University College of North Denmark

AP Degree Program - Computer Science

5th semester / Class DMAJ0914 / Group 4

Final project report

The time management software

Group Members:

Andrej Ivankov

Miroslav Pakanec

Supervisor: Anita Lykke Clemmensen

Submission date: January 9th 2017

University College of North Denmark

AP Computer Science / DMAJ0914 – Group 4

Abstract:

This project consists of researching, planning, designing and implementing the C# language software and database for the time management system, with aspects of different types of technology, agile system development and programming points of view.

Preface:

This project is done by fifth semester students of the Computer Science course. The group consists of only two members, aged 21 and 32, respectively coming from two different European countries – Slovakia and Croatia.

The focus of the report is creating a completely new software system where the main focus is the time management system which will be able to optimize time spent to arrange different events. The system will be able to handle all relevant information regarding users, businesses, locations, spots and events.

Problem statement:

Time is a limited and precious resource for all. People have been meeting one another on a daily basis for all kinds of reasons. However, today the hectic era is putting a pressure on people to perfect their time management, be organized and punctual. Whether it is a conference, cultural event, family or business meeting, there is a need for digitalised space where one could get a complete overview of his and other´s activities and more efficiently manage his time.

Is it possible to create a system that could incorporate the time and event management in a way that users are capable to create and view the events that fit their needs?

Table of contents:

1 Introduction

The report is focused on creating a software system that will be able to handle event time management, both from the business and from the personal point of view and free time or cultural or self-realization. There are several phases for developing this kind of a software product, therefore this report will focus only on the initial phase - building up a workable version of the system. To be more precise, the base of the system, which would serve as a starting point to build upon. To be able to present all of the information about this first phase, the report is divided in four parts:

1. Research and planning
2. System overview
3. Development of the system
4. Future steps

Every system has a part which starts before the actual development – the research and planning part. This part focuses on the actual research about the product that is going to be developed. This part also finds out the business value, which would justify the creation of the system. The research consists of going deeper in the analysis of the problem statement, finding out the target group of potential users, followed by the initial business research. Next step is to agree upon the method that is going to be used, as well as to set up the system requirements.

Second part of the report focuses on the architecture and technologies, which will determine the shape of the system. This part is more theoretical, because it describes how system will utilize certain technologies. This point is crucial not only to understand the advantages of selected technologies, but also to realize possible potential issues of the system. This part ensures that the product solution can be developed in the best possible way.

After this first two parts are done, the next part is developing the product solution. Development part describes the 7 weeks working process on the software development. It explains all stages behind iterations – planning, management, and complete overview at the end.

The final part is the theoretical part as it focuses on what will happen in the future. To be more precise, this part describes the phases which will happen afterwards, from starting a company, recruitment process, and more. This in detail describes what has to be done to launch the final version of the product to the market, in the phase by phase overview.

2 Research and planning

The part which brings the biggest responsibility for a successful project is the initial part – Research and planning. This part starts prior to the actual start of the software development and focuses on what and how is the project going to be developed in the upcoming time.

The fact is that the whole project has to be concluded within 8 weeks. So the part of actual research and planning will be done in the first week and it will cover several key areas: going deeper into the problem statement research, business research and related work, the choice of the software development method, and it concludes with the system requirements.

2.1 Detailed problem statement

Time is a limited and precious resource for all. People have been meeting one another on a daily basis for all kinds of reasons. However, today the hectic era is putting a pressure on people to perfect their time management, be organized and punctual. Whether it is a conference, cultural event, family or business meeting, there is a need for digitalised space where one could get a complete overview of his and other´s activities and more efficiently manage his time.

All events have some things in common:

- Finding the perfect location
- Suitable time to fit everyone
- Remembering the event

Going in a bigger depth, it is obvious that there is not a tool that connects all of the above mentioned things together in a simple and an efficient fashion. This brings us to the following statement:

Would it be possible to create a program which would reduce the current time needed for arranging and organizing events, making it easier to attend the events, optimizing the time management process for any individual or a business, and utilizing the event information?

As the statement above is quite broad for the project, it is narrowed down to:

Is it possible to create a system that could incorporate the time and event management in a way that users are capable to create and view the events that fit their needs?

2.2 Related work

The section related work focuses on the business aspect of the research. For any product solution, there are certain things that need to be taken into consideration. In this section, research will go into depth about the current market situation and the target groups, which could be using time management solution developed in this project.

2.2.1 Market research

Currently there are several similar products on the market. They can be classified in a couple of subcategories:

- Calendars. There are many companies providing advanced calendars where users can create the meeting that they want to have. Although the items are easy to add in, the location has to be added manually. In addition, the attendance of the participants is questionable. Products on the market with biggest share in use include: Google Calendar, Microsoft (Outlook) Calendar, IBM Connections calendar
- Meeting management systems. Tools that go one step forward and focus on the meetings, for example: setting up agendas, minutes, moderators, reports and similar business orientated items. While these systems are a bit more complicated, they are still simple enough to be used. Some of them are: MeetingSphere, BoardPad, BoardPacks, MindManager, MeetingBooster.

Tools that are currently on the market show the necessary system characteristics and obviously, some specifications are repeating:

- Security. In most cases, users want their information secure and visible only to them.
- Usability. One of the most common specification – a tool should be as user friendly as possible. Some of the products simply confuse users with all the information shown on the screen. “Simpler” tools have the best success on the market (e.g. Google Calendar)

2.2.2 Business point of view

The main reason to start this project is to change the way how people meet. To be able to know how to achieve this, it is important to know how this system could change it. The answer is simple: Connect people with locations. Then it is needed to identify who would use the system (this is analysed in the next subchapter – target groups).

Next, it is necessary to find out whether it is possible to sell the product to the users. After careful consideration, it has been decided that the focus will be the website with several types of packages. The initial one will be the freeware – free version of the software to be used. The business value of the system will be in the user’s need to buy additional software features.

The only real question that remained was: What is the plan? The scope of launching this project is quite big, therefore it was decided to split it into three phases:

- Foundation phase
- Design and release phase
- Further development phase

In the first phase, this project, the main goal is to set up the core of the software. There it will be determined whether it is possible to create a system that can handle all the information needed for the development of the product. Due to this, there are two types of goals – the mandatory (the ones that need to be done so that the product could be considered viable), and the secondary (the system functionality, which would be good to have). The mandatory goals are:

- Implementing users (registration, login, and edit users)
- Implementing businesses (create, view, and update business information)
- Implementing locations (create, view, and update locations)
- Implementing spots (create, view, and update spots)
- Implementing meetings (create, view, and update meetings)

The secondary goals are:

- Implementing calendars (view and compare different calendars)
- Implementing friend lists (CRUD friend lists)
- Gathering statistics (view different statistics about users, locations and spots)

Although the business value of the product had been set up, this was just beginning. These values have to be transformed into the functional requirements for the project, in order to accomplish them.

2.2.3 Target groups

Target group is a group of customers at whom a business would like to focus its marketing efforts and sell their products. In this project the main target group are people that would like to book, create or attend an event. It is possible to specify the main target group in four categories:

- **Specific target group 1 (STG1):** private user that would like to book or attend an event with a friend
- **Specific target group 2 (STG2):** private user that would like to book or attend an event with multiple users
- **Specific target group 3 (STG3):** employees that would like to book or attend an event with other employees or attend professional events
- **Specific target group 4 (STG4):** company owners (managers) who would like to oversee the spots and their employees (STG4 will be referred as business admin)

To understand how specific target groups would act, some hypothetical situations for potential actors (personas) using the system were prepared. The personas help to gain better understanding of the system requirements – sometimes it is easier to write down the user stories with having a specific person in mind. After careful consideration 6 different scenarios were created:

- Persona 1: User wants a meeting with a friend (all)
- Persona 2: User wants to create an event for friends (all)
- Persona 3: User wants to have a meeting with a business partner (STG2, STG3, STG4)
- Persona 4: Manager wants to see what events did his employees had (STG4)
- Persona 5: User wants to create new spots (or locations) (STG4)
- Persona 6: User is looking for an event to join (all)

It can be noticed that in the parentheses above it is specified to which specific target group this persona is referred to. More specific script for this personas could be described as follows:

- Person wants to grab a beer with a friend (Persona 1)
- Person wants to make a dinner with several friends (Persona 2)
- Person wants to make a business lunch with a business partner (Persona 3)
- Manager wants an overview of all sales department events (Persona 4)
- Manager wants to create available event places (spots) to be booked (Persona 5)
- Person wants to look for different concerts in the city and join one (Persona 6)

The information gathered here will help to develop a system so that multiple different users could use it in the best possible way for them.

2.2.4 Initial planning

One of the most interesting parts of the research was designing the system business value. The whole planning process started with making mock-ups with a pen and paper, which produced an initial view of the system.

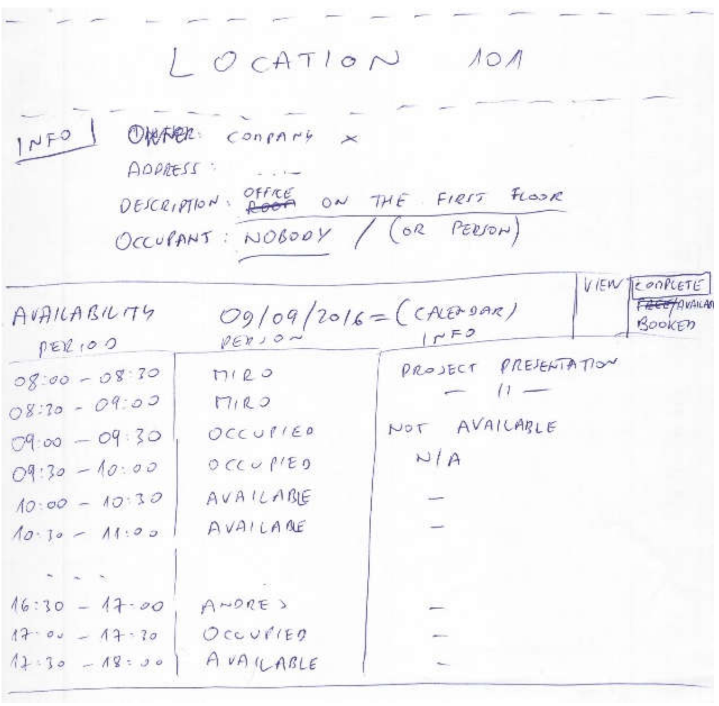


Figure 1 Mock-up created early in the planning phase

Initial view led to a discussion about the booking procedure. It was decided that there should be an important distinction between the physical location and the location that is rented out. Each physical location can “rent out” rooms, halls, or even chairs and it was obvious that it will be hard to make different classes for all of them. Guided by the fact that the system needs to be easy to use, it was decided to name the system simply the “spots”.

The next decision was to make one class with the basic information (name of the spot, its capacity and visibility). With these three attributes, each spot will have the minimum requirements to be used in the system. Unique name will make a spot familiar to the location. The capacity will state the maximum number of participants that can meet at that spot. The visibility will determine whether the spot will be seen by everyone else or it is going to be used only for internal use within the business. Decision about spot’s information is made by the spot owner.

Each spot will be connected to a physical location which will have the address, zip code, city, country. For now, the location is connected with a business, although in the future, it will be enabled to be created by a “regular” person.

After setting up the “business base”, it was time to go to the drawing board and try to sketch the whole overview of the system. Here the main goal was to try to see how to set up the relationships between entities.

All of this material was prepared for the development the technology requirements of the system. The first phase of the plan can officially start.

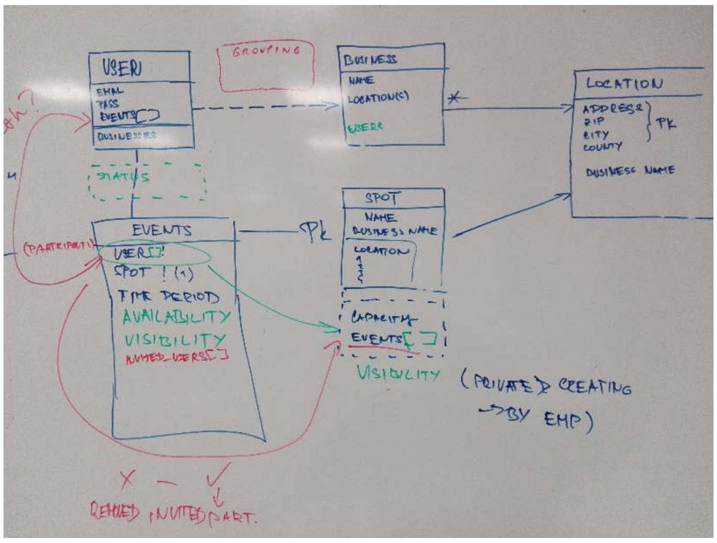


Figure 2 Drawing created when brainstorming about the architecture

2.3 Method choice

There are two specific ways to manage the process of creating the software solution. The plan driven development with predefined and planned scope at the beginning of the project and the agile development where the scope and project plans are created and changed during the entire project.

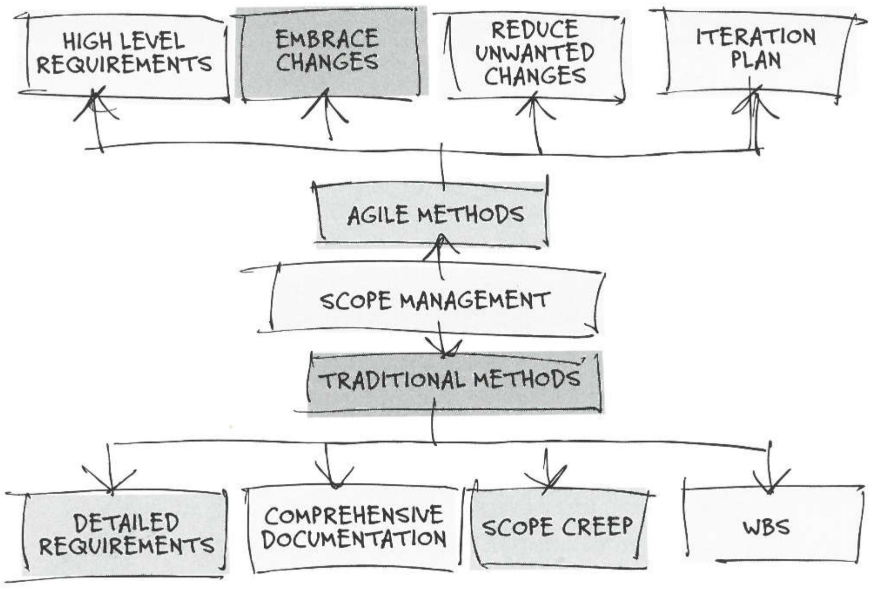


Figure 3 Differences between the agile and plan driven scope management[1]

Agile methodology was developed with the purpose of finding better ways how to develop the software. To understand the methodology, a Manifesto was created with four firm statements, where each statement compares two values. In the statements, agile values more the ones on the left side[2]:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to the changes** over following a plan

[1] Figure from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 61
[2] Agile manifesto from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 45

For this project agile methodology will be used. There are several reasons behind choosing the "new", agile methodology over the "old", plan driven methodology, and they are:

- Unknown requirements - creating software is hard, but creating it from a scratch is even harder. The main problem is that, although most of the requirements are known from the beginning, it is impossible to predict how the development will proceed and what new requirements could be discovered and demanded during the process. This means that a long-term detailed plan is almost impossible to make and the agile methodology is a better solution since it is more adjustable to the changes.
- Essential communication – the project is developed by a two-member team that need to create a product in a short period of time. The agile methodology is a better choice because it supports short iterations and focuses more on the communication between the members.
- Working solution –one of the main ambitions was to have the functional base product at the end of the project, instead of having comprehensive documentation.

One of the most important advantages of the agile development is the fact that projects can adapt the Agile in different shapes because they need to choose the best way to achieve the best possible result for the end product.

Choosing the methodology type does not mean that everything is set. There are several different types of agile methodologies which differ in the approach to the development. However, all of them include these activities:

- Requirement specification
- Design
- Implementation and testing
- Evolution

There are several types of Agile methods and methodologies. After comprehensive research and careful consideration, it has been decided that Scrum in combination with Extreme Programming and Kanban will be used as the team already has some experience with these methodologies and would like to explore whether it is possible to use them in a small team environment The next few subchapters describe the methods that were used in this project, as well how they were used.

2.3.1 Scrum

The Scrum is a method which covers project management. It has five values[3] (Focus, Openness, Respect, Commitment and Courage) which address the human aspect of the method. The Scrum consists of three important artefacts (Product backlog, Sprint backlog and definition of done) and four important events (Sprint, Sprint planning, Sprint retrospective and Daily Scrum) that are done in iterations by a team. Iterations are suggested to be done in a period of one to six weeks.

Roles in the Scrum are:

- The product owner is responsible to maximize the value of the product, represent the stakeholders as their voice, and to prioritize the product backlog. The product owner is responsible for the project outcome.

[3] Quote from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 78

- Team members are members of a self-organizing team who determine how they will work. They deliver the product.
- Scrum master ensures that the team focuses on the goal and acts as a liaison between the product owner and the team. Scrum master tasks are to advise the product owner, support the team and enforce the rules and processes.

In the Scrum, the product owner sets up the priorities of items that should to be developed into something called the product backlog. The team decides how much of that product backlog will be done in an iteration, depending how much time do they have available. At the end of the iteration, the team shows workable software to the stakeholder. For the next sprint, the process is repeated, and it starts when the product owner prioritizes the items in the product backlog.

2.3.1.1 Product backlog

Product backlog represents the requirements for the product which are sorted by the priority of what should be developed first. In this project, product backlog consists out of different user stories that have to be done so that the system can be developed.

The project backlog exists as long as the whole project exists. It is important to stress out that the priority could change, depending what is the most important for the development. The product backlog for this project can be viewed in appendix on page 83.

2.3.1.2 Sprint

Sprint is a period of time, usually two to four weeks. During this time, team members try to finish the tasks set in the sprint backlog. All sprint iterations should last the same amount of time, and they should start and end at a scheduled time. Having the same duration is important from several aspects including: team discipline, sprint planning and measurability.

Sprint involves daily stand-up meetings (described in chapter 2.3.4) where team members share the status of their work, prepare for the sprint review, and of course, they work on solving the sprint tasks. End result for every sprint should be a workable software. In this project, one sprint iteration has a duration of one week. There is one planning iteration (described in chapter 4.1) and six development iterations (described in chapters from 4.2 to 4.7).

2.3.1.3 Sprint planning

Sprint planning meeting is held at the start of the each sprint[4]. Everyone from the team are present (product owner, scrum master and team members). The meeting consists of several phases:

- Firstly the product owner sets the highest priority of the remaining user stories
- Secondly the team asks product owner questions to find out more about the user stories
- Finally the team determines how many user stories will be moved to the sprint backlog depending on how many story points they can accomplish in the sprint.

