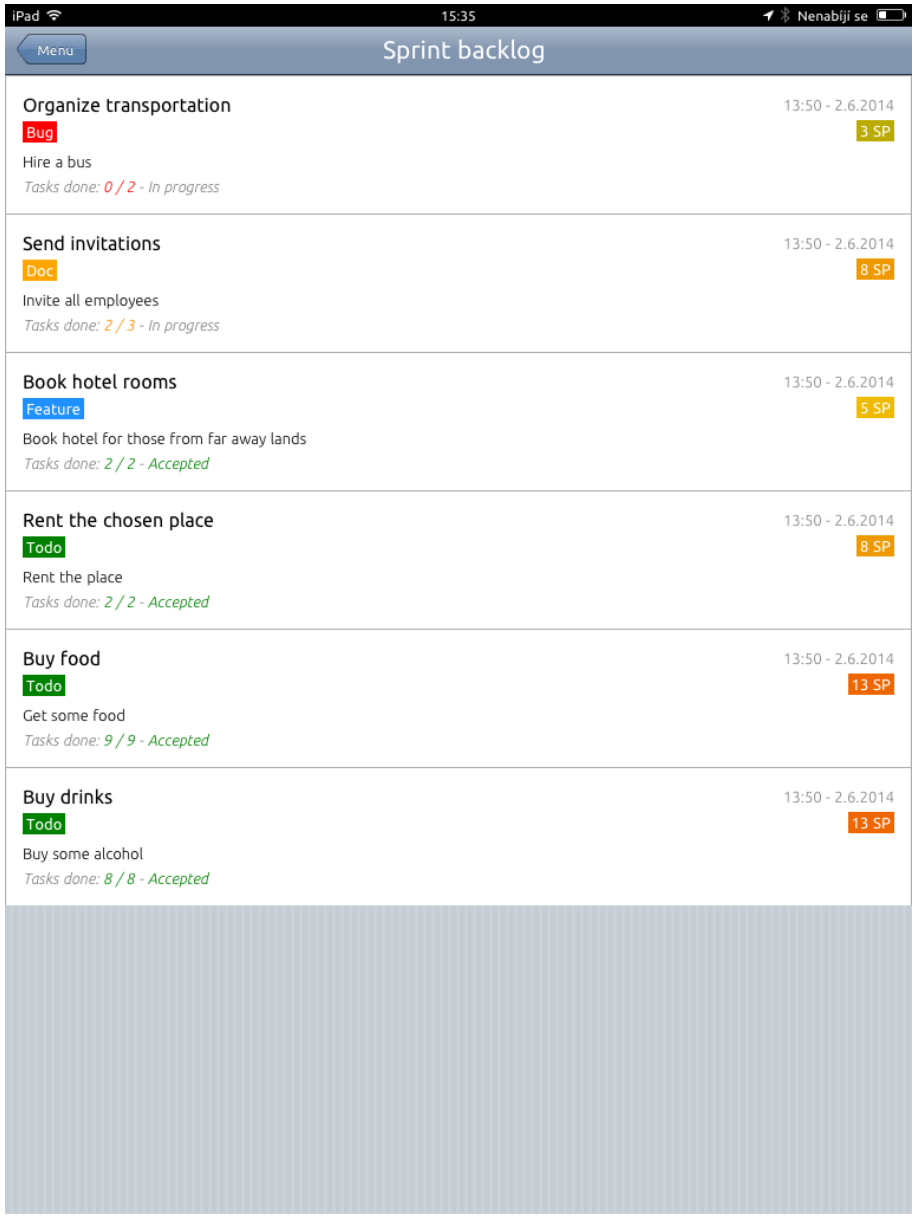


Screenshot 5: Scrum board

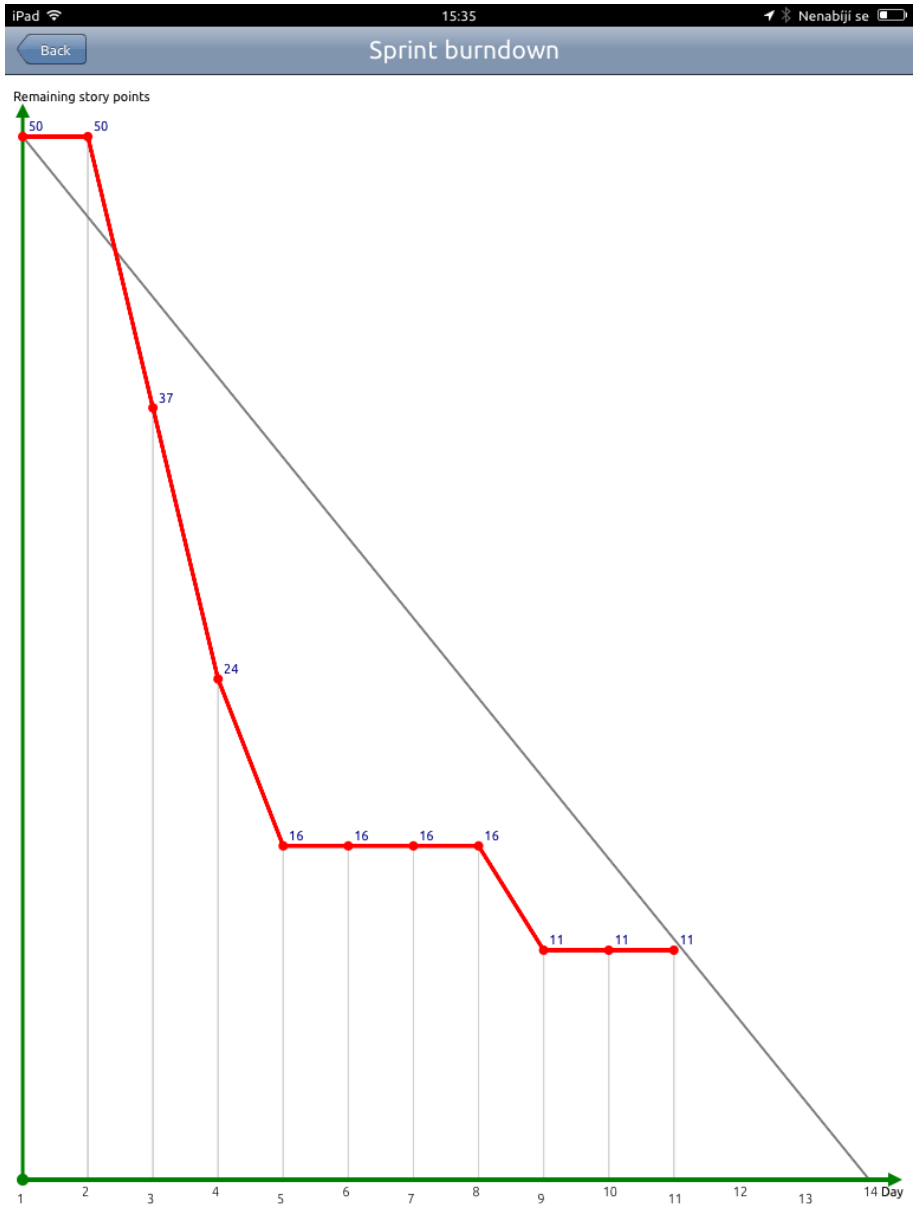
Screenshot 8: List of users



Screenshot 7: Editation of current sprint

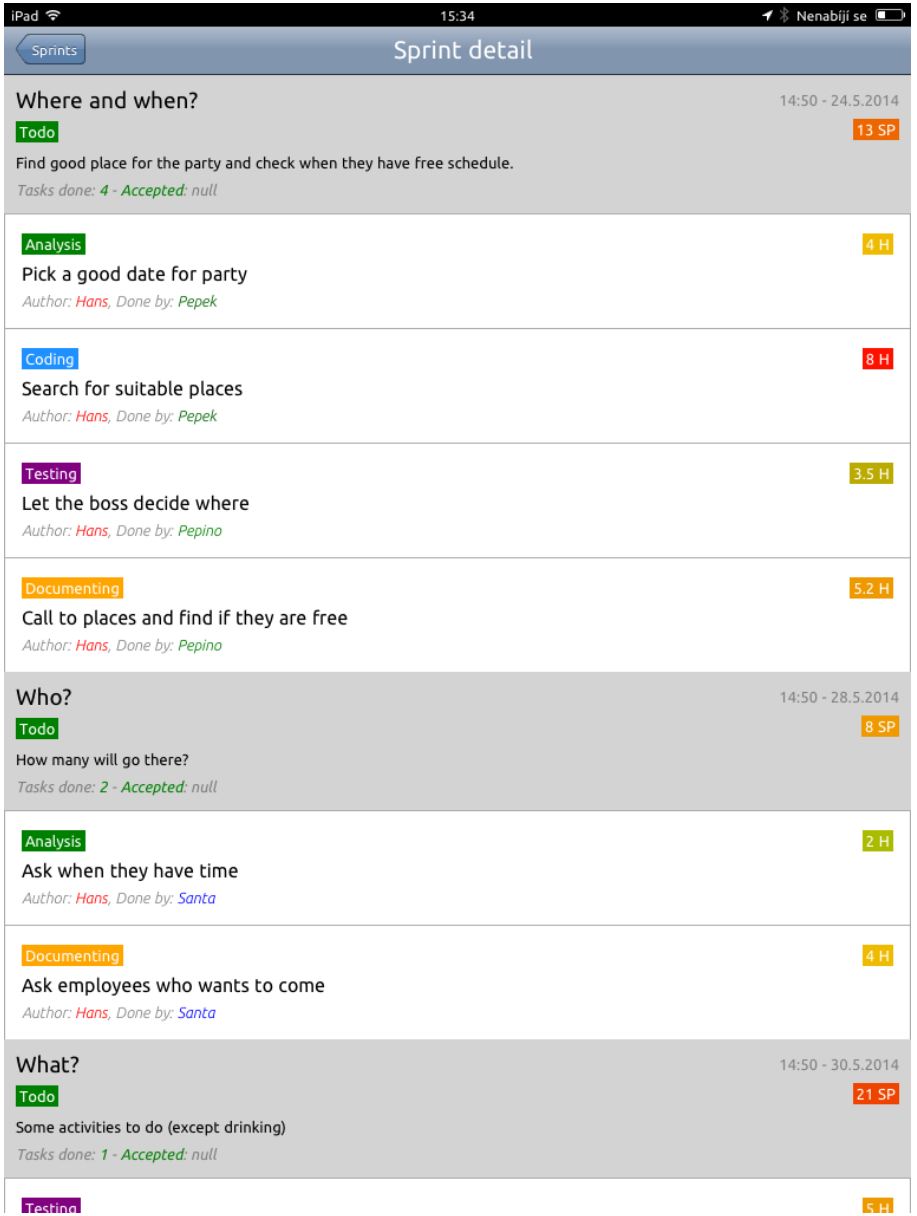


Screenshot 9: Sprint backlog



Screenshot 10: Sprint burndown chart

Screenshot 12: List of finished sprints



Screenshot 11: Detail of finished sprint

University of West Bohemia

Faculty of Applied Sciences

Department of Computer Science and Engineering

ScrumApp

User manual

Pilsen, 2015

Jan Strejc

Table of contents

1 Program **1**
1.1 Compilation and developement 1
1.2 Installation 2
1.3 System requirements 2

2 User interface **3**
2.1 General controls 3
2.2 Dashboard 4
2.3 Working with projects 4
2.4 Adding and editing users 5
2.5 Working with sprints 6
2.6 Working with issues 6
2.7 Product and sprint backlog 8
2.8 Scrum board 8
2.9 Scrum poker 9
2.10 Backups, File import and export 9

1 Program

1.1 Compilation and developement

The most convenient way for compiling all three projects is using an IDE.

GWT

1. First you will need installed Eclipse IDE and a GWT plugin for it with GWT SDK.

   You can follow this page for instructions:

   http://www.gwtproject.org/usingeclipse.html

2. Then open the project located on CD in folder **`/src/GWT`** with Eclipse
3. You can use Eclipse and development mode to run the multiplatform part of application in a compatible browser (Chrome is preferred). Not all functionality will work in this mode as networking and other are not available. Database will work using embedded SQLite in Chrome. See the GWT documentation for up to date guide.
4. For compiling the Java sources to JavaScript you can right click the project and select Google > GWT Compile. This will create compiled JavaScript in selected folder.
5. You can copy the resulting JavaScript to one of the native projects to their **www** folder to use it.