It is important that the user stories are put in the sprint backlog, according to the priority set by the product owner, stopping at the point where the team thinks it is a maximum of story points that they can accomplish. If the team is ahead or behind the schedule, user stories can be added or removed from the sprint backlog, if the product owner agrees.

[4] Quote from Mike Cohn: *User stories applied for agile software development*, page 168

2.3.1.4 Sprint backlog

Sprint backlog is a list of tasks which need to be done to accomplish a certain user story in a current sprint. Sprint backlog derives from the sprint planning meeting. At the sprint planning meeting the team determines the quantity of user stories they can accomplish from the Product Backlog. The team breaks down the requirements of every user story into the tasks. Sprint backlog is created and maintained by the team during the sprint. It is important to put into the sprint backlog only what can be completed. In rare occasions, sprint backlog can be changed if the whole team agrees. This depends on the time that they have on hand. At the end of the sprint, sprint backlog comes to the end of its lifespan and the new one will be created for the next sprint.

2.3.1.5 Sprint review

Sprint review meeting is held at the end of a sprint, where the team will demonstrate the working solution. This meeting can be open for everyone who is interested in the project. To make sure that the sprint review does not take too much time from the team's schedule, tasks for the demonstration are prepared in the sprint planning meeting. Preparation tasks (e.g. preparing a demo workflow script) should not take too much time to do, their aim is to alleviate the stress connected to the demonstration.

One of the most beneficial parts of the sprint review meeting is the Q&A session – part of the meeting where participants can ask questions to the development team. It is common that this session brings out suggestions, which should be taken into consideration for the future development process.

2.3.1.6 Sprint retrospective

Sprint retrospective happens at the end of every sprint. The purpose of this meeting is to look back on what the team has done in the Sprint in order to adapt their methods and teamwork. To be able to do a quality retrospective meeting, evaluation questions like these are used:

- What went well?
- Were there any issues?
- Were there any surprises?
- Which areas could be improved in the future?

The best way to answer these questions is to ask every team member to answer them on a sticky note. When everyone writes down the notes, they are put on a board. Looking at the board, similar items are aligned in groups and circled. It can be that there are some outliners (items outside of the groups), but usually, after brief team discussion, outliners are either from misinterpretations or misunderstanding, rather than from conflicting opinions. The final step for the team is to address the circles on the board and plan actions for the upcoming sprints to improve these areas. This approach to retrospective is good as[5]:

- It avoids putting anyone on the spot
- There is plenty of physical motion, which keeps the team alert and active
- It is tactile and visual, which makes it more interesting
- The team is able to immediately gauge the collective priorities by viewing sticky-note groupings on the board

[5] Quote from Ilan Goldstein: *Scrum shortcuts without cutting corners: agile tactics, tools & tips*, Page 126

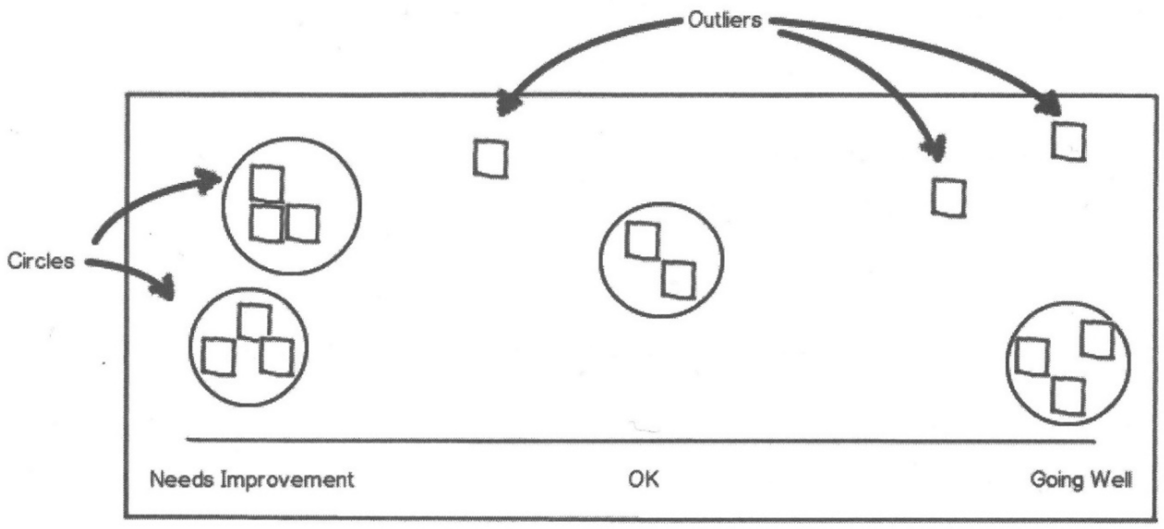


Figure 4 Sprint retrospective – the "circle approach"[6]

The benefits of conducting iteration retrospectives are[7]:

- Ability to develop and deliver the software improves
- Team grows closer and more cohesive
- Being honest and open about successes and failures increases confidence and team is more comfortable with the changes

2.3.1.7 Definition of done

The definition of "done" is derived from human tendency not to completely finish their actions. That is why it is important to agree on the criteria for "done" during the iteration planning.

In this project, the "definition of done" is accomplished if the user story:

- Is written using the INVEST method
- It was estimated
- It was prioritized

2.3.2 Extreme Programming

Extreme Programming (XP) is a method that focuses on the development practices. It is based on values of simplicity, communication, feedback, courage, and respect[8]. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation[9].

The XP is usually done in the three-week iterations where each of them result in running and tested code which is of direct use to the customers[10]. XP uses user stories written by the customer which discover the functionality. They are developed by splitting the user story in tasks, which are done within one iteration.

[6] Figure from Ilan Goldstein: *Scrum shortcuts without cutting corners: agile tactics, tools & tips*, Page 126
[7] Adapted from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 142
[8] Values from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 84
[9] Quote from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 84
[10] Quote from Alistair Cockburn: *Agile Software Development*, page 199

Working together, developers and the customer can always re-prioritize the iterations, add new user stories or tasks, remove user stories or redefine tasks. One team member is designated as the coach – a person who works with the team members to see how they are using their key practices.

XP has several different roles, which are[11]:

- XP coach has the tasks to stay on Process, help the team learn, mentor the team, and reinforce the communication
- XP customer writes the system features , defines what to build and creates the acceptance tests
- XP programmer writes the code
- XP tracker collects the information and tracks release plans, iteration plans and acceptances tests
- XP tester defines, automates, sets up and implements the acceptance tests.

The XP consists of 12 practices that implement the XP values[12]:

- Planning Game – The programmer estimates the effort needed for the implementation of customer stories and the customer decides the scope and timing of releases based on the estimates
- Small releases – an application is developed in a series of small, frequently updated versions
- Metaphor – the system is defined by a set of metaphors between customer and the programmers which describes how the system works
- Simple design – The emphasis is on designing the simplest possible solution that is implemented and unnecessary complexity and extra code are removed immediately
- Refactoring – it involves restructuring the system by removing duplication, improving communication, simplifying and adding flexibility, but without changing the functionality of the program
- Pair programming –code written by two programmers on one computer
- Collective ownership – no single person owns or is responsible for individual code segments rather anyone can change any part of the code at any time
- Continuous integration – a new piece of code is integrated with the current system as soon as it is ready. When integrating, the system is built again, and all tests must pass for the changes to be accepted
- 40 hour week – No one can work two overtime weeks in a row. A maximum is a 40-hour working week, more is treated as a problem.
- On-site customer – customer must be available at all times with the development team.
- Coding standards – coding rules exist and are followed by the programmers to achieve consistence and improve communication between the development team.
- Test driven development – an approach where the tests are written down before the code.

[11] Quote from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 89
[12] Adapted from multiple sources

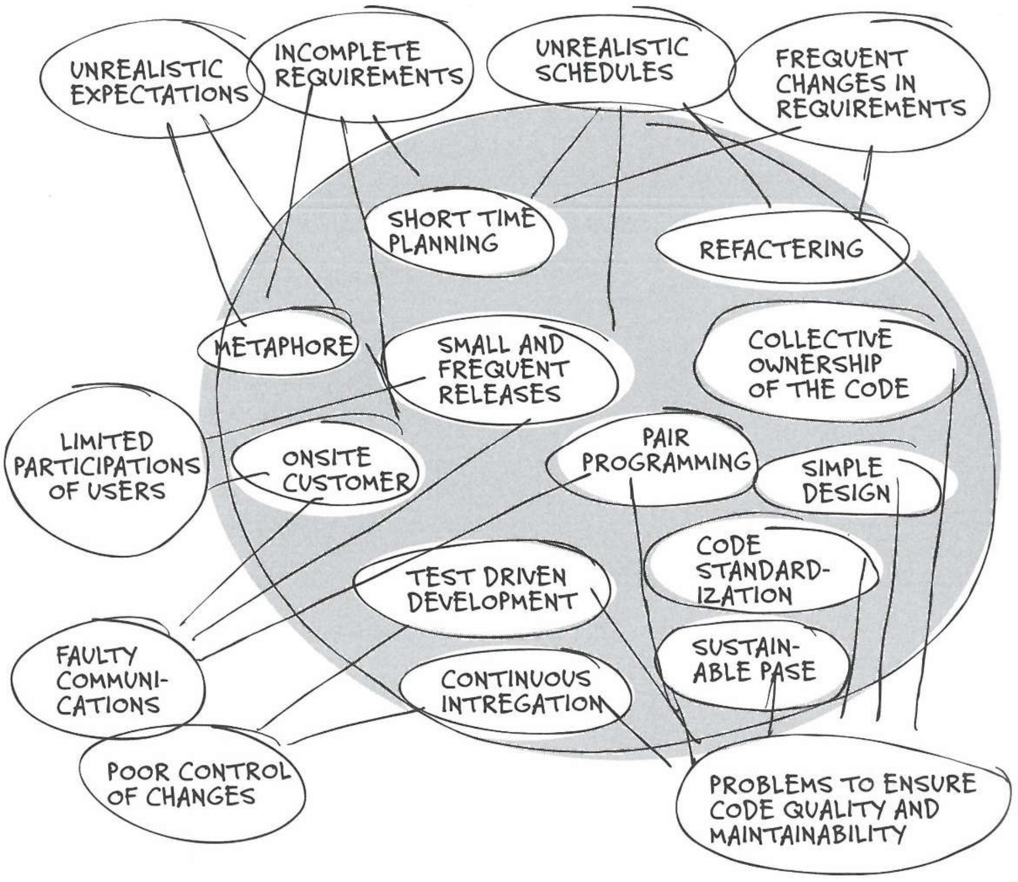


Figure 5 How the XP practices prevent problems of managing software projects[13]

For the XP to be successful, it is not necessary to follow all 12 practices, it is important that the team follows the principles of XP. That means that the team[14]:

- Provides the rapid feedback to its customer and learns from that feedback
- Favours simplicity and always attempts a simple solution before moving to a more complex one
- Improves the software through small, incremental changes
- Embraces change because they know they are truly adept at accommodating and adapting
- Insists that the software consistency exhibits the highest level of quality workmanship

2.3.3 Kanban

Kanban is a change management method, which consists of a set of management practices for software derived from Toyota Production Systems and Goldratt's Theory of Contracts. It is defined by three principles[15]:

- Start with what you know
- Agree to pursue incremental, evolutionary change
- Respect the current process, roles, responsibilities and titles

[13] From Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 88
[14] From Mike Cohn: *User stories applied for agile development*, pages 241/242
[15] From Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 90

To achieve the success with the change management method, there are five core properties:

- Visualize the workflow
- Limit work in progress
- Manage and focus on the flow
- Make process policies explicit
- Improve collaboratively

Kanban contains a wide range of optional aspects such as team size, burn down charts, cross functional teams. Kanban is usually used in conjunction with other Agile methods.

2.3.4 Decision

After carefully going through all of the methods, and the team's previous experience with the Scrum and the Extreme Programming (XP) it was decided to continue with the same combination as in the previous project. However, with a different approach, because this project has only two team members.

The main reason is that Scrum is going to be used for the planning and controlling of the development process, while XP is going to be used for the planning and the development of the software. This is a really good combination as some parts of these methods overlap with each other. The best examples for this overlaps are:

- Demonstration for stakeholders in Scrum can be considered as an expansion to XP
- Stand-up meeting in XP is the same as the stand-up meeting in Scrum
- Roles are basically the same: Scrum's product owner and XP's customer, Scrum master and XP's coach, while the Scrum's team members can be merged with XP's programmer, tracker and tester

Scrum method is used completely as it is, with only one issue regarding roles, due to the number of the members. The minimum number for the Scrum are three people – due to the three different roles. That is why it has been decided that the product owner is Andrej, the team member is Miroslav, while the Scrum master role is alternated between both of them. It is not a perfect solution, but due to the member size limitation it has to work.

On the other hand, the number of the XP practices is reduced. To be more precise, they are put in divided into two groups– the ones used in the project, and the ones that cannot be used.

The list of the practices that used in the project are:

- Planning Game –working as a team, Andrej creates user stories and both Miro and Andrej estimate each of them
- Small releases – during the process, the common idea is that after every iteration there is a functional software that can be tested and used
- Simple design –the simplest possible solution is developed. It could happen that it is hard to see if there is some unnecessary code. To deal with this, extra checking of the code is done every week, so the extra code could be removed
- Refactoring – it involves restructuring the system by removing duplication, improving communication, simplifying and adding flexibility without changing the program functionality

- Collective ownership – although most of the code will be designed by Miroslav, Andrej has the tasks to check whether all of the code is written as it should be and suggest all possible changes before the end of the iteration
- Continuous integration – as mentioned in the small releases, after every iteration new addition to the system is added. When integrating, the system is built again, and the tests are performed to make sure that everything is in order
- 40 hour week – the team is not to exceed the maximum of 40 hours of working every week. This is described in chapter 4.1.2
- On-site customer – As already mentioned in the Scrum chapter, there is an on-site customer at all time during the project
- Coding standards – Before the coding starts, the code standards are prepared to make sure that code is "clean" and "easy to read". More about code standards in chapter 2.4.3.1

On the other hand, there are three XP practices that are, unfortunately, not going to be applied:

- Metaphor –as there are only two members in the team and the metaphors could complicate things much more than make them easier
- Pair programming –as the team will be working long-distance it is impossible to use one workstation or one common area
- Test driven development – the team will try to follow this practice, but with the knowledge gained from the previous project, this approach usually derives a lot of time

Daily routine for the team consists of a meeting every morning, lasting no longer than 15 minutes. This meeting is known as the stand-up meeting or Daily Scrum where three questions are asked to determine the progress of the project. They are[16]:

- What did I do yesterday?
- What will I do today?
- What obstacles are in my way?

The answers to these questions give the feedback, whether the adjustment needs to be done for that iteration in order to reach its goals. To be able to see how the project is progressing and adapting to the changes, it is necessary to track the process. For this, two important tools are used – the Kanban boards and WIP limits.

Kanban board is a task board that shows the status of the tasks. It usually shows which tasks have to be done (To Do column), which are currently in progress (Doing column) and which tasks have been finished and implemented (Done column). WIP limit is a strategy for preventing bottlenecks in the software development[17]. By using this strategy, developers focus is on the small amount of tasks, which yield in higher quality of the software.

In this project, a more detailed Kanban board is used. It consists of the sprint backlog (list of user stories for that sprint), new column (list of the tasks needed to be done for that user story), active column (list of tasks currently developed; limited to three tasks per sprint) and closed column (list of tasks that have been tested and implemented).

[16] Quote from Mike Cohn: *User stories applied for agile software development*, page 171
[17] Quote from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 138

Even though in the past team used different tools as Trello (virtual task board where tasks are moved through the boards) and physical whiteboard (where post-it papers are moved through the board), for this project it is decided to use Virtual Studio Team Services (VSTS) to track the iteration progress. VSTS is a software that is integrated with Virtual Studio, which means that the tasks can be seen inside of the Visual Studio which is used for programming. VSTS also creates the whole product backlog and track project progress that makes the tracking much easier and more informative.

To be able to start with the tasks, beforehand it is important to know all of the requirements beforehand. For that purpose user stories (more information in chapter 4.1.1.) that cover all functional requirements for the project are created. To be able to create the best possible user stories, the INVEST method is used. INVEST acronym was for the first time suggested by Bill Wake in 2003 for writing the user stories. INVEST stands for:

- Independent –user stories do not overlap and can be planned and implemented in any order
- Negotiable – user story details are negotiated in dialogue (short is good)
- Valuable – user story contains business value for the customer
- Estimable – every user story is estimated by the team
- Small – making user story small ensures that additional breakdown or splitting is not required
- Testable – all user stories should be testable by the end user as soon as possible.

At the end it is important to specify some of the advantages and disadvantages as a part of a risk assessment analysis for this project’s development.

Advantages:

- Face to face communication – fast answers to questions
- Better requirement management
- Focus on risks and changes

Disadvantages:

- Absence of documentation
- Potential conflict between the small size of an increment and the size of important functionality
- Challenging from the architectural point of view
- “Simplicity” costs resources (refactoring)

2.4 Quality management

Software quality management is concerned with ensuring that software has a low number of defects reaching the required standards for maintainability, reliability, portability and so forth. It includes defining standards for process and products, and establishing processes to check that these standards have been followed[18]. Software quality management techniques are derived from the manufacturing industries where terms quality assurance and quality control are widely used. Quality assurance is the definition of processes and standards that should lead to high-quality products, while quality control is applying these processes to weed out the products that are not of the required level of quality[19].

In the software development there are two main types of the quality assurance: Traditional and Agile type. As this project is developed with agile development, the agile quality assurance is used as well. There it is stated that the project management specifies the scope of the system as a variable, while amount of the time and costs as fixed items. That means to be able to determine the quality of the product it is needed to be able to measure the quality of the features in it.

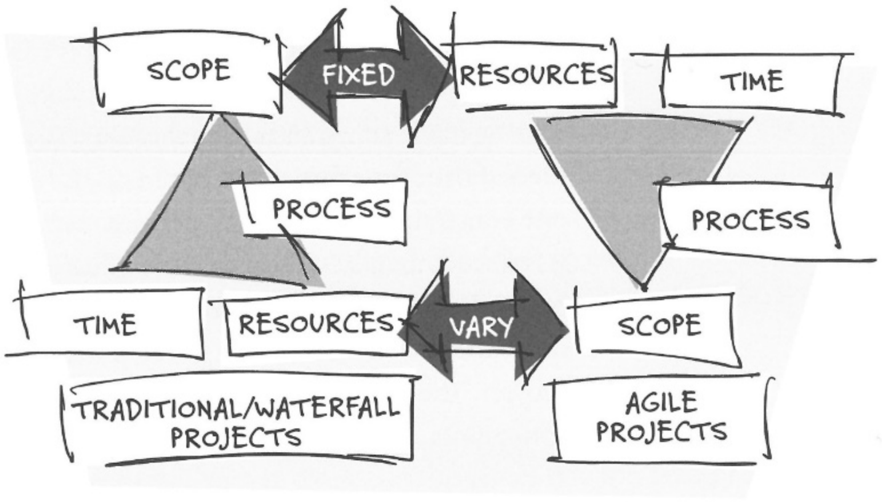


Figure 6 Comparison of traditional and Agile project management[20]

To be able to measure the quality of the software product, code quality will be achieved through continuous integration, refactoring, standards, manual testing, non-functional and functional requirements, described in the upcoming subchapters.

2.4.1 Continuous integration

Continuous integration is one of the Extreme Programming practices where all code changes are checked in, the system is built and tested as often as possible. By applying the continuous integration, team can decrease the amount of the integration problems at the end of the project. Example of the continuous integration applied in the project can be viewed in figure 64 in the appendix.

[18] From Ian Sommerville (2016): *Software Engineering*, page 727
[19] Adapted from Ian Sommerville (2016): *Software Engineering*, page 701
[20] From Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 60

2.4.2 Refactoring

Refactoring is also one of the Extreme Programming practices, where small changes are made to the source code to improve its design. To prevent the decay of the code, a state which happens with time as the code gets entirely rewritten, refactoring is the technique used to make small changes to already existing code without changing its external behaviour. Because the changes are small, the system is usually fully working after each change.

2.4.3 Standards

Software standards play a crucial role in the software quality management. There are two types of the software engineering standards: product standards and process standards. Product standards are connected to the software product that is being developed and the process standards determine the processes to be done during the development.

The focus of this chapter will be on the product standards. They include the document standards (e.g. structure of requirement documents), documentation standards (e.g. standard comment header), and coding standards, which define how the programming language should be used[21]. The product standards used in this project are described more in the upcoming subchapters as well as in chapters 3.2.1, 3.2.2 and 3.3.

2.4.3.1 Code standards

Coding standards have two functions in this project – they are a tool that improves quality and they are also one of the Extreme programming practices. They provide rules and conventions that team members should follow when writing the code. This is done because the source code of the system is collectively owned by the whole team. They are written down before the developers start programming. Some of the code standards used in the project are named below.

Server side naming conventions:

- Namespaces start with a capital letter. Every following word starts with a capital letter
- Classes start with a capital letter. Every following word starts with a capital letter
- Interfaces start with a capital “I”. Every following word starts with a capital letter
- Methods start with a capital letter. Every following word starts with a capital letter
- Properties start with a capital letter. Every following word starts with a capital letter
- Private fields start with underscore. Word followed after the underscore starts with a lowercase letter. Every following word starts with a capital letter
- Constants start with a lowercase letter. Every following word starts with a capital letter

Client side naming conventions:

- View files start with a capital letter. Every following word starts with a capital letter
- Partial view files start with underscore. Every following word starts with a capital letter
- JavaScript files start with a lowercase letter. Every following word starts with a capital letter
- JavaScript functions start with a lowercase letter. Every following word starts with a capital letter
- Variables start with a lowercase letter. Every following word starts with a capital letter

[21] Quote from Ian Sommerville (2016): Software Engineering, page 706

- HTML attributes start with a lowercase letter

Database naming conventions:

- Document database name starts with a capital letter. Collections start with a lowercase letter. Every following word starts with a capital letter.
- Collections start with a lowercase letter and consists of a single word.
- Every document field except the standard identifier starts with a capital letter. Every following word starts with a capital letter.
- Document identifier starts with an underscore and is followed by a single word starting with a lowercase letter.
- Tables start with a capital letter. Every following word starts with a capital letter.
- Columns start with a capital letter. Every following word starts with a capital letter.

Automated test standards:

- The name of the folder in which a test file is located, represents a page name it operates on.
- The name of the test file starts with a capital letter. Every following word starts with a capital letter.
- The name of the test file describes the functionality that is tested in the file.
- A test file can have no more than one test class. The class is marked with attribute "TestFixture".
- At the top of the class there is a declaration of a test case object, (if necessary) followed by other global variables.
- Every test class has a setup and teardown method.
- Every test method is marked with the attribute "Test".
- Every test method has a summary describing what the method tests.
- The name of the test method consists of the following parts - test functionality, test name and test outcome. Parts are divided by an underscore.
- Every test step is marked with a comment above it.
- Test step comment starts and ends with a dash separator. The first line of the comment describes the characteristics of the test step followed by a colon. Following line describes the test step. If there are more test steps of the same characteristics in a row, comments are stacked similarly inside the same dash separators. The order of the test step comments and test steps has to be kept.
- Every test method ends with an execution of the test case.
- Every test class can only have one test case execution.

2.4.3.2 Solid principles

Solid principles are a collection of design principles that help to create a good software architecture. SOLID is an acronym and it stands for[22]:

- S – Single responsibility principle
- O – Open/closed principle
- L – Liskov substitution principle
- I – Interface segregation principle
- D – Dependency inversion principle

Single responsibility principle (SRP) states, that a class should only have one reason to change. This means that there should only be one requirement that causes a class to change[23].

Open/closed principle (OCP) states, that a software entity should be open for extensions, but closed for modification[24]. One of the biggest benefits is that unit tests do not break and thus do not need to be rewritten.

Liskov substitution principle (LSP) states that objects in a program should be replaceable with instances of their subtypes without altering the correctness of the program[25]. This means that a subclass should behave in a way that it does not cause problems when used instead of a superclass.

LSP principle follows with a number of rules[26]:

- Any method arguments in the subclass can be accepted in a base class (contravariance of method arguments in a subclass).
- Return value in subclass is the same as the parent's return type or is its descendant (covariance of return value in a subclass). It is impossible not to follow this rule in C#.
- No new exception types are allowed to be thrown, unless they are subclasses of previously used ones.
- Preconditions are not strengthened in a subclass. Things that are not expected from the base class, are not expected from the subclass.
- Postconditions are not weakened in a subclass.
- History constraint - an immutable class cannot be made mutable (things that cannot change in the base class, cannot change in the subclass)

All in all, the LPS rules help not to change a class in a way that would break the application. In addition, when a new implementations are added, the existing system will not break.

Interface Segregation Principle (ISP) states that many client-specific interfaces are better than one general-purpose interface[27]. The client should not be forced to depend upon interfaces that are not going to be used. Breaking interfaces into smaller pieces makes them easier to implement, offers more control and avoids unneeded code.

[22] Acronym from http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code (Retrieved in 12/2016)
[23] Quotes from https://www.youtube.com/watch?v=gwIS9cZlrhk (Retrieved in 12/2016)
[24] Qoute from http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code (Retrieved in 12/2016)
[25] Qoute from http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code (Retrieved in 12/2016)
[26] Adapted from https://www.youtube.com/watch?v=gwIS9cZlrhk (Retrieved in 12/2016)
[27] Qoute from http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code (Retrieved in 12/2016)

Dependency inversion principle (DIP) has two key points[28]:

- Abstractions should not depend upon details
- Details should depend upon abstractions

This means to use the same level of abstraction at a given level. Interfaces should depend on other interfaces. Concrete classes should not be added to method signatures of an interface. However, interfaces should be used in class methods. Ensuring that classes do not depend upon specific implementations eases maintainability and adaptability.

2.4.4 Manual testing

It is impossible to fully automate all tests for all environments[29]. For that reason, manual testing can be performed with a main goal to find bugs. Manual testing can also be done to identify missing test cases, check if the user story idea has been understood and provide feedback about the implemented feature.

The testing during the development phase improves the quality of a product while it is being developed as mistakes can be noticed, the feedback can be collected, moreover, there is not that much pressure to do all of the testing at the end of the project. All of the "negative" feedback can be used to create new items for the product backlog.

2.4.5 Requirements

The requirements for the system are the descriptions of the services that a system should provide and the constraints on its operation[30]. All system requirements are classified as functional or non-functional requirements[31]:

- Functional requirements are statements of services that the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.
- Non-functional requirements are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and constraints imposed by standards. Non-functional requirements often apply to the system as a whole rather than individual system features or services.

Both types of the requirements sometimes interconnect with each other, for example, one functional requirement is connected to one of the non-functional constraints.

[28] Qoute from http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code (Retrieved on 12/2016)
[29] Quote from Mike Cohn: *Succeeding with Agile: Software Development Using Scrum*, page 314
[30] Quote from Ian Sommerville (2016): *Software Engineering*, page 102
[31] Quote from Ian Sommerville (2016): *Software Engineering*, page 105

In this project, all of the functional requirements for the system are dealt with writing user stories (described in chapter 4.1.1.) while non-functional requirements are addressed by constraints shown in the table below.

Non-functional requirement	Constraint
Interoperability	Language – The system should be written in the C#, JavaScript, CSS and HTML Configuration – All configuration should be stored in the XML files Data – Account data should be stored on the SQL Server. Data regarding users, events, businesses, location, spots and groups should be stored in the MongoDB
Maintainability	Code changes – system should allow easy and fast code changes and code adaptation Flexible schema – system should be able to handle frequent database schema changes
Performance	Database response time – database response time should be as quick as possible Available spots – Processing and showing the available spots should be as optimal as possible
Scalability	The system should remain effective with significant increase of number of resources and number of users
Security	Confidentiality – confidential information is accessible only by the authorized users Passwords – passwords are stored securely Requests – server distinguishes whether the request is authorized or not Confirmation – account has to be confirmed in order to be activated Reset password – user can reset his password by clicking on a link sent to the user's email Two factor authentication – if user wishes to delete his profile, he has to enter a code sent to his email address
Usability	Responsive web design – web application responds to different devices Navigation – user should be able to navigate to any part of the website within three clicks Simple design – design of the website should be as simple as possible Dynamicity – website should be dynamic in order to improve user experience

For the project the most important non-functional requirements are performance (holding the biggest business value), security (being the most important from the user point of view to insure that all data is protected), and usability ensuring that any new potential user can easily use the system.

3 System overview

This second part of the report directly connects to the previous, the research part, and to the next one, the development. The system overview is the part focused on the technological overview of the system, which covers the system architecture, the documentation used for the development, the back-end and front-end technologies used in the project, and concludes with the communication and testing subchapters.

3.1 Architecture

Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system[32]. The decision about the architecture is one of the most important parts of developing the software as it affects the price of the system, the date when the system will be finished, the system quality and the system maintainability.

Although the definition of the agile development is to answer on the changes and to be developed in increments, this cannot be applied to the architecture as changing the architecture is not practical, and it is usually really expensive and time consuming. Architectural design is a process where the system should be designed to satisfy both functional and non-functional requirements of the system.

Fundamental decisions for the architectural design should answer next questions[33]:

- Is there a generic application architecture that can act as a template for the system that is being designed?
- How will the system be distributed across hardware cores or processors?
- What architectural patterns or styles might be used?
- What will be the approach used to structure the system?
- What strategy will be used to control the operation of the components in the system?
- How will the structural components in the system be decomposed into sub-components?
- What architectural organization is the best for delivering the non-functional requirements of the system?
- How should the architecture of the system be documented?

Architectural pattern (or style) is a set of principles that improve partitioning and promote design reuse by providing solutions to frequently recurring problems[34]. The most important benefit of architectural patterns is that they provide a common language. They facilitate a higher level of conversation that is inclusive of patterns and principles, without getting into specifics. For example, by using architecture styles, there can be a discussion about the advantages and disadvantages of client/server compared to n-tier architectural style. Next subchapters focus on the system architecture which include MVC, MVVM, ASP.NET MVC framework and quality criteria.

[32] From Ian Sommerville (2016): *Software Engineering*, page 168
[33] List of questions from Ian Sommerville (2016): *Software Engineering*, page 171
[34] From https://msdn.microsoft.com/en-us/library/ee658117.aspx (Retrieved in 12/2016)

3.1.1 MVC-Based Web Application

For this project application, the Model-View-Controller (MVC) architectural pattern was used. This pattern separates the application into three main components: the model, the view, and the controller[35].

3.1.1.1 The model

Model objects are the parts of the application that implement the logic for the application's data domain and often they retrieve and store model state in a database[36]. In the project application, the model does not communicate with the database directly. Instead, it depends upon Data access layer, which then retrieves and stores data in the database.

For example, the User object might retrieve information from a database (calling Mongo context interface in the DAL), operate on it, and then write the updated information back to the User document in Mongo database.

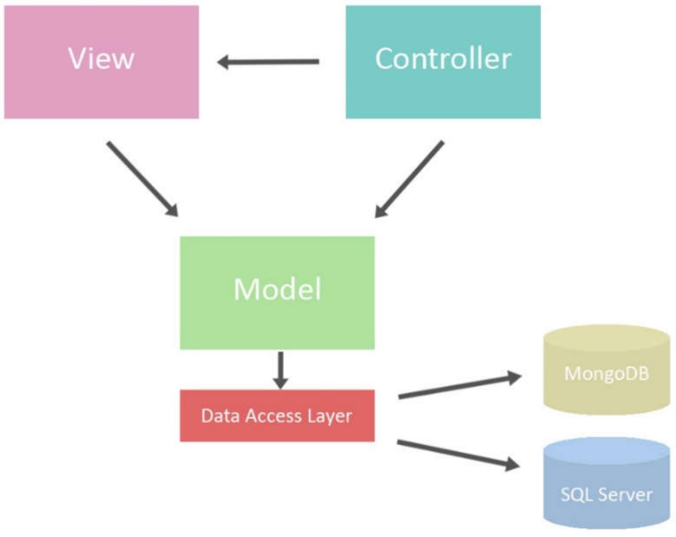
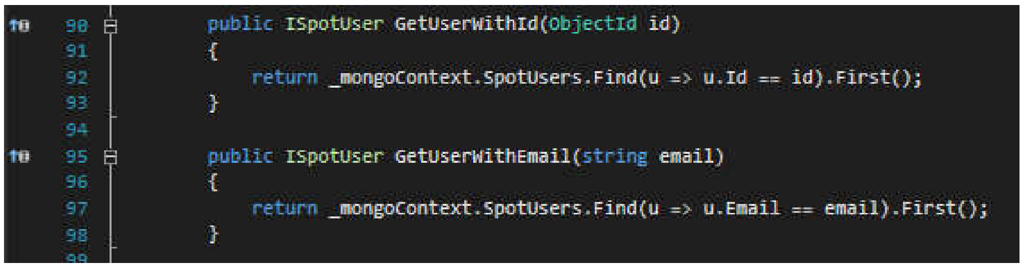
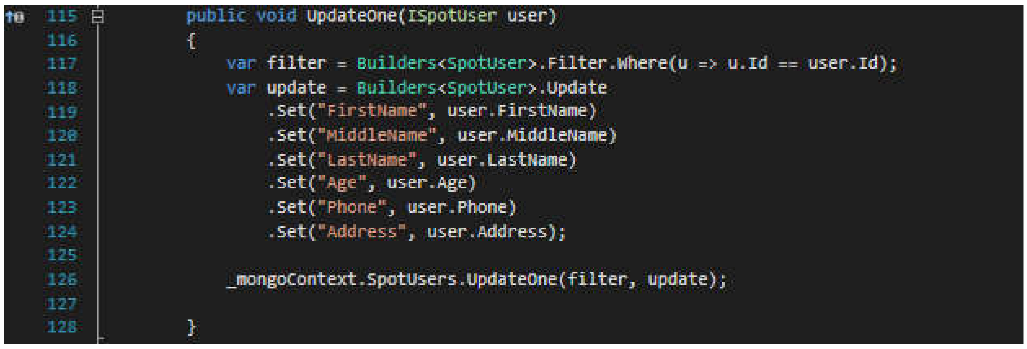


Figure 7 The overview of the system architecture



```
public ISpotUser GetUserWithId(ObjectId id)
{
    return _mongoContext.SpotUsers.Find(u => u.Id == id).First();
}

public ISpotUser GetUserWithEmail(string email)
{
    return _mongoContext.SpotUsers.Find(u => u.Email == email).First();
}
```

Figure 8 Get user from MongoDB with either email or ID



```
public void UpdateOne(ISpotUser user)
{
    var filter = Builders<SpotUser>.Filter.Where(u => u.Id == user.Id);
    var update = Builders<SpotUser>.Update
        .Set("FirstName", user.FirstName)
        .Set("MiddleName", user.MiddleName)
        .Set("LastName", user.LastName)
        .Set("Age", user.Age)
        .Set("Phone", user.Phone)
        .Set("Address", user.Address);

    _mongoContext.SpotUsers.UpdateOne(filter, update);

}
```

Figure 9 Update user information in MongoDB

[35] From https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx (Retrieved in 12/2016)
[36] From https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx (Retrieved in 12/2016)

3.1.1.2 The view

Views are the components that display the application's user interface (UI). Typically, this UI is created from the model data[37]. An example would be a view of the User profile details, which displays text boxes, labels and checkboxes based on the current state of the User object.

3.1.1.3 The controller

Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction[38].

3.1.1.4 Advantages

The advantages of the MVC-Based web application and ASP.NET framework are[39]:

- Separation of concerns (SoC) - the organization of code within MVC is very clean, organized and granular, making it easier for a web application to scale in terms of functionality. SoC promotes great design from the development standpoint
- Easier integration with client side tools - web applications are increasingly becoming as rich as the applications you see on your desktops. With MVC, it gives you the ability to integrate with such toolkits (such as jQuery) with greater ease and more seamless than in Web Forms
- Search Engine Optimization (SEO) Friendly / Stateless - URL's are more friendly to search engines (i.e. https://localhost:44371/Event/Details/1 - retrieve event with an ID of 1 vs https://localhost:44371/Event/Details/getevent.aspx, where ID is passed in the session)
- High degree of the control - it works well for the Web applications that are supported by large teams of developers and for Web designers who need a high degree of control over the application behaviour
- Test driven development - all core contracts in the MVC framework are interface-based and can be tested by using mock objects, which are simulated objects that imitate the behaviour of actual objects in the application.

3.1.2 Model-View-ViewModel

MVVM (Model-View-ViewModel) is an architectural pattern based on the MVC, which attempts to more clearly separate the development of user-interfaces from that of the business logic and behaviour in an application[40].

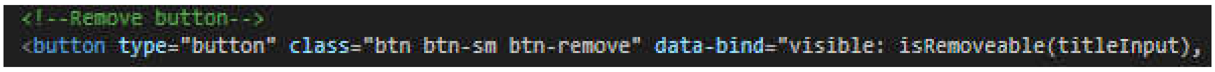
3.1.2.1 Model

Model is either domain data or business object which holds real time data. The model does not carry behaviours. Behaviour is mostly implemented in the business logic[41].