Eclipse and GWT SDK works on Windows, Linux and Mac.

Android

1. First you will need installed Android studio IDE and Android SDK.

   You can follow this page for instructions:

   https://developer.android.com/sdk/index.html

2. Then open the project located on CD in folder **`/src/Android`** with Android studio. You can run the application in emulator or on real device.

Android studio and Android SDK works on Windows, Linux and Mac.

iOS

1. First you will need installed Xcode on your Mac, you can download and install it from iTunes.
2. Then open the project located on CD in folder **`/src/iOS`** with Xcode studio. You can run the application in simulator.
3. To run the application on real device you need an Apple developer account and set up your device to enable development, You also need to set up the application in the developer account, this process is not simple, please see the official documentation from Apple for more details:

   https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/LaunchingYourApponDevices/LaunchingYourApponDevices.html

1.2 Installation

The application is not available on Google play and Apple AppStore. Installation on iOS can be currently done only using Xcode and the above described method.

Installation on Android requires allowed installation from unknown sources in the device's settings. You can then install the application from provided installable APK package located on CD in **`/bin/Android`** folder.

1.3 System requirements

Application requires iOS 7 and newer or Android 2.3.4 and newer.

The mobile device must have Wi-Fi.

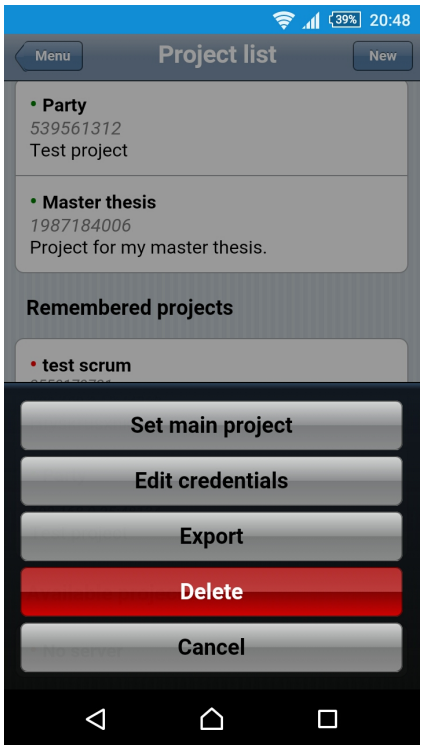
2 User interface

2.1 General controls

The ScrumApp application is organized into pages, each page has a header bar that contains the page name in center. On left side the header has a back navigation button, that goes to previous page, devices with hardware or software back button (android) can use it instead. On many pages on right part of the header is an action button often used to add new content to the page.

The rest of the page is occupied by it's content and can be vertically scrolled. Some pages have special controls in a footer, like tab switcher on the scrum board.

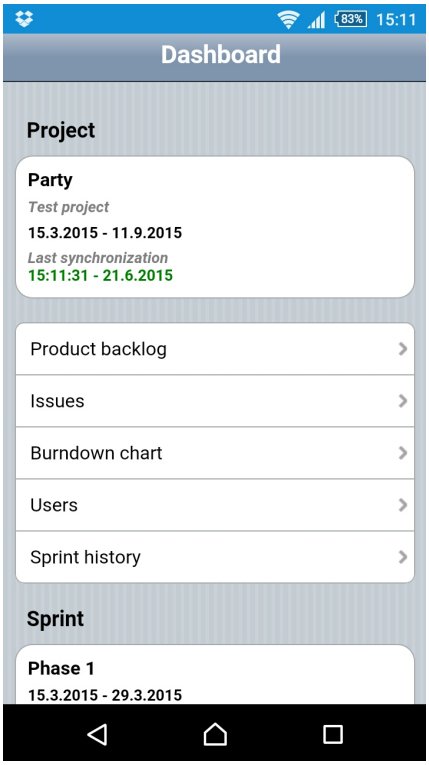
When multiple actions are available for some content, (for example after clicking a project in the project list) a multi choice menu is displayed on the bottom of the page (screenshot 1).



Screenshot 1: General controls

2.2 Dashboard

The dashboard (screenshot 2) is a central point of the application, it serves as an overview of currently selected project and sprint and as a menu for navigating to other parts of application.