[37] Quote from https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx (retrieved in 12/2016)
[38] Quote from https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx (retrieved in 12/2016)
[39] Quote from https://www.quora.com/What-are-the-advantages-of-ASP-Net-MVC-over-ASP-Net-Web-Forms (retrieved in 12/2016)
[40] Quote from https://addyosmani.com/blog/understanding-mvvm-a-guide-for-javascript-developers/ (retrieved in 12/2016
[41] Quotes from https://www.tutorialspoint.com/knockoutjs/knockoutjs_mvvm_framework.htm (retrieved in 12/2016)

3.1.2.2 View

View is a Graphical User Interface created using markup language to represent data. View binds to properties of a ViewModel through data-bind concept which indirectly connect to model data. View need not be changed for any alteration done in the ViewModel. Changes made to data in ViewModel is automatically propagated in View due to binding[42].

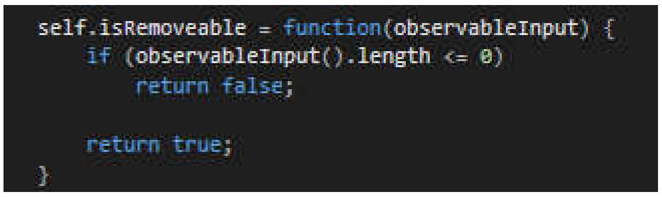


```
<!--Remove button-->
<button type="button" class="btn btn-sm btn-remove" data-bind="visible: isRemoveable(titleInput),
```

Figure 10 The example of how view binds to properties in ViewModel

3.1.2.3 ViewModel

ViewModel is the centre place, where data from Model and view's display logic are bundled together. ViewModel holds dynamic state of data. There is an implicit binder in between View and ViewModel to communicate with each other. This binding is inclusive of declarative data and command binding. Synchronization of View and ViewModel is achieved through this binding. Any change made in the View is reflected in the ViewModel and any change in the ViewModel is reflected in the View automatically. The existence of this 2 way binding mechanism is a key aspect of this MVVM pattern[43].



```
self.isRemoveable = function(observableInput) {
    if (observableInput().length <= 0)
        return false;

    return true;
}
```

Figure 11 ViewModels function isRemovable

An example of the ViewModel is CreateEventViewModel, which represents a connection between Event model and CreatEvent view. Its function isRemovable, determines if there is any text in the input field that can be removed. This particular function is used for visibility of the remove button.

3.1.3 ASP.NET MVC Framework

The ASP.NET MVC framework provides the following features[44]:

- Separation of application tasks (input logic, business logic, and UI logic), testability, and test-driven development (TDD). All core contracts in the MVC framework are interface-based and can be tested by using mock objects, which are simulated objects that imitate the behaviour of actual objects in the application. You can unit-test the application without having to run the controllers in an ASP.NET process, which makes unit testing fast and flexible. You can use any unit-testing framework that is compatible with the .NET Framework.
- An extensible and pluggable framework. The components of the ASP.NET MVC framework are designed so that they can be easily replaced or customized. You can plug in your own view engine, URL routing policy, action-method parameter serialization, and other components. The ASP.NET MVC framework also supports the use of Dependency Injection (DI) and Inversion

[42] Quotes from https://www.tutorialspoint.com/knockoutjs/knockoutjs_mvvm_framework.htm (retrieved in 12/2016)
[43] Quotes from https://www.tutorialspoint.com/knockoutjs/knockoutjs_mvvm_framework.htm (retrieved in 12/2016)
[44] Feature list from https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx (retrieved in 11/2016)

of Control (IoC) container models. DI enables you to inject objects into a class, instead of relying on the class to create the object itself. IoC specifies that if an object requires another object, the first objects should get the second object from an outside source such as a configuration file. This makes testing easier.

- Extensive support for ASP.NET routing, which is a powerful URL-mapping component that lets you build applications that have comprehensible and searchable URL's. URL's do not have to include filename extensions, and are designed to support URL naming patterns that work well for search engine optimization and representational state transfer (REST) addressing.
- Support for using the markup in existing ASP.NET page (.aspx files), user control (.ascx files), and master page (.master files) markup files as view templates. You can use existing ASP.NET features with the ASP.NET MVC framework, such as nested master pages, in-line expressions (<%= %>), declarative server controls, templates, data-binding, localization, and so on.
- Support for existing ASP.NET features. ASP.NET MVC lets you use features such as forms authentication and Windows authentication, URL authorization, membership and roles, output and data caching, session and profile state management, health monitoring, the configuration system, and the provider architecture.

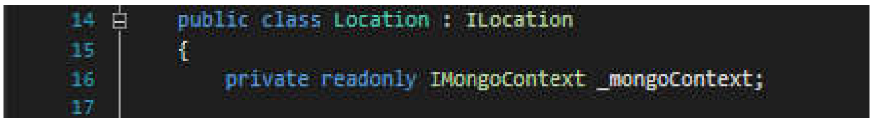
3.1.4 Quality criteria (Code & Design Pattern)

A design pattern is a general, reusable solution to a commonly occurring software problem[45]. In this project, a number of different design patterns were used in order to overcome particular problem.

3.1.4.1 Dependency Injection

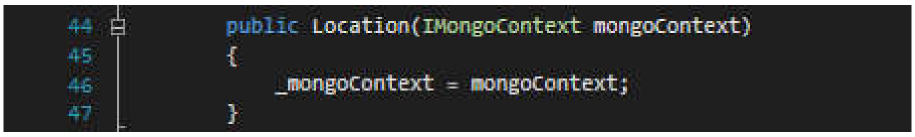
Dependency is just another object that class needs to function[46]. For example, the model class Location in this project, fetches data from a database using DAL object, therefore the model class has a dependency on the DAL object.

Dependency injection means, that rather than instantiating an object from inside of the class using the "new" operator, the dependency object abstraction (interface) is pushed into the class from the outside, via a constructor or a setter.



```
public class Location : ILocation
{
    private readonly IMongoContext _mongoContext;
```

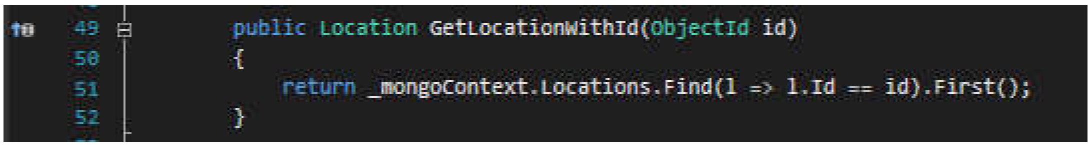
Figure 12 Private field of dependencies interface



```
    public Location(IMongoContext mongoContext)
    {
        _mongoContext = mongoContext;
    }
```

Figure 13 Dependency is pushed to the class from outside via the constructor

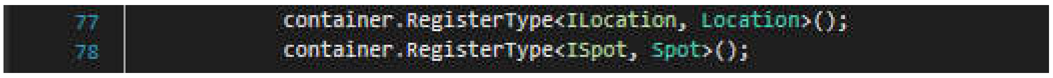
[45] Quote from https://msdn.microsoft.com/en-us/library/ff649977.aspx (retrieved on 12/2016)
[46] Quote from https://www.youtube.com/watch?v=IKD2-MAkXyQ (retrieved on 12/2016)



```
public Location GetLocationWithId(ObjectId id)
{
    return _mongoContext.Locations.Find(l => l.Id == id).First();
}
```

Figure 14 Using the dependency to retrieve data from a database

Additionally, in order to construct each class, Inversion of Control (IoC) container is used. In this project Unity IoC container was used, which figures out both - which dependencies classes need and how to instantiate them. The IoC container maps the dependencies that the class needs and creates them if they haven't been created yet.



```
container.RegisterType<ILocation, Location>();
container.RegisterType<ISpot, Spot>();
```

Figure 15 Instructing IoC container how to instantiate dependencies

There are several benefits to use dependency injection[47]:

- Reduced Dependencies - Dependency injection makes it possible to eliminate, or at least reduce, a component's unnecessary dependencies. A component is vulnerable to changes in its dependencies. If a dependency changes, the component might have to adapt to these changes.
- More Reusable Code - Reducing a component's dependencies typically makes it easier to reuse in a different context. The fact that dependencies can be injected and therefore configured externally, increases the reusability of that component.
- More Testable Code - Dependency injection also increases a component's testability. When dependencies can be injected into a component it is possible to inject mock implementations of these dependencies. Mock objects are used for testing as a replacement for a real implementation. The behaviour of the mock object can be configured.
- More Readable Code - Dependency injection moves the dependencies to the interface of components. This makes it easier to see what dependencies a component has, making the code more readable.

3.1.4.2 Command design pattern

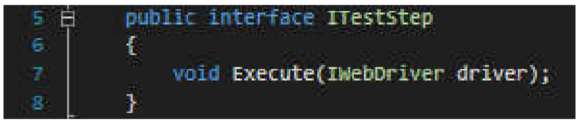
Command design pattern allows the application to add a work (code) to a queue and process it at a later time. Besides, it also adds the ability to retry the command later or undo it.

In this project, the command pattern was used as a base for the automated test framework. In the automated test itself, the test steps are added to the queue on the test case object. Later, they are executed calling the execute method. Besides already mentioned benefits, this usage of command pattern also provides a level of abstraction of the code. The tester is not required to use the selenium web driver directly, and the code is more readable, manageable and consistent. The pattern consists of 4 parts: Command, Receivers, Invoker and Client.

Command is a class that executes the code. All commands inherit from the same interface. In this project application Test step is the implementation of the command. Test step can either inherit an IActionTestStep interface or IVerificationTestStep interface. Both of these interfaces have a

[47] List of benefits from http://tutorials.jenkov.com/dependency-injection/dependency-injection-benefits.html (retrieved in 12/2016)

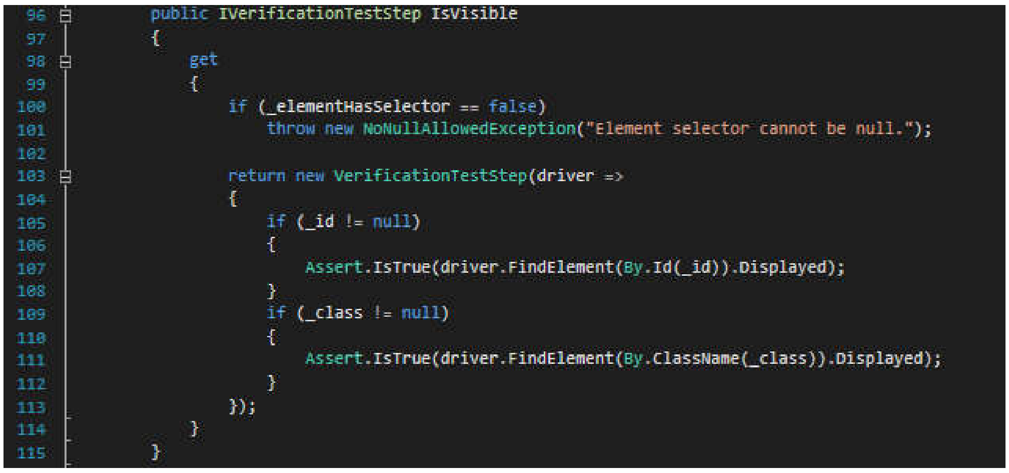
common parent ITestStep. This inheritance ensures that all commands will be able to be added to the same list and that all commands will have the necessary Execute method implementation.



```
public interface ITestStep
{
    void Execute(IWebDriver driver);
}
```

Figure 16 ITestStep interface that all commands inherit from

An example of the command is IsVisible verification command. This checks if a particular element is visible on the web page.

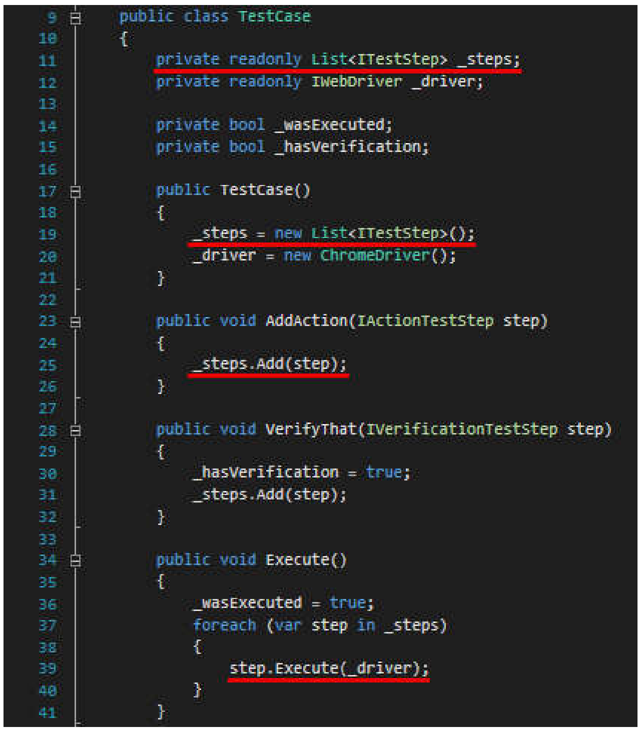


```
public IVerificationTestStep IsVisible
{
    get
    {
        if (_elementHasSelector == false)
            throw new NoNullAllowedException("Element selector cannot be null.");

        return new VerificationTestStep(driver =>
        {
            if (_id != null)
            {
                Assert.IsTrue(driver.FindElement(By.Id(_id)).Displayed);
            }
            if (_class != null)
            {
                Assert.IsTrue(driver.FindElement(By.ClassName(_class)).Displayed);
            }
        });
    }
}
```

Figure 17 Command that verifies if an element is visible

Receivers are business objects that receive actions from the command. In the project application, an example of the receiver is a selenium web driver.

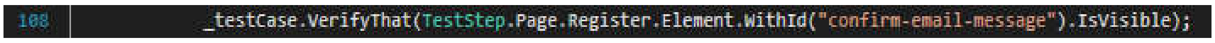
Invoker is a class where the action and verification test steps are added to the private list. The list is of type ITestStep which is a parent of both IActionTestStep and IVerificationTestStep. When an execute method is called upon the test case object, the list is iterated and every test step is executed individually. In addition the invoker also manages the web driver object.



```
public class TestCase
{
    private readonly List<ITestStep> _steps;
    private readonly IWebDriver _driver;

    private bool _wasExecuted;
    private bool _hasVerification;

    public TestCase()
    {
        _steps = new List<ITestStep>();
        _driver = new ChromeDriver();
    }

    public void AddAction(IActionTestStep step)
    {
        _steps.Add(step);
    }

    public void VerifyThat(IVerificationTestStep step)
    {
        _hasVerification = true;
        _steps.Add(step);
    }

    public void Execute()
    {
        _wasExecuted = true;
        foreach (var step in _steps)
        {
            step.Execute(_driver);
        }
    }
```

Figure 18 Command pattern invoker - Test case class

Client is the actual program that is going to run. In the project application the client represents the automated tests that request the commands.

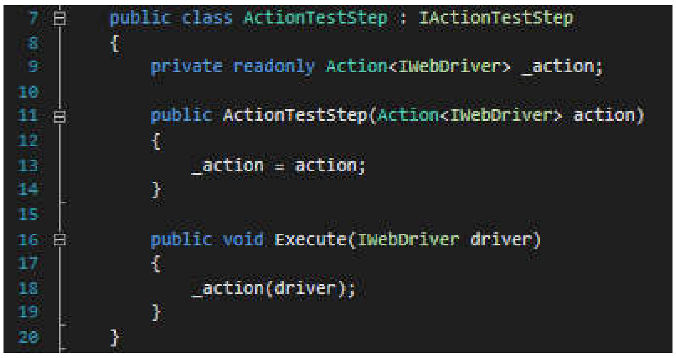


```
            _testCase.VerifyThat(TestStep.Page.Register.Element.WithId("confirm-email-message").IsVisible);
```

Figure 19 Example which verifies that an element with the id "confirm-email-message" is visible on the registration page.

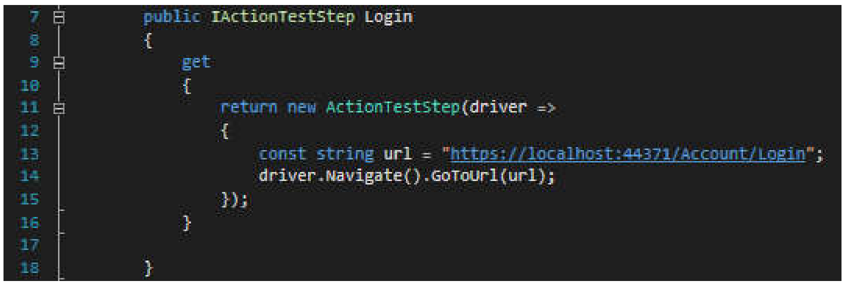
The downside of the command pattern is that it requires a lot of small classes that store commands. In order to overcome this problem, the following has been done:

- When the command is requested, a new instance of either action or verification test step is created.
- Both classes require an action in their constructor.
- The action is later stored in a private field and executed in the execution method.



```
public class ActionTestStep : IActionTestStep
{
    private readonly Action<IWebDriver> _action;

    public ActionTestStep(Action<IWebDriver> action)
    {
        _action = action;
    }

    public void Execute(IWebDriver driver)
    {
        _action(driver);
    }
}
```

Figure 20 Code sample of the ActionTestStep class



```
public IActionTestStep Login
{
    get
    {
        return new ActionTestStep(driver =>
        {
            const string url = "https://localhost:44371/Account/Login";
            driver.Navigate().GoToUrl(url);
        });
    }

}
```

Figure 21 Passing an action into a constructor of ActionTestStep

3.2 Documentation

Doing agile methodology in the software development implies that there is not a lot of the documentation during the process. Nevertheless, the documentation is important because it helps the developers to know what they should implement.

Therefore, in this project two main types of the documentation are covered. Firstly, there is a documentation connected to the agile methodologies (e.g. product backlog, sprint backlog, etc.) which are described in the chapter four. Secondly, there are specific documentation artefacts that are used: domain model, use case model, use case testing and test specifications.

Documentation is also a part of product standards. For that purpose use case testing and test specifications used predetermined templates. This is important to keep the same procedures so that the tests cover all of the important fields.

3.2.1 Domain model

In software engineering, a domain model is a conceptual model of the domain that incorporates both behaviour and data[48]. The domain model is used to represent the data of the business and set of rules how does this data relate. Although domain model is a reference point that is used during the project, in agile development it could go through some changes.

For the project developers, an initial version of the domain model was created (figure below) where the data that will be used in the system was set up. Additionally, it describes how does data interact with each other. It is important to say that this was done early in the process to know all of the parts would interact when they are integrated.