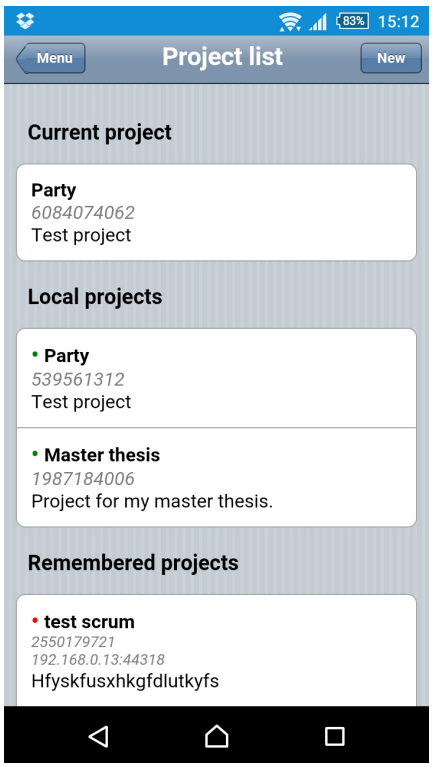
Screenshot 2: Dashboard

2.3 Working with projects

On the projects page selected from dashboard is a list of projects organized into groups (screenshot 3):

- On top is currently selected project.
- Next are local projects, for which this device acts as a master. You can edit, disable, export and delete local projects.
- Remembered projects are remote project, for which local data copy is available (you have connected to them). You can export, edit your credentials and delete local copy of data for remembered projects.
- Available projects are remote projects available on the network. You can connect to available projects, if you provide correct credentials.

This list removes duplicities, if a project is in remembered it won't show int available, but it will have a green dot as it's master device is available on the network. Local and remembered projects can be set as current (only one project can be selected as current and worked with).



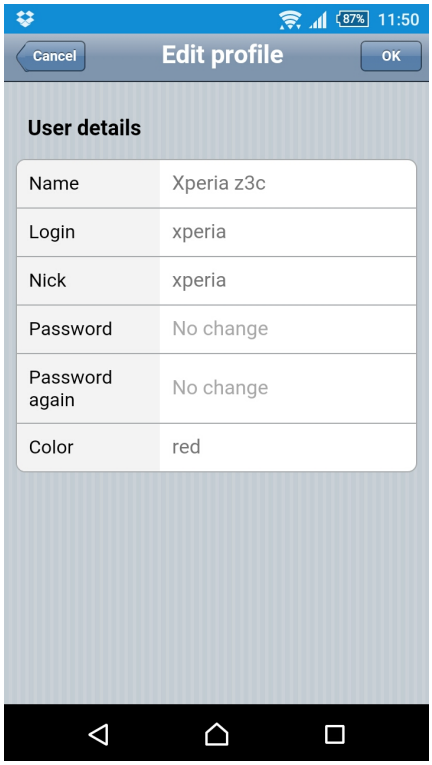
Screenshot 3: Projects

2.4 Adding and editing users

On the users page there is a list of users in the project (screenshot 4), if you are a scrum master, you can edit their profiles, add new users or disable users to prevent them from accessing the project. You can also assign roles to users, there can be only one scrum master (you, it is not possible to change scrum master), one product owner and several team members, if needed one user can have more than one role.

Users can edit their profile on the edit profile page in the dashboard. They need a working connection to master device to save the changes. (screenshot 5)