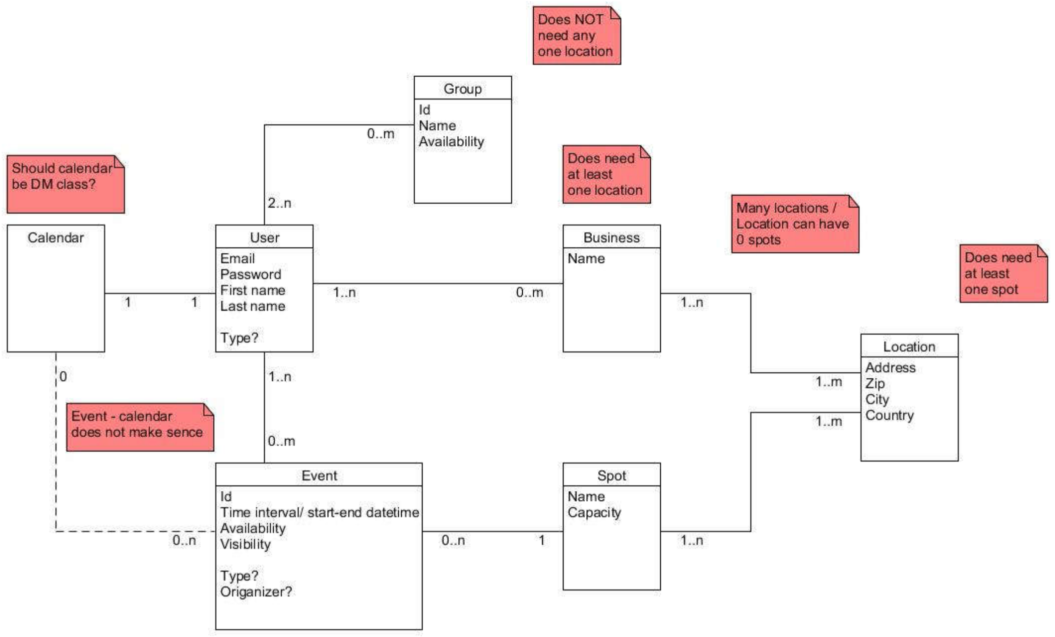

Does NOT need any one location
Group
Id
Name
Availability
0..m
2..n
Should calendar be DM class?
Does need at least one location
Many locations / Location can have 0 spots
Does need at least one spot
Calendar
User
Email
Password
First name
Last name
Type?
Business
Name
1
1
1..n
0..m
1..n
Location
Address
Zip
City
Country
1..m
1..m
0
1..n
Event - calendar does not make sence
0..m
Event
Id
Time interval/ start-end datetime
Availability
Visibility
Type?
Origanizer?
0..n
0..n
1
Spot
Name
Capacity
1..n

Figure 22 Initial drawing of the domain model

As it can be seen from the figure, some notes that could be important during the creation of the software had been written down, as they are important to specify all of the potential issues that could happen.

3.2.2 Use case model

Use cases are important for this system as they give a list of actions that are occurring as interactions between actors and the system. Although the actors can be either people or another system, in the original estimate four different actors were defined, all of them human:

- Anonymous (User that has not been registered in the system)
- Person (Registered user)
- Employee (User that is a part of at least one business)
- Business owner (User that owns at least one business)

After defining the actors and the actions which they could be doing in the system, use case model can be created.

[48] Quote from Martin Fowler: *Patterns of Enterprise Application Architecture*, page 116

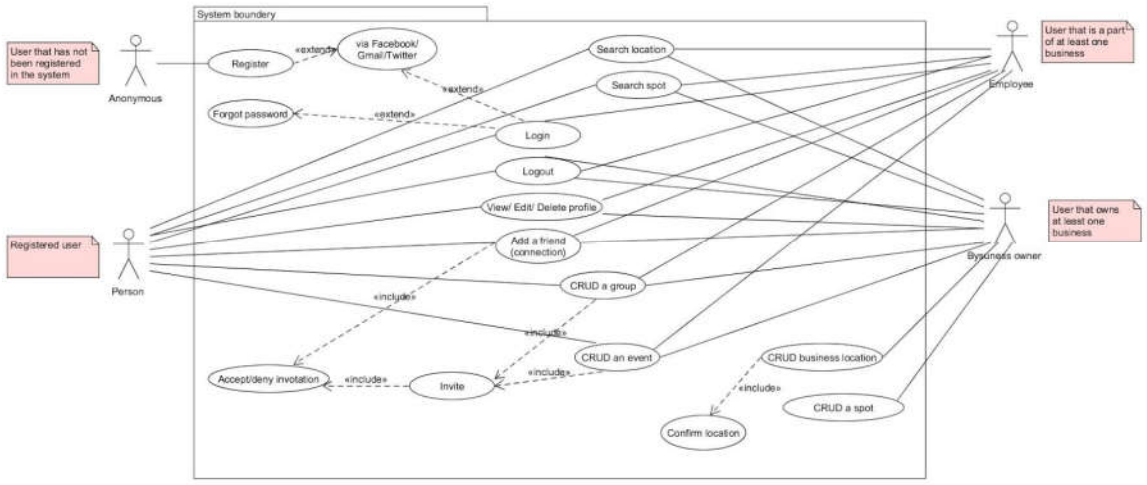


Figure 23 Use case model for the project

The use case model helps to set up the level of security for each actor, and provides an overview of all of the requirements that need to be implemented. Use cases are also a great tool for doing the testing, which is described in the next chapter.

3.2.3 Use case testing

Use case testing is used as the requirement testing in this project. Use case testing describes the system functionality and the processes that are checking whether the system is doing what is supposed. Use case testing is a technique that helps to identify test cases that exercise the whole system on a transaction by transaction basis from start to finish[49].

Use case testing is a document that describes what the actor does and sees. In the document it is stated: the name, the description of the interactions, the trigger for the test, the actors, the preconditions that have to be satisfied, the use case goals, failed conclusion, extensions and the execution steps. The document is usually written by using the language and terms of the business rather than technical terms.

The biggest advantages of using the use case testing are:

- They serve as the foundation for developing the acceptance testing levels
- They could uncover integration defects, or to be more precise, the defects caused by a bad interaction in the system procedures
- They describe the system process flows based on the most likely scenario which makes it a perfect tool for finding bugs in the real-world use of the system
- Each test has a mainstream scenario, but it can also have additional alternative branches that can cover some special cases or exceptional conditions.

Use case testing was done for the whole project and examples can be seen in the appendix, pages 79 and 81.

[49] From http://istqbexamcertification.com/what-is-use-case-testing-in-software-testing/ (Retrieved in 12/2016)

3.2.4 Test specifications

Test specifications consist of a detailed summary of what scenarios will be tested, how they will be tested, how often, and so on, for a given feature[50]. In this project the test specification will be done for the manual testing. Manual test specification document for this project consist of:

- ID
- Test name
- Test description
- Browser used
- URL used
- Data that will be used
- Test steps

Manual tests are used in this project (example in the appendix, page 78) with two main purposes: to test for any potential bugs and to propose visual changes for the platform that was used.

3.3 Back-end

The back-end is the part of the system that is not directly accessed by the user, and which covers how the system operates. In this project the main focus of the back-end will be dedicated to one of the biggest architectural decisions that had to be made – the project databases.

Next few subchapters focus on the decision process, starting from the differences between database types, consistency types, summing up with the decision and then continuing on with some of the characteristics of the chosen database.

3.3.1 SQL vs NoSQL

The first choice was to decide between SQL and NoSQL type of databases. To be able to make the decision the focus was on five areas that needed to be discussed: nature of data, database properties, scalability, impedance mismatch, and CAP theorem.

3.3.1.1 Nature of data

While relational databases deal strictly with structured data, NoSQL databases are typical for semi-structured data. Structured data are represented in a strict format. Each record of the table follows the same format as every other record in the table[51]. While certain structure is present, this type of data lacks very rigid and strict data model structure. In addition, not all data must follow the same structure. Semi-structured data also does not have a predefined schema - additional attributes may be introduced in some of the newer data items[52].

[50] From https://blogs.msdn.microsoft.com/saraford/2004/10/28/developing-a-test-specification/ (Retrieved in 12/2016)
[51] From Ramez Elmasri: *Fundamentals of Database Systems*, Chapter 12.1
[52] From Ramez Elmasri: *Fundamentals of Database Systems*, Chapter 12.1

3.3.1.2 Database properties

The crucial difference between these two types of databases is the consistency. While RDBMS creates a very safe environment when handling data, using ACID properties, it often comes with a cost. Achieving such write consistency requires sophisticated locking, which is typically a heavyweight pattern for most use cases. It can be also unnecessarily pessimistic. In the NoSQL world, ACID transactions are less fashionable as some databases have loosened the requirements for immediate consistency, data freshness and accuracy in order to gain other benefits, like scale and resilience[53].

3.3.1.3 Scalability

Relational databases are designed to scale up, while NoSQL databases are designed to scale out[54]. Scaling up, usually refers to adding more power (buying more expensive, robust server, better processor, RAM, etc.). It comes with a number of advantages - consuming less power, less cooling costs, less licensing costs and in general it is easier to implement. On the other hand, horizontal scaling costs much more and there is a bigger risk of failure. There is also a limited upgradability in the future. Scaling out, generally refers to adding more servers with less processors and RAM. This solution is much cheaper and the expenses are much more predictable. It is also easier to run fault-tolerance. When compared with vertical scaling, utility costs and licensing fees are higher[55].

3.3.1.4 Impedance mismatch

The object-relational impedance mismatch is a set of conceptual and technical difficulties that are often encountered when relational database management system is being used by a program written in an object-oriented programming language or style, particularly when the object or class definitions are mapped in a straightforward way to database tables or relational schema[56].

3.3.1.5 CAP theorem

CAP theorem states that, in a distributed system (a collection of interconnected nodes that share data.), there can only be two out of the following three guarantees across a write/read pair: Consistency, Availability, and Partition Tolerance - one of them must be sacrificed[57].

The CAP theorem options are as follows[58]:

- Consistency - A read is guaranteed to return the most recent write for a given client.
- Availability - A non-failing node will return a reasonable response within a reasonable amount of time (no error or timeout).
- Partition Tolerance - The system will continue to function when network partitions occur.

[53] From http://neo4j.com/blog/why-nosql-databases/ (retrieved in 12/2016)
[54] Adapted from https://www.youtube.com/watch?v=XPqrY7YEs0A (retrieved in 12/2016)
[55] Adapted from http://www.vtagion.com/scalability-scale-up-scale-out-care (retrieved in 12/2016)
[56] From https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch (retrieved in 12/2016)
[57] From http://robertgreiner.com/2014/08/cap-theorem-revisited/ (retrieved in 12/2016)
[58] List acquired from http://robertgreiner.com/2014/08/cap-theorem-revisited/ (retrieved in 12/2016)

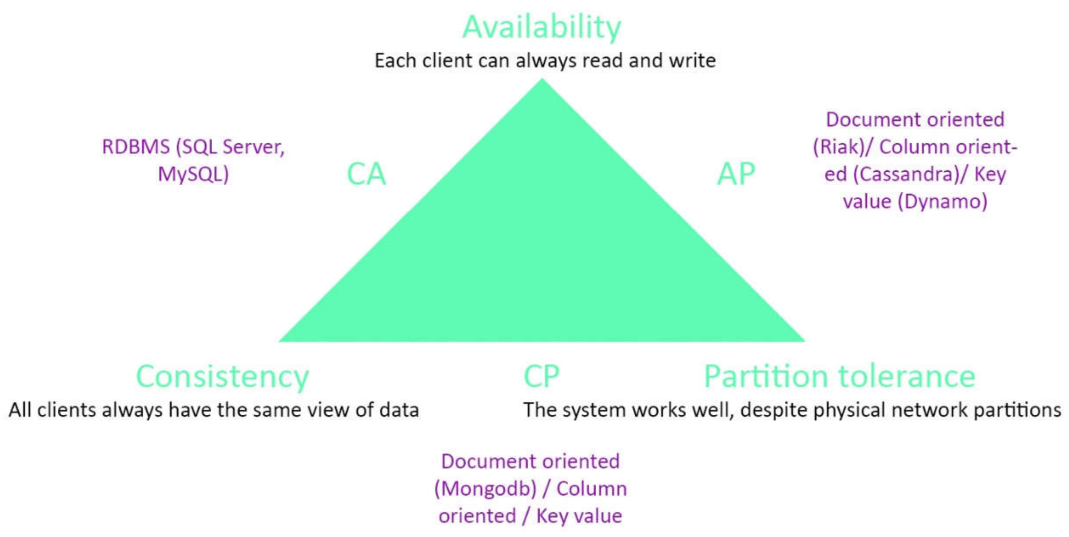

Availability
Each client can always read and write
RDBMS (SQL Server, MySQL)
CA
AP
Document oriented (Riak)/ Column oriented (Cassandra)/ Key value (Dynamo)
Consistency
All clients always have the same view of data
CP
Partition tolerance
The system works well, despite physical network partitions
Document oriented (Mongodb) / Column oriented / Key value

Figure 24 CAP theorem

3.3.2 ACID vs BASE

The second choice was to decide between ACID or BASE consistency. To be able to clarify which option is better, the properties of both were listed down.

ACID stands for Atomicity, Consistency, Isolation, and Durability. These are the properties of a transaction. All of the ACID properties have next attributes[59]:

- ATOMICITY: The atomicity property identifies that the transaction is atomic. An Atomic transaction is either fully completed, or is not begun at all. If for any reason an error occurs and the transaction is unable to complete all of its steps, then the system is returned to the state it was in before the transaction was started.
- CONSISTENCY: A transaction enforces Consistency in the system state by ensuring that at the end of any transaction the system is in a valid state. If the transaction completes successfully, then all changes to the system have been properly made, and the system will be in a valid state. If any error occurs in a transaction, then any changes already made will be automatically rolled back.
- ISOLATION: When a transaction runs in Isolation, it appears to be the only action that the system carried out at that time. If there are two transactions that are both performing the same function and are running at the same time, transaction isolation will ensure that each transaction behaves as it has exclusive use of the system.
- DURABILITY: A transaction is Durable once it has been successfully completed and all of the changes it made to the system are permanent. There are safeguards that will prevent the loss of any information, even in the case of the system failure.

[59] From http://www.c-sharpcorner.com/blogs/sql-server-acid-properties1 (retrieved in 12/2016)

The BASE acronym (**B**asically **A**vailable, **S**oft state, **E**ventual consistency) is used to describe the properties of certain databases, usually NoSQL databases. The BASE properties are as follows[60]:

- Basically available indicates that the system does guarantee availability, in terms of the CAP theorem
- Soft state indicates that the state of the system may change over time, even without input. This is because of the eventual consistency model.
- Eventual consistency indicates that the system will become consistent over time, given that the system doesn't receive input during that time.

A BASE datastore values availability (since that is important for scale), but it does not offer guaranteed consistency of replicated data at write time. Overall, the BASE consistency model provides a less strict assurance than ACID – data will be consistent in the future[61].

There is no correct answer to a question does a certain application needs an ACID or BASE consistency. Given BASE's loose consistency, it is important to be more knowledgeable and rigorous about consistent data if a BASE store is selected for an application. On the other hand, planning around BASE limitations can sometimes be a major disadvantage when compared to the ACID transactions. A fully ACID database is the perfect fit for use cases where data reliability and consistency are essential[62].

3.3.3 Selecting a database

After discussing two main choices mentioned above, a conclusion has been made that both SQL and NoSQL properties are beneficial for the project application, so it was decided to use both for different purposes. This approach is called polyglot persistence.

Polyglot persistence means, that when storing data, it is best to use multiple data storage technologies. Technologies are chosen based upon the way data is used by individual applications or components of a single application. It means picking the right tool for the right use case[63].

When it comes to Account and Identity Security, it was decided not to try to re-invent the wheel. ASP.NET Identity solves the problem of authentication security and user identity management. Account information about the user, logins, claims and roles are stored in the SQL Server database.

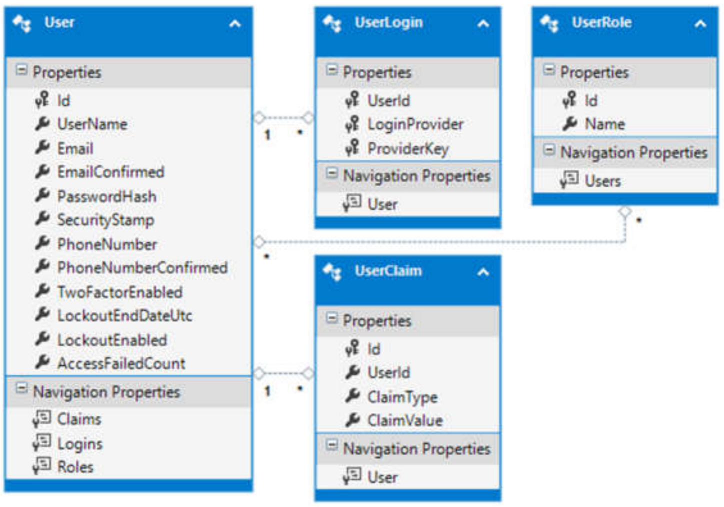
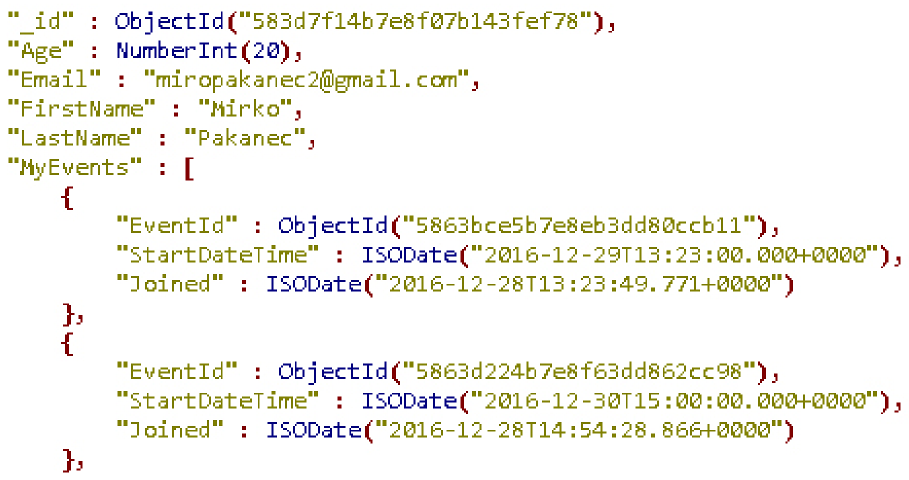


Figure 25 Account information schema (in the SQL server database)

[60] Definition and list acquired from http://stackoverflow.com/questions/3342497/explanation-of-base-terminology (retrieved in 12/2016)
[61] From http://neo4j.com/blog/acid-vs-base-consistency-models-explained/ (retrieved in 12/2016)
[62] From http://neo4j.com/blog/acid-vs-base-consistency-models-explained/ (retrieved in 12/2016)
[63] From http://www.jamesserra.com/archive/2015/07/what-is-polyglot-persistence/ (retrieved in 12/2016)

On the other hand, for the other user information or data about the businesses, locations, spots or events, it had been decided to use NoSQL document database MongoDB. There are several reasons why NoSQL provides a better fit for the application:

- Semi-structured data. One of the main reasons why NoSQL is much more beneficial than SQL's rigid structure. In addition, the project system is much heavier on reads compared to writes. It is much faster to, for example, have an array of user's events embedded in the document than to seek them in another table.



```
"_id" : ObjectId("583d7f14b7e8f07b143fef78"),
"Age" : NumberInt(20),
"Email" : "miropakanec2@gmail.com",
"FirstName" : "Mirko",
"LastName" : "Pakanec",
"MyEvents" : [
    {
        "EventId" : ObjectId("5863bce5b7e8eb3dd80ccb11"),
        "StartDateTime" : ISODate("2016-12-29T13:23:00.000+0000"),
        "Joined" : ISODate("2016-12-28T13:23:49.771+0000")
    },
    {
        "EventId" : ObjectId("5863d224b7e8f63dd862cc98"),
        "StartDateTime" : ISODate("2016-12-30T15:00:00.000+0000"),
        "Joined" : ISODate("2016-12-28T14:54:28.866+0000")
    },
```

Figure 26 Example of semistructured data

- Data volume. It is expected that the application would be used internationally and the amount of data could grow into a huge number of records. Due to this, a query execution time could cause huge performance issues in the relational data model.
- Data variety. Today's data change dramatically. Relational models can have problems with the sustained level of write loads and therefore crash during the activity peak. The data is far more varied than what relational databases were originally designed for. In fact, that is why many of today's RDBMS deployments have a number of nulls in their tables and null checks in their code – it is all to adjust to the contemporary data variety[64]. Another point in favour of NoSQL databases, is that all facts about the relational model are not known at the design time. Changes can always occur and flexible schema provided by MongoDB is an advantage[65].
- Environment - Avoiding impedance mismatch is by itself a huge benefit from the development point of view. In addition C# - MongoDB driver is very well documented and supports multiple feature, such as LINQ.