Screenshot 4: Users list



Screenshot 5: Profile editation

2.5 Working with sprints

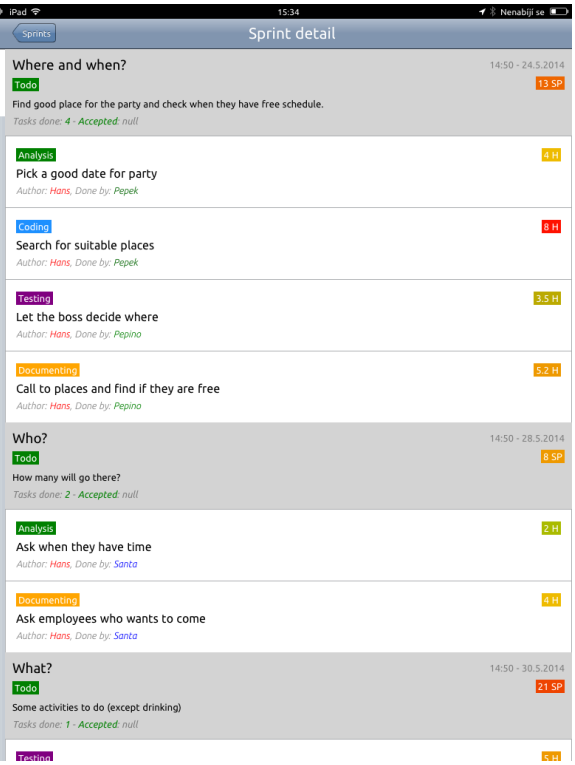
Information about current sprint can be seen on dashboard. Old sprints can be viewed on the sprint history page, velocity, finished stories, tasks and burndown chart is displayed here (screenshot 6, 7 and 8).

Only product owner can edit or create sprints from the dashboard. He can create or edit current or next sprint. When product owner switches to the next sprint, all unfinished stories from the current sprint are moved to it.

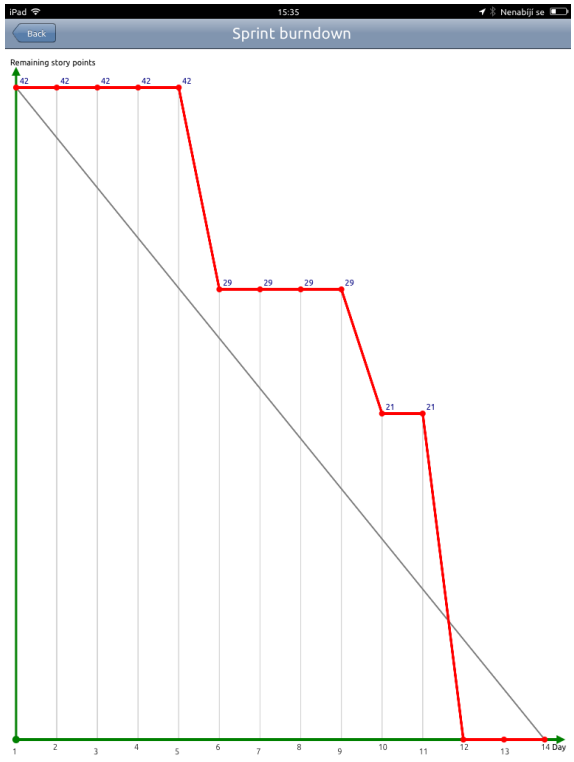
2.6 Working with issues

The issues page contains two lists of issues, one for open and one for closed (screenshot 9). Anyone can create issues, but only product owner can accept them or refuse them. I he accepts the issue he can create a new story from it immediately.

Screenshot 6: Sprint history



Screenshot 7: Sprint history detail



Screenshot 8: Burndown chart

Screenshot 9: Issues

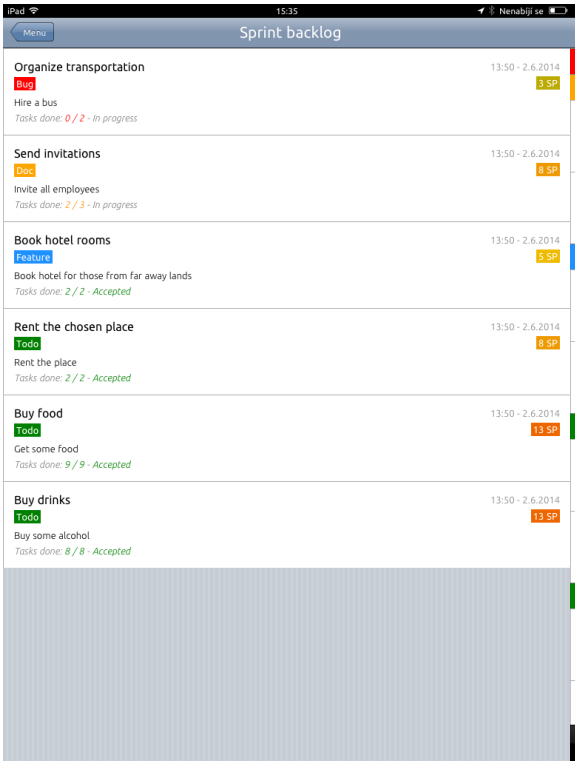
2.7 Product and sprint backlog

The product and sprint backlogs allow the product owner to manage stories. As a product owner you can create new stories here, edit them an order them. You can also move stories from the product backlog to the sprint backlog (screenshot 10). The scrum master can start the scrum poker game to estimate a story from here.