[64] Qoute taken from http://neo4j.com/blog/why-nosql-databases/ (retrieved in 12/2016)
[65] Adapted from http://www.zdnet.com/article/rdbms-vs-nosql-how-do-you-pick/ (retrieved in 12/2016)

3.3.4 Document databases

Document databases store and retrieve documents. One of the advantages is that documents in the same collection might be different. Documents store semi-structured data and use the flexible JSON schema. Since the schema is flexible, document databases have the possibility to store arrays (which would break the 1st normal form when using RDBMS) or polymorphic attributes (RDBMS would case nulls)[66].

While document databases might face redundancy in data, they are advantageous when it comes to performance. Another advantage is that data is in a format that the application expects and therefore developers avoid object relational independence mismatch. Document databases are better for reading heavy systems, because retrieving data is a question of single seek, rather than performing joins across multiple tables.

3.3.5 Data modeling in MongoDB

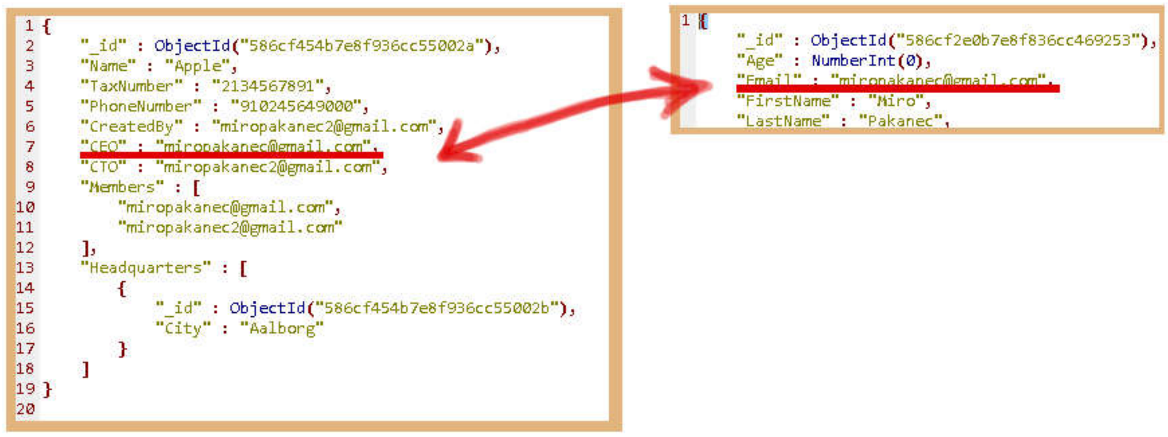
Data in MongoDB has a flexible schema. Unlike SQL databases, where the table's schema must be determined and declared before inserting data, MongoDB's collections do not enforce document structure. This flexibility facilitates the mapping of documents to an entity or an object[67].

The key decision in designing data models for MongoDB applications revolves around the structure of documents and the representation of relationships between data. There are two tools that allow applications to represent these relationships: references and embedded documents[68].

3.3.5.1 References

References store the relationships between data by including links or references from one document to another. Applications can resolve these references to access the related data[69].

An example of a reference in the project is the relationship between user and business. A CEO of a business has to be a user registered in the application.



```
{
    "_id" : ObjectId("586cf454b7e8f936cc55002a"),
    "Name" : "Apple",
    "TaxNumber" : "2134567891",
    "PhoneNumber" : "910245649000",
    "CreatedBy" : "miropakanec2@gmail.com",
    "CEO" : "miropakanec@gmail.com",
    "CTO" : "miropakanec2@gmail.com",
    "Members" : [
        "miropakanec@gmail.com",
        "miropakanec2@gmail.com"
    ],
    "Headquarters" : [
        {
            "_id" : ObjectId("586cf454b7e8f936cc55002b"),
            "City" : "Aalborg"
        }
    ]
}
```

```
{
    "_id" : ObjectId("586cf2e0b7e8f836cc469253"),
    "Age" : NumberInt(0),
    "Email" : "miropakanec@gmail.com",
    "FirstName" : "Miro",
    "LastName" : "Pakanec",
```

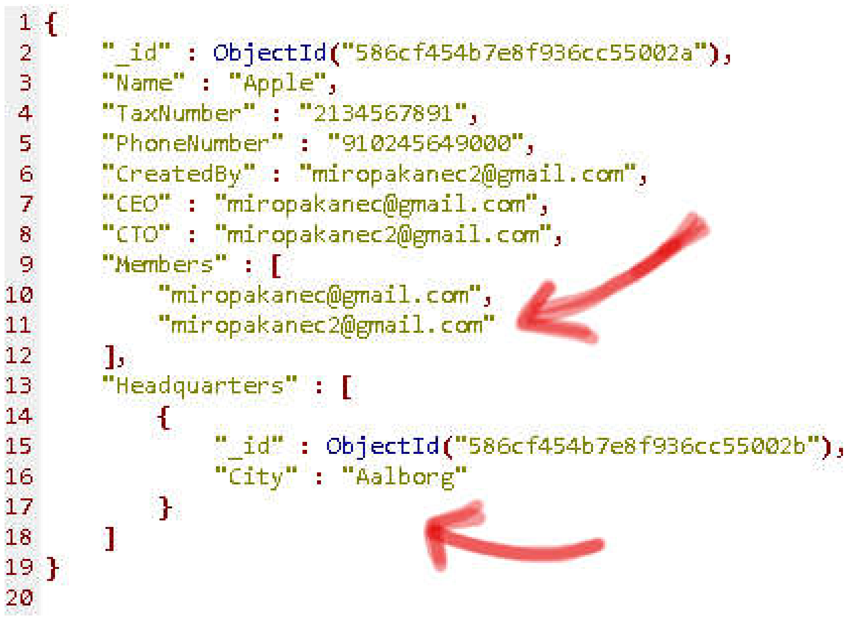
Figure 27 Reference example in the project

[66] Whole subchapter adapted from https://www.youtube.com/watch?v=X1yNTq_R2JA (retrieved in 12/2016)
[67] From https://docs.mongodb.com/manual/core/data-modeling-introduction/ (Retrieved in 12/2016)
[68] From https://docs.mongodb.com/manual/core/data-modeling-introduction/ (Retrieved in 12/2016)
[69] From https://docs.mongodb.com/manual/core/data-modeling-introduction/ (Retrieved in 12/2016)

3.3.5.2 Embedded data

Embedded documents capture relationships between data by storing related data in a single document structure. MongoDB documents make it possible to embed document structures in a field or array within a document. These denormalized data models allow applications to retrieve and manipulate related data in a single database operation[70].

An example of the embedded data in this project are Members of the business. In order to access all of the members quickly, their identifiers (in this case emails) are stored within an array.



```
{
    "_id" : ObjectId("586cf454b7e8f936cc55002a"),
    "Name" : "Apple",
    "TaxNumber" : "2134567891",
    "PhoneNumber" : "91024564900",
    "CreatedBy" : "miropakanec2@gmail.com",
    "CEO" : "miropakanec@gmail.com",
    "CTO" : "miropakanec2@gmail.com",
    "Members" : [
        "miropakanec@gmail.com",
        "miropakanec2@gmail.com"
    ],
    "Headquarters" : [
        {
            "_id" : ObjectId("586cf454b7e8f936cc55002b"),
            "City" : "Aalborg"
        }
    ]
}
```

Figure 28 Example for embedded data in the project

3.3.5.3 Atomicity considerations

In MongoDB, write operations are atomic at the document level, and no single write operation can atomically affect more than one document or more than one collection. A denormalized data model with embedded data combines all related data for a represented entity in a single document[71].

This facilitates atomic write operations since a single write operation can insert or update the data for an entity. Normalizing the data would split the data across multiple collections and would require multiple write operations that are not atomic collectively. However, schemas that facilitate atomic writes may limit ways that applications can use the data or may limit the ways to modify applications[72].

[70] Quotes from https://docs.mongodb.com/manual/core/data-modeling-introduction/ (Retrieved in 12/2016)
[71] Quotes from https://docs.mongodb.com/manual/core/data-modeling-introduction/ (Retrieved in 12/2016)
[72] Quotes from https://docs.mongodb.com/manual/core/data-modeling-introduction/ (Retrieved in 12/2016)

3.4 Front-end

The front-end is everything involved with what the user sees, including design and languages like HTML and CSS[73]. In next subchapters, topics from the front-end point are covered.

3.4.1 Views

In an ASP.NET MVC application, there is not a page on disk that corresponds to the path in the URL that you type into the address bar of your browser. The closest thing to a page in an ASP.NET MVC application is something called a view[74].

The view encapsulates the presentation details of the user's interaction with the application. Views are HTML templates with embedded code that generate content to send to the client[75]. Application maps incoming browser requests to controller actions. In the majority of cases the controller action returns a view, but it can also redirect to another action or returns something else, for example pure text.

An example of how controller and views operate is when a user tries to access the create event page. First, he enters the URL to the browser.

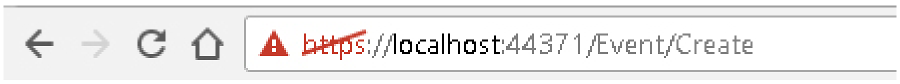
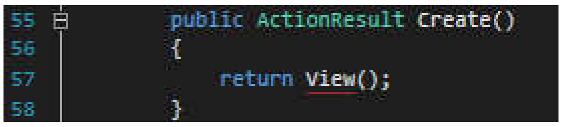


Figure 29 How controller and view operate: First step - entering the URL

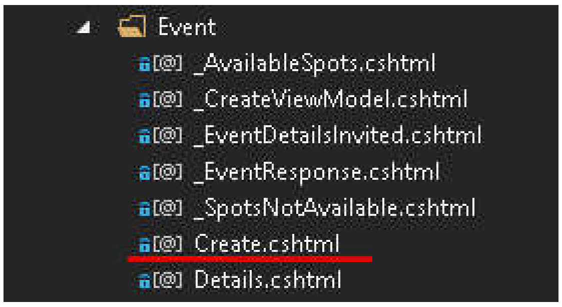
This request invokes a Create action method in the Event controller.



```
        public ActionResult Create()
        {
            return View();
        }
```

Figure 30 How controller and view operate: Second step - Create action method is invoked

This action method returns a particular View.



```
◢ Event
    _AvailableSpots.cshtml
    _CreateViewModel.cshtml
    _EventDetailsInvited.cshtml
    _EventResponse.cshtml
    _SpotsNotAvailable.cshtml
    Create.cshtml
    Details.cshtml
```

Figure 31 How controller and view operate: Third step - particular View is returned

Lastly, the View then generates a content and sends it back to the user.

[73] From http://blog.digitaltutors.com/whats-difference-front-end-back-end/ (Retrieved in 12/2016)
[74] Quotes from https://www.asp.net/mvc/overview/older-versions-1/views/asp-net-mvc-views-overview-cs (Retrieved in 12/2016)
[75] Quotes from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview (Retrieved in 12/2016)

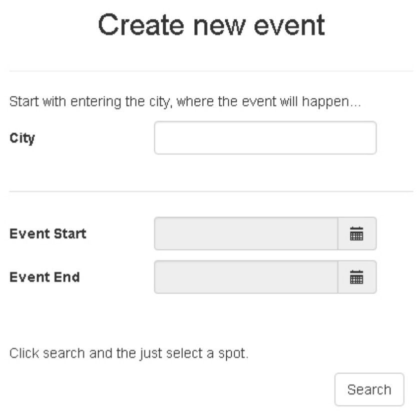
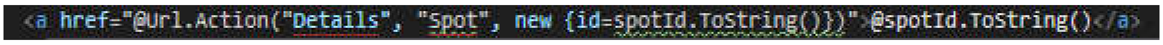

Create new event
Start with entering the city, where the event will happen...
City
Event Start
Event End
Click search and the just select a spot.
Search

Figure 32 How controller and view operate: Final step - content is generated

3.4.2 Razor syntax

Razor is a markup syntax for embedding server based code into web pages. The Razor syntax consists of Razor markup, C# and HTML. Razor supports C# and uses the "**@**" symbol to transition from HTML to C#. Razor evaluates C# expressions and renders them in the HTML output. Razor can transition from HTML into C# or into Razor specific markup. When the "**@**" symbol is followed by a Razor reserved keyword it transitions into Razor specific markup, otherwise it transitions into plain C#[76].



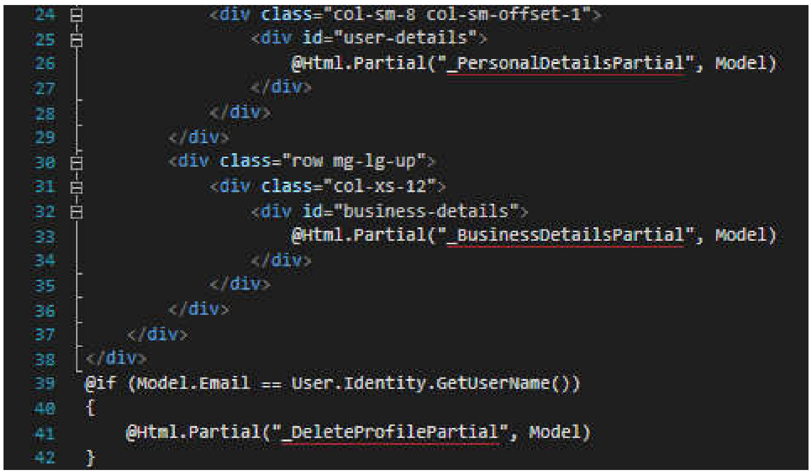
```
<a href="@Url.Action("Details", "Spot", new {id=spotId.ToString()})">@spotId.ToString()</a>
```

Figure 33 Razor syntax in the view

3.4.3 Partial views

A partial view is a view that is rendered within another view. The HTML output generated by executing the partial view is rendered into the calling (or parent) view[77].

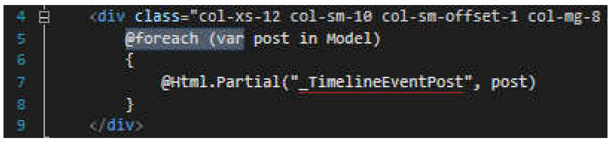
Usage of a partial view is beneficial, when a view itself can be split into multiple pieces, in order to increase maintainability. For example, the user details view can be divided into many logical pieces, like personal details, business details or a section to delete a profile.



```
                <div class="col-sm-8 col-sm-offset-1">
                    <div id="user-details">
                        @Html.Partial("_PersonalDetailsPartial", Model)
                    </div>
                </div>
            </div>
            <div class="row mg-lg-up">
                <div class="col-xs-12">
                    <div id="business-details">
                        @Html.Partial("_BusinessDetailsPartial", Model)
                    </div>
                </div>
            </div>
        </div>
    </div>
@if (Model.Email == User.Identity.GetUserName())
{
    @Html.Partial("_DeleteProfilePartial", Model)
}
```

Figure 34 Partial views within a view

[76] Paragraph from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor (Retrieved in 12/2016)
[77] Quotes from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial (Retrieved in 12/2016)



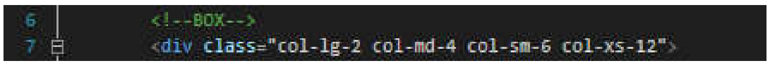
```
<div class="col-xs-12 col-sm-10 col-sm-offset-1 col-mg-8
    @foreach (var post in Model)
    {
        @Html.Partial("_TimelineEventPost", post)
    }
</div>
```

Figure 35 Reusability of the partial view

Another case when partial views can be used, is when some parts of the markup repeat. Partial views reduce duplication of content and allow reusability. For example, when a timeline posts are generated from the list of the ViewModels, the list is iterated with foreach loop and individual view models are passed to another partial view.

3.4.4 Responsive Web Design

Responsive Web design is the approach that suggests that design and development should respond to the user's behaviour and environment based on screen size, platform and orientation[78]. This approach is described the best by the Jeffrey Veen's quote: "Day by day, the number of devices, platforms, and browsers that need to work with your site grows. Responsive web design represents a fundamental shift in how we'll build websites for the decade to come"[79]. In other words, the website should have the technology to automatically respond to the user's preferences. This would eliminate the need for a different design and development phase for each new gadget on the market[80].