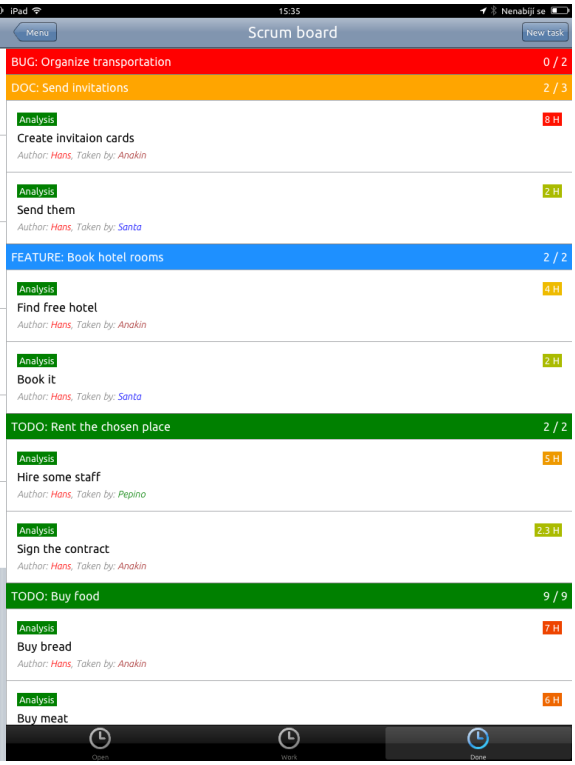
2.8 Scrum board

Scrum board (screenshot 11) is for task management, it shows stories from current sprint and allows breaking them down to tasks. Any one can add a task to a story. Tasks are organized to three tabs:

- open – unassigned tasks, any team member can claim it and start working on it, if he has a working connection to master device.
- work – tasks that are in progress are displayed here, you can free (move to open) or close (move to done) tasks that are assigned to you. You can also edit them.
- done – these tasks have been finished (closed)