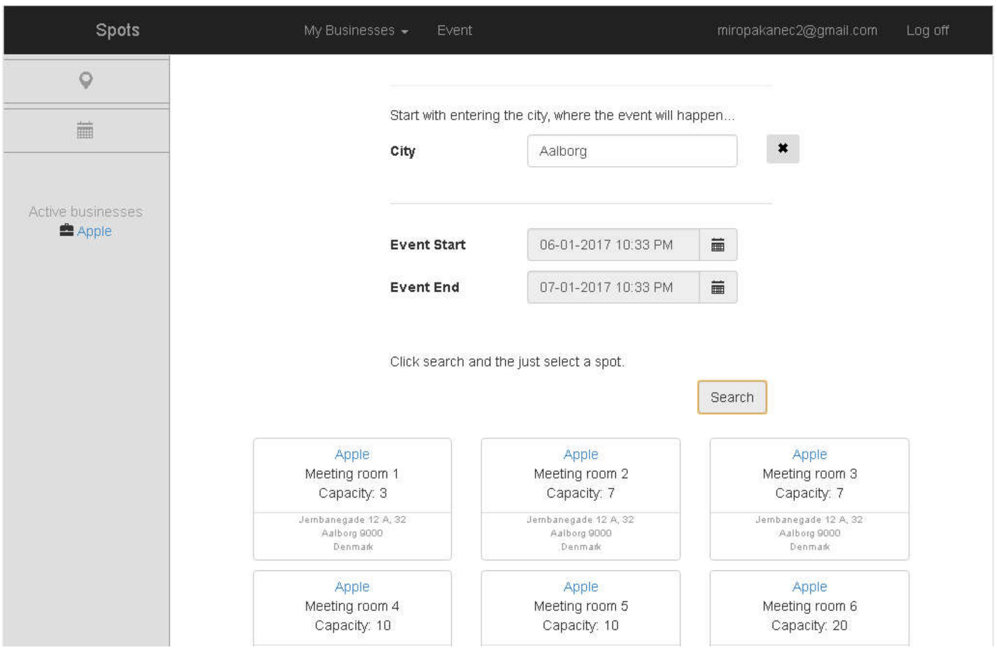
```
<!--BOX-->
<div class="col-lg-2 col-md-4 col-sm-6 col-xs-12">
```

Figure 36 Different column setups using bootstrap

Bootstrap library has been used in order to achieve the responsibility of the website. Bootstrap offers a 12 column grid system that adjusts according to viewport size. While the bootstrap's grid system has been used in many places in the application, a very clear example is displaying available spots, when an event is created. The width of the column that holds the box of an individual spot changes according to the viewport size.

[78] From https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/ (Retrieved in 12/2016)
[79] From http://www.azquotes.com/author/38434-Jeffrey_Veen (Retrieved in 12/2016)
[80] From https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/ (Retrieved in 12/2016)

For example, if a user views the web page on a large desktop device (with width greater or equal to 1200px, example – figure 62 in appendix), the spot box has a width of 2 columns, which is 1/6 of the overall screen width. On medium size desktop devices (992px - 1200px, example – figure 37) the spot box width increases to 4 columns, which is 1/3 of the screen width. In the same manner, the width increases for tablets (768px - 992px, example – figure 63 in appendix) and small mobile devices (with width smaller than 768px example on figure 38) to 6 columns (1/2 of the screen width) and 12 columns (the width of the whole screen).



Start with entering the city, where the event will happen...

City: Aalborg

Event Start: 06-01-2017 10:33 PM

Event End: 07-01-2017 10:33 PM

Click search and the just select a spot.

Search

Apple
Meeting room 1
Capacity: 3
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 2
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

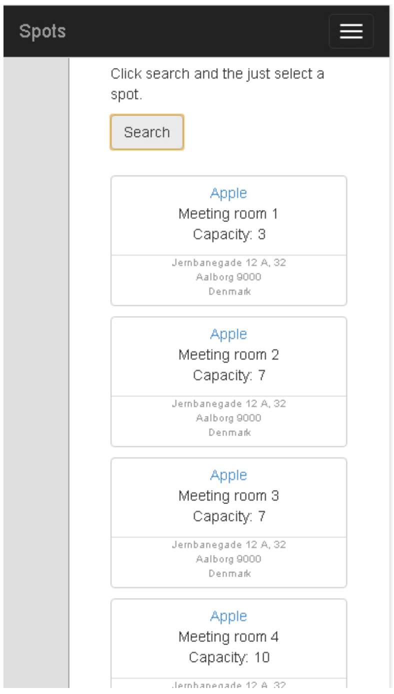
Apple
Meeting room 3
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 4
Capacity: 10

Apple
Meeting room 5
Capacity: 10

Apple
Meeting room 6
Capacity: 20

Figure 37 Web page opened on iPad Pro (Medium device)



Click search and the just select a spot.

Search

Apple
Meeting room 1
Capacity: 3
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 2
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 3
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 4
Capacity: 10

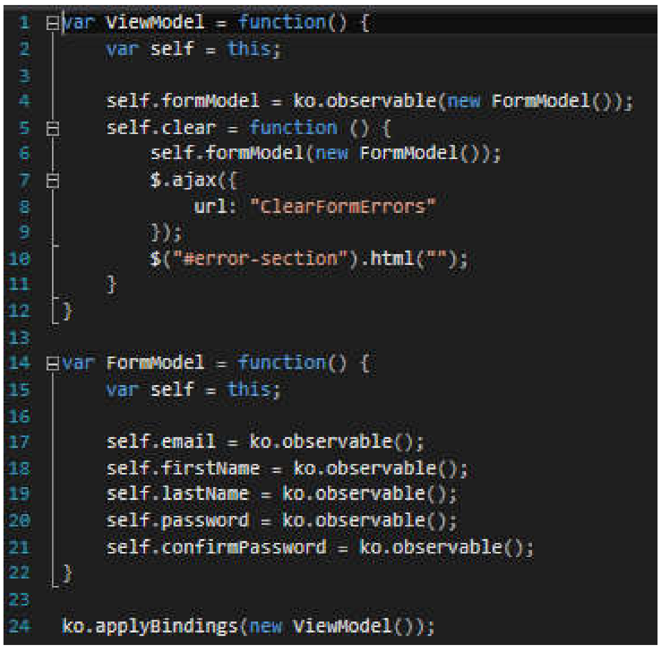
Figure 38 Web page opened on iPhone 6 (extra small device)

3.4.5 Knockout

Knockout is a JavaScript library that helps you to create rich, responsive display and editor user interfaces with a clean underlying data model[81]. It provides a support to implement simpler and more maintainable UI. One of the main features of knockout is the dependency tracking - it ensures, that the right parts of the UI update automatically, whenever a data model changes. In addition, it uses declarative binding, which is a simple way to connect UI parts to the data model.

Knockout uses the Model-View-ViewModel pattern, which is explained in detail in chapter 3.1.2. The ViewModel is essentially any JavaScript object. In order to add properties to the ViewModel, so that they will be updated automatically when parts of the ViewModel change, they have to be added as observables. Observables are JavaScript objects that can notify subscribers about changes, and can automatically detect dependencies[82]. To connect the ViewModel to the markup, a data-bind attribute is added to the HTML code.

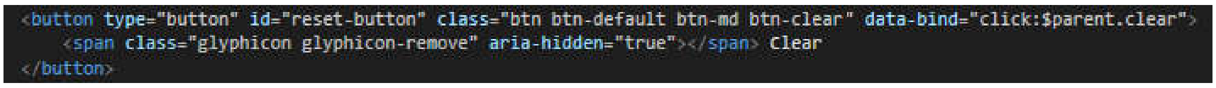
An example of knockout usage in the application is a simple ViewModel used for account registration. It consists of a form model and a function that replaces current form model with new empty instance. In addition, the clear function sends an AJAX request to the server and clears validation errors. Since the data-bind attribute is not native to HTML, knockout has to be activated, calling the applybindings function.



```
var ViewModel = function() {
    var self = this;

    self.formModel = ko.observable(new FormModel());
    self.clear = function () {
        self.formModel(new FormModel());
        $.ajax({
            url: "ClearFormErrors"
        });
        $("#error-section").html("");
    }
}

var FormModel = function() {
    var self = this;

    self.email = ko.observable();
    self.firstName = ko.observable();
    self.lastName = ko.observable();
    self.password = ko.observable();
    self.confirmPassword = ko.observable();
}

ko.applyBindings(new ViewModel());
```

Figure 39 Account registration ViewModel

Finally, a data-bind attribute has to be added to the markup. In this example, a click event is added to the reset button, which then calls the view model's clear function.



```
<button type="button" id="reset-button" class="btn btn-default btn-md btn-clear" data-bind="click:$parent.clear">
    <span class="glyphicon glyphicon-remove" aria-hidden="true"></span> Clear
</button>
```

Figure 40 Example where click event is added to the reset button

[81] From http://knockoutjs.com/documentation/introduction.html (Retrieved in 12/2016)
[82] From http://knockoutjs.com/documentation/observables.html (Retrieved in 12/2016)

3.5 Communication

This chapter covers how the communication outside and inside of the front-end and back-end systems is handled in the project. Next few subchapters will go deeper in the topic and explain OWIN, Katana and AJAX.

3.5.1 System.Web

System.Web is an assembly, which contains all of the ASP.NET functionality. This monolithic approach faces several problems. One of them are not that frequent releases. Any new ASP.NET functionality that can be released has to be part of the framework release. This means that any new features of ASP.NET are dependent on the framework. In addition, System.Web has a very tight coupling to IIS. This creates a dependency between the web server and the web framework, therefore there is only one hosting option for development.

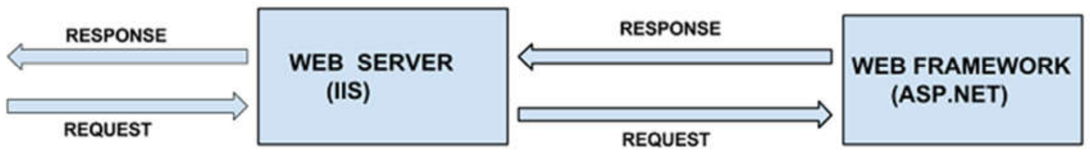


Figure 41 Tight coupling between the IIS and ASP.NET

3.5.2 OWIN

The OWIN (Open Web Interface for .Net) specification provides a solution to the above mentioned problems. OWIN defines a standard interface between .NET web servers and web applications. Its goal is to decouple the web server and the application[83].

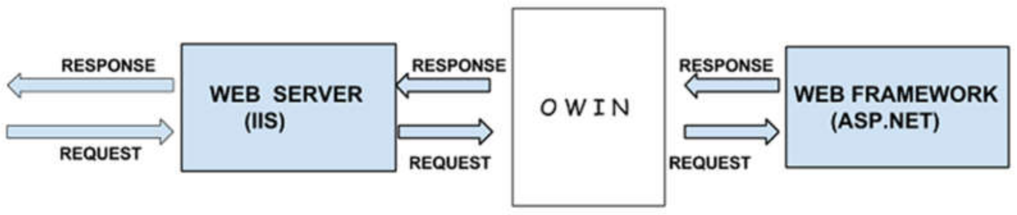


Figure 42 OWIN approach

[83] Adapted from https://www.codeproject.com/Articles/826757/Understanding-OWIN-and-Katana (Retrieved in 12/2016)

This approach provides several benefits, such as being lightweight, flexible and portable. The project application, build upon OWIN specification, consists of several components[84]:

- Host is responsible for starting the process. It also creates the pipeline by fetching the information from the application.
- Server binds to a port and listens for the requests. Once it receives the request it passes the request in the pipeline that the application has configured.

For both Host and server the project application uses IIS.

Application framework provides a development model to the project application. For example, the application uses MVC.

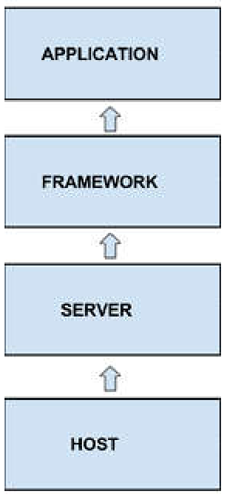
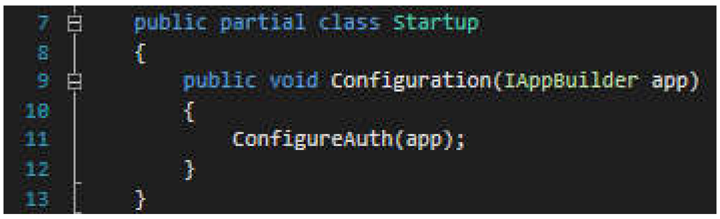


Figure 43 Components of the OWIN specification

3.5.3 Katana

Because OWIN is just a specification, it does not have the concrete implementation. This is what Katana provides. Katana is a set of open source OWIN components built by Microsoft .It is used to build OWIN based web applications[85].

In the project case, OWIN host communicates with the application before loading the server. Host needs information from the application about how to create the OWIN pipeline. This pipeline is used to process the request. When the application loads, Katana looks for a method with signature “void Configure (IAppBuilder app)” within a Startup class.



```
public partial class Startup
{
    public void Configuration(IAppBuilder app)
    {
        ConfigureAuth(app);
    }
}
```

Figure 44 Configuration method discovered by Katana

3.5.4 AJAX

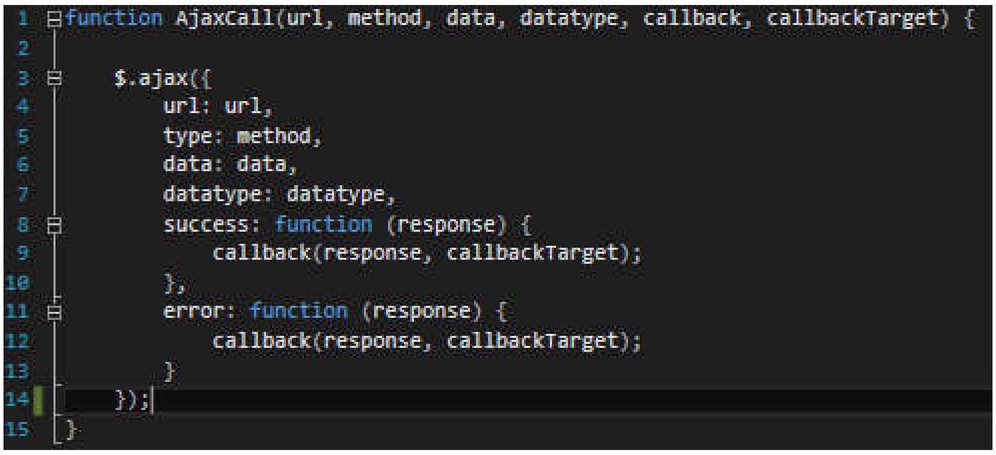
AJAX stands for Asynchronous JavaScript and XML[86]. It is a technique to create more interactive web applications with the help of XML (data do not have to necessarily be in the XML format - JSON or plain text can be used as well), HTML, CSS, and JavaScript.

AJAX provides a possibility to update a web page without the need to reload it. Additionally, it can request or receive data from the server or send data to the server.

Writing regular AJAX code can be tricky, because different browsers have different syntax for AJAX implementation[87]. However, with the help of jQuery library, AJAX is very simple and easy to use, because it takes the responsibility to handle the browser differences.

[84] Components explanation from https://www.codeproject.com/Articles/826757/Understanding-OWIN-and-Katana (Retrieved in 12/2016)
[85] Quotes from https://www.codeproject.com/Articles/826757/Understanding-OWIN-and-Katana (Retrieved in 12/2016)
[86] Quote from https://www.tutorialspoint.com/ajax/what_is_ajax.htm (Retrieved in 12/2016)
[87] Quote from http://www.w3schools.com/jquery/jquery_ajax_intro.asp (Retrieved in 12/2016)

Even with the usage of jQuery, the AJAX implementation can be broad and with a large number of uses a lot of duplicate code can be produced. In order to overcome this, there is only one method defined that handles AJAX requests. The method has several parameters that are necessary to perform the AJAX request - such as URL, method (get, post, …), data, datatype (for example JSON), callback function that will be called when an AJAX request is finished and target element ID that will contain the response. The response can have various forms, it can be a plain text, JSON, or even a partial view.

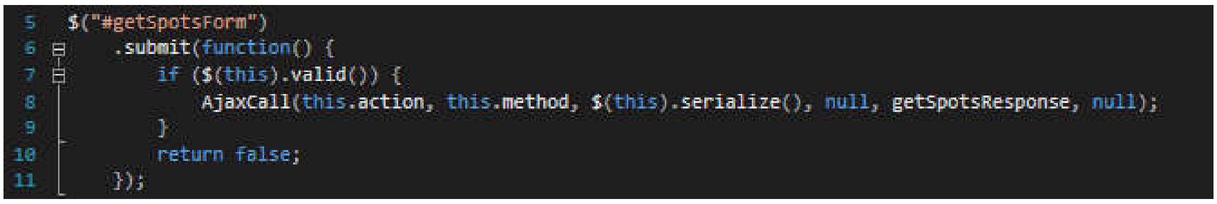


```
function AjaxCall(url, method, data, datatype, callback, callbackTarget) {

    $.ajax({
        url: url,
        type: method,
        data: data,
        datatype: datatype,
        success: function (response) {
            callback(response, callbackTarget);
        },
        error: function (response) {
            callback(response, callbackTarget);
        }
    });
}
```

Figure 45 AJAX method definition

Due to this, the implementation of AJAX is reduced to a single line of code. In some cases, instead of the usage of the target element to display the response, it was directly assigned to an observable using knockout.

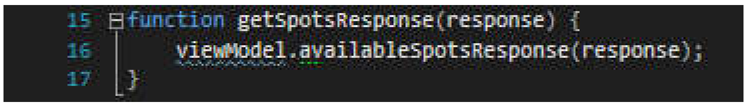
For example, AJAX is used when a form called “getSpotsForm” is submitted. When a response is received, the method “getSpotsResponse” is called.



```
$("#getSpotsForm")
    .submit(function() {
        if ($(this).valid()) {
            AjaxCall(this.action, this.method, $(this).serialize(), null, getSpotsResponse, null);
        }
        return false;
    });
```

Figure 46 Example of the AJAX implementation

The callback function takes the response itself as a parameter and assigns it to an observable.



```
function getSpotsResponse(response) {
    viewModel.availableSpotsResponse(response);
}
```

Figure 47 AJAX callback function

3.6 Testing

Testing is essential for software development as it checks is the program doing what it should be doing and to find bugs before the software is released. Program testing, where the system is executed using simulated test data, is the principal validation technique[88].

There are many different ways to conduct the testing of the software, and in next subchapters focus is on developing the testing strategy and going deeper into it.

3.6.1 Test pyramid

The test pyramid was used for the development of the right testing strategy. The test pyramid serves as a visual way of where the test effort should be focused.

Unit Tests make up the largest section of the pyramid, forming a solid base. Unit tests are the easiest to create and have the biggest value. The middle of the pyramid consists of Integration Tests. In the upper part of the pyramid, there are automated GUI tests. Finally, at the very top, manual tests take place.