Screenshot 10: Sprint backlog

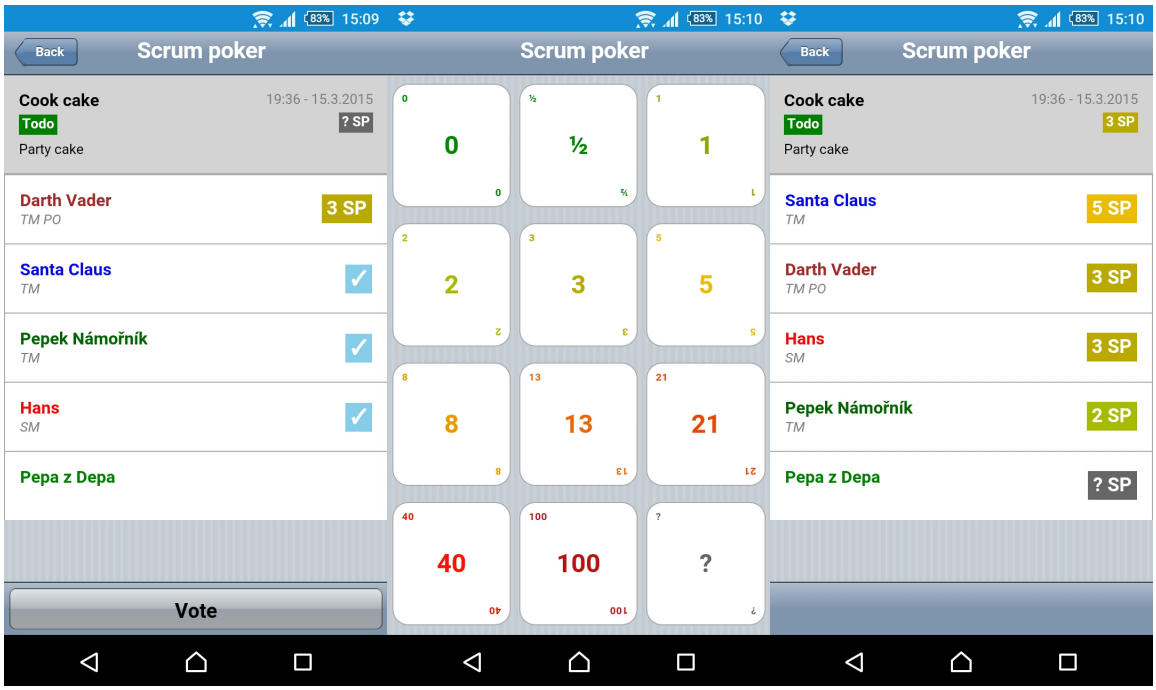


Screenshot 11: Scrum board, done tasks

2.9 Scrum poker

The scrum poker game can be launched from product or sprint backlog by scrum master. It is always played with one story at a time. After scrum master started the game, other users from the project can join it from the dashboard. Scrum poker game has two phases:

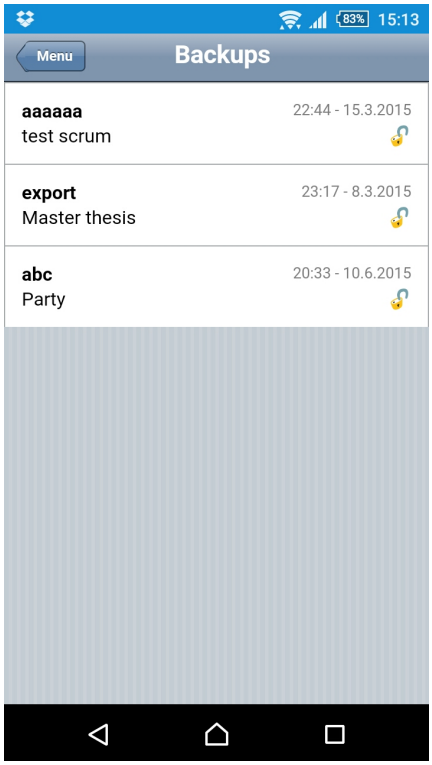
- Voting phase – users can vote their estimation for the story and they see on the list who already voted, but not their actual estimation. Scrum master can end the voting, when enough users voted. (screenshot 12 and 13)
- Results – When voting ends all users see estimations of others and can discuss it. Scrum master can then start the poker again for the story or move to another story from the backlog. (screenshot 14)



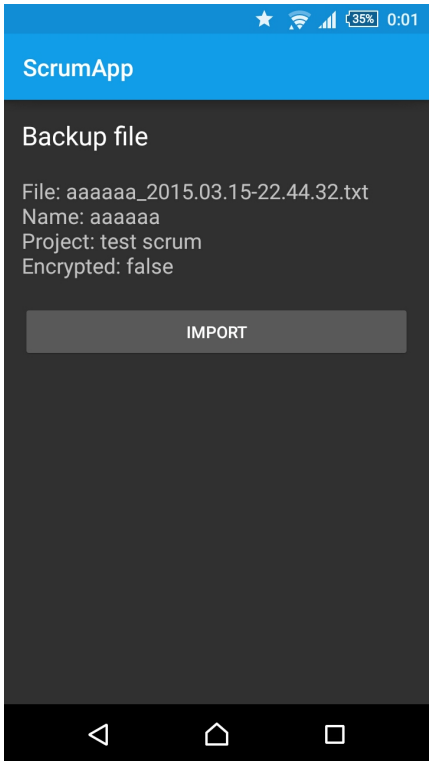
Screenshot 12: Voting *Screenshot 13: Estimation* *Screenshot 14: Results*

2.10 Backups, File import and export

From the project list you can export project's data of a local or remembered project. It will then be available in the backup list (screenshot 15). Here you can import the backups back overwriting the current data or send it to someone (email or iTunes on iOS, sharing by any compatible application on Android). You can also import a backup you received from someone (in iTunes on iOS or by sharing it with the ScrumApp from another application like file managers or email on Android, screenshot 16).



Screenshot 15: Backups



Screenshot 16: File import