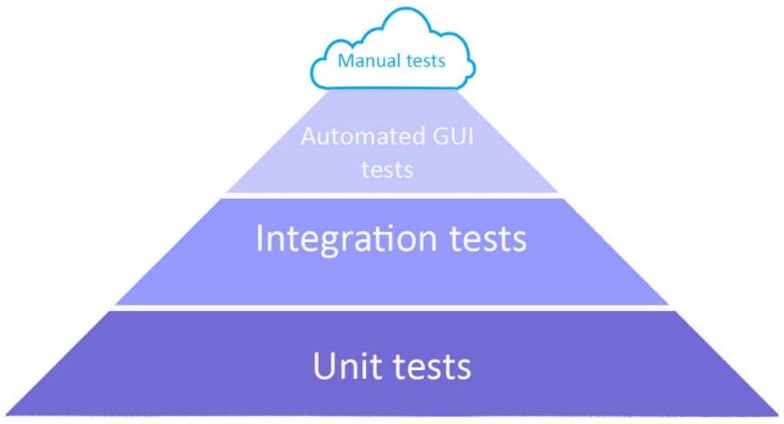


Figure 48 The test pyramid

One of the common mistakes that happen within the organizations is that instead of the pyramid they end up with ice-cream cone anti-pattern. This happens when there is not enough low-level testing (unit, integration and component), too many tests that run through the Graphical User Interface (GUI) and an even larger number of manual tests[89].

3.6.2 Unit testing

The primary goal of the Unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as expected[90]. There are several benefits associated with unit testing:

- Finding software bugs early. By adding Unit tests to the software build process, or as part of the continuous integration process, as the code base grows larger, these tests run automatically. When a failure occurs, either the failure is caused by a bug in the code or a

[88] Adapted from: Ian Sommerville (2016): Software Engineering, page 58
[89] From https://www.thoughtworks.com/insights/blog/introducing-software-testing-cupcake-anti-pattern (Retrieved on 12/2016)
[90] From https://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx (Retrieved on 12/2016)

problem with the actual unit test. Either way, pinpointing the location of failure is easily traced. Since unit test failures alert the developers before the code is pushed to testers or clients, it is still early in the development cycle, making the fix less costly than if found later in the development cycle[91]

- Facilitates Change. Unit tests ensure that the code still functions properly as the code base changes with code refactoring and as the code base grows. Code refactoring is the process of restructuring the existing code without changing its original behaviour[92]
- Improves code quality. Unit test can also uncover code design issues. For example, in the second iteration of the project, Unit testing uncovered the need to use dependency injection and inversion of control to get rid of direct dependencies between application components.

The example of unit testing in this project can be viewed on figure 65 in appendix.

3.6.2.1 *Testing framework*

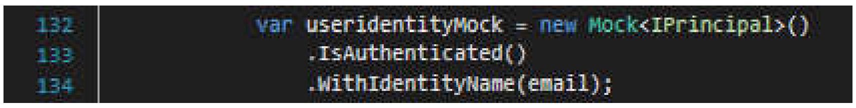
There are several frameworks that could have been selected for writing unit tests. Two prime candidates that were considered are the Microsoft Visual Studio Test and the NUnit test framework. Finally NUnit was selected, because it has several advantages over MS-Test[93]:

- More readable assert methods:
    - Assert.AreEqual(expected, actual)
    - Assert.That(actual, Is.EqualTo(expected))
- Suite attribute can aggregate tests and execute them separately
- NUnit has frequent version updates - MS-Test has only one per VS version
- Nunit has many integrated runners, including resharper
- Expected exception message assertion - can be done using attribute in NUnit but must be done using Try-Catch in MS-Test
- NUnit allows for parameterized tests

3.6.2.2 *Mocking library*

Mock objects are instances of test-provided classes that simulate the behaviour of external components. Mock objects isolate the application code under test. They create conditions that are otherwise difficult to produce[94].

MOQ library was used to create mock objects, while addition extension was used to improve the code readability. For example, for the unit test to get to the authorized controller, it is necessary to bypass authentication. To do this a mock object of user identity is used. Using the concepts of extension methods and method chaining a mock object of the IPrincipal interface is created.

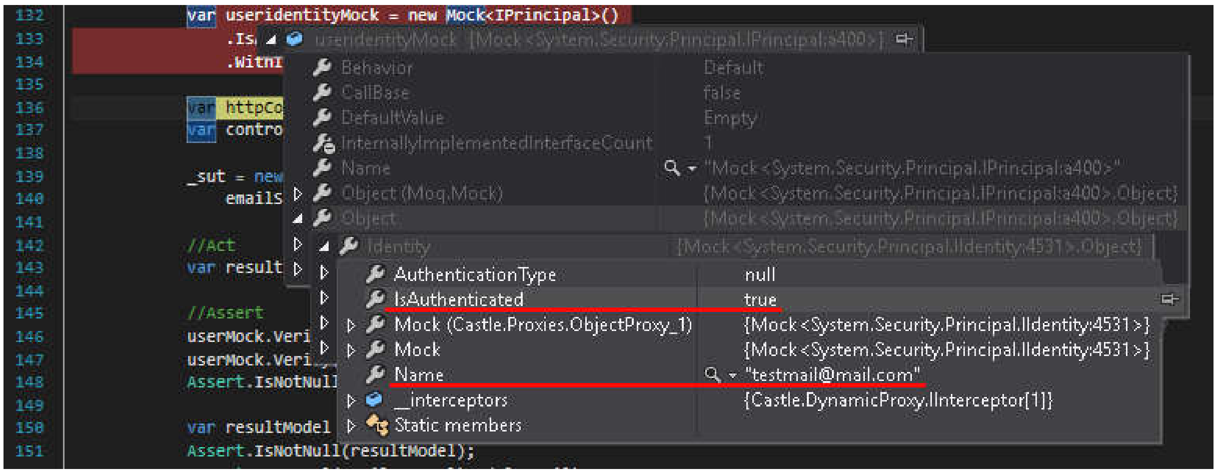


```
            var useridentityMock = new Mock<IPrincipal>()
                .IsAuthenticated()
                .WithIdentityName(email);
```

Figure 49 Example of initialization of a mock object

[91] Paragraph from http://www.seguetech.com/the-benefits-of-unit-testing/ (Retrieved on 12/2016)
[92] Paragraph from http://www.seguetech.com/the-benefits-of-unit-testing/ (Retrieved on 12/2016)
[93] Adapted from http://stackoverflow.com/questions/1554018/unit-test-nunit-or-visual-studio (Retrieved on 12/2016)
[94] Quotes from https://msdn.microsoft.com/en-us/library/ff650441.aspx (Retrieved in 12/2016)

When the object is created, only the properties that are necessary for a particular unit tests are set up. The IsAuthenticated property is set to true and identity email is set to "testmail@mail.com" - a string constant from the arrange section of the unit test. Other object's properties have not been set, therefore for example AuthenticationType property returns null.



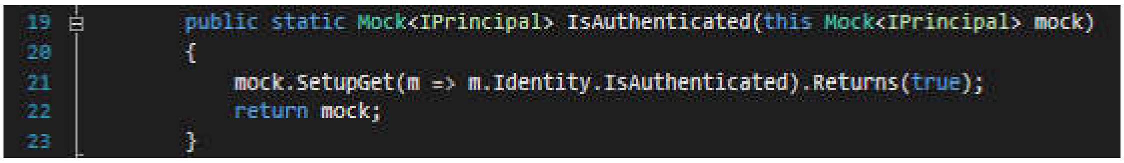
```
var useridentityMock = new Mock<IPrincipal>()
    .Is
    .WithI

var httpCo
var contro

_sut = new
    emails

//Act
var result

//Assert
userMock.Veri
userMock.Veri
Assert.IsNotNull

var resultModel
Assert.IsNotNull(resultModel);
```

```
useridentityMock  {Mock<System.Security.Principal.IPrincipal:a400>}
  Behavior                              Default
  CallBase                              false
  DefaultValue                          Empty
  InternallyImplementedInterfaceCount   1
  Name                                  "Mock<System.Security.Principal.IPrincipal:a400>"
  Object (Moq.Mock)                     {Mock<System.Security.Principal.IPrincipal:a400>.Object}
  Object                                {Mock<System.Security.Principal.IPrincipal:a400>.Object}
    Identity                            {Mock<System.Security.Principal.IIdentity:4531>.Object}
      AuthenticationType                null
      IsAuthenticated                   true
      Mock (Castle.Proxies.ObjectProxy_1) {Mock<System.Security.Principal.IIdentity:4531>}
      Mock                              {Mock<System.Security.Principal.IIdentity:4531>}
      Name                              "testmail@mail.com"
      __interceptors                    {Castle.DynamicProxy.IInterceptor[1]}
      Static members
```

Figure 50 IPrincipal mock object example

Extension methods are defined as static methods, but are called by using instance method syntax. Their first parameter specifies which type the method operates on, and the parameter is preceded by the "this" modifier. Extension methods are only in scope when the namespace is imported in source code with a using directive[95].

The IsAuthenticated extension method is present within the "PrincipalExtensions" class, among other methods that set up different properties of the mock object.



```
public static Mock<IPrincipal> IsAuthenticated(this Mock<IPrincipal> mock)
{
    mock.SetupGet(m => m.Identity.IsAuthenticated).Returns(true);
    return mock;
}
```

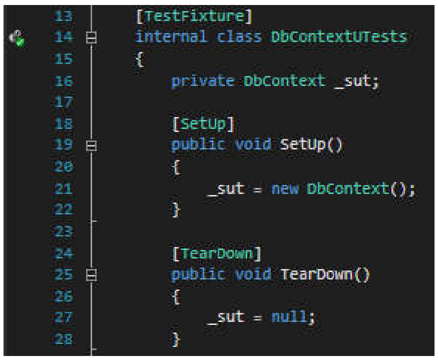
Figure 51 Extension method example

[95] Paragraph from https://msdn.microsoft.com/en-us//library/bb383977.aspx (Retrieved in 12/2016)

3.6.2.3 Structure of the test class

A test class does not contain only test methods. Every test class should contain a system under test (SUT) object. SUT object represents all actors (one or more classes) that are going to be tested and are neither mocks nor stubs[96].

For most test classes it is necessary to have a setup and teardown methods, which execute before and after every unit test. This approach can get rid of repeated code and dispose objects after the test is finished.



```
[TestFixture]
internal class DbContextUTests
{
    private DbContext _sut;

    [SetUp]
    public void SetUp()
    {
        _sut = new DbContext();
    }

    [TearDown]
    public void TearDown()
    {
        _sut = null;
    }
```

Figure 52 SUT object initialization and disposal

3.6.2.4 Structure of the Unit test

Unit test itself consists of three main parts. In the Arrange part objects are initialized and value of the data that is passed to the method under test is set. The Act section invokes the method under test with the arranged parameters. The Assert section verifies that the action of the method under test behaves as expected. This approach clearly separates setup and verification test steps[97].

3.6.3 Integration tests

Integration testing is similar to unit testing in that tests invoke methods of application classes in a unit testing framework. However, integration tests do not use mock objects to substitute implementations for service dependencies. Instead, integration tests rely on the application's services and components. The goal of the integration tests is to exercise the functionality of the application in its normal run-time environment[98].

3.6.4 Automated tests

An automated testing tool is able to playback pre-recorded and predefined actions on the GUI level, compare the results to the expected behaviour and report the success or failure of these manual tests to a test engineer. Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing. Because of this is automated software testing an essential component of successful development[99].

[96] Adapted from http://stackoverflow.com/questions/7321407/what-is-sut-and-where-did-it-come-from (Retrieved in 12/2016)
[97] Adapted from https://msdn.microsoft.com/en-us/library/hh694602.aspx (Retrieved in 12/2016)
[98] Quotes from https://msdn.microsoft.com/en-us/library/ff647876.aspx (Retrieved in 12/2016)
[99] Adapted from https://smartbear.com/learn/automated-testing/ (Retrieved in 12/2016)

3.6.4.1 Selenium web driver

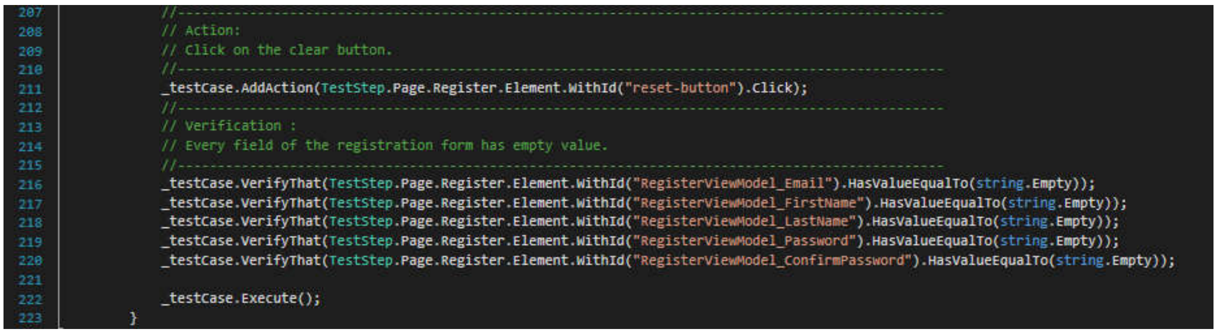
WebDriver is a web automation framework that allows you to execute your tests against different browsers[100]. The web drive provides a number of useful methods to test the web application: redirecting within the browser, locating and analysing HTML elements, performing actions (e.g. clicking or inputting values to the text fields).

3.6.4.2 The project structure

In order to produce a clear and maintainable automated tests, the automated test project differentiates between two parts - the tests themselves and the test step library.

The tests themselves are written using a command pattern, which is described in detail in chapter 3.1.4.2. In addition, there is a separation between the selenium web driver and the tests. The tests are not calling the driver directly. Instead, they consist of the test steps that are present within the test step library. The test step can be either an action test or a verification test step.

Every test has exactly one test case object. A test starts with its initialization. Throughout the test, the test steps are added into a queue of the test case. Finally, every test case has to call the execution method.



```
            //--------------------------------------------------------------------------------------------
            // Action:
            // Click on the clear button.
            //--------------------------------------------------------------------------------------------
            _testCase.AddAction(TestStep.Page.Register.Element.WithId("reset-button").Click);
            //--------------------------------------------------------------------------------------------
            // Verification :
            // Every field of the registration form has empty value.
            //--------------------------------------------------------------------------------------------
            _testCase.VerifyThat(TestStep.Page.Register.Element.WithId("RegisterViewModel_Email").HasValueEqualTo(string.Empty));
            _testCase.VerifyThat(TestStep.Page.Register.Element.WithId("RegisterViewModel_FirstName").HasValueEqualTo(string.Empty));
            _testCase.VerifyThat(TestStep.Page.Register.Element.WithId("RegisterViewModel_LastName").HasValueEqualTo(string.Empty));
            _testCase.VerifyThat(TestStep.Page.Register.Element.WithId("RegisterViewModel_Password").HasValueEqualTo(string.Empty));
            _testCase.VerifyThat(TestStep.Page.Register.Element.WithId("RegisterViewModel_ConfirmPassword").HasValueEqualTo(string.Empty));

            _testCase.Execute();
        }
```

Figure 53 Automated tests example

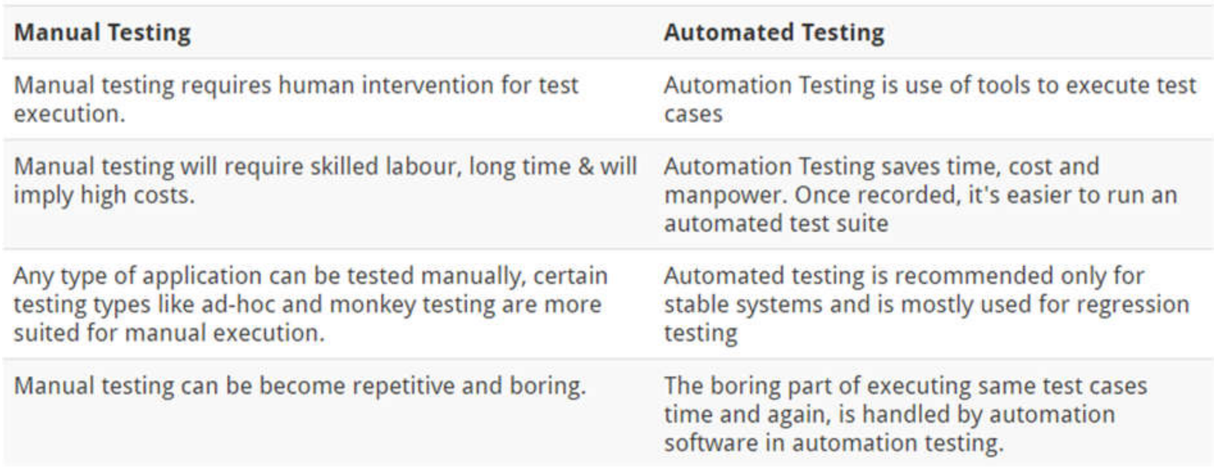
3.6.5 Manual tests

Manual Testing is a type of Software Testing where Testers manually execute test cases without using any automation tools. It is the most primitive of all testing types and helps find bugs in the software system[101]. A test plan document acts as a guide to the testing process. This is done to have full test coverage. Manual tests have been performed according to manual test specification, described in the chapter 3.3.4.

[100] From http://www.guru99.com/introduction-webdriver-comparison-selenium-rc.html (Retrieved in 12/2016)
[101] Paragraph from http://www.guru99.com/manual-testing.html (Retrieved in 12/2016)

3.6.6 Comparing manual and automated testing

There are several main differences between manual and automated testing, which influence choosing the type and intensity of the testing for the project. By taking a look on the figure below it is obvious that automated can bring some interesting benefits for stable systems and on the long-term can save resources, while manual testing is more used in the cases to detect design flaws and gather feedback from the user point of view.



Manual Testing	Automated Testing
Manual testing requires human intervention for test execution.	Automation Testing is use of tools to execute test cases
Manual testing will require skilled labour, long time & will imply high costs.	Automation Testing saves time, cost and manpower. Once recorded, it's easier to run an automated test suite
Any type of application can be tested manually, certain testing types like ad-hoc and monkey testing are more suited for manual execution.	Automated testing is recommended only for stable systems and is mostly used for regression testing
Manual testing can be become repetitive and boring.	The boring part of executing same test cases time and again, is handled by automation software in automation testing.

Figure 54 Differences between manual and automated testing

4 Development of the system

Development of the system is the part that took the most of the project time. To be more precise, it took 7 weeks starting from November 14th with the planning sprint - Sprint 0 and ending on December 30th with the last development sprint – Sprint 6.

As mentioned in chapter 2.3, the methods used for the development of this project are consisting of a mix of three different methodologies: Scrum and Extreme Programming are two main methodologies, while from Kanban only parts for measuring and monitoring the progress are used.

First step, to set up the development, is done in Sprint 0. The development is done in sprints after. All steps are described in the subchapters, reflecting the process sprint by sprint.

4.1 Sprint 0

During the Sprint 0 the main focus was to prepare everything to be able to start creating the product solution. Preparation in question consisted of setting up the technical environment and setting up the process rules.

Regarding the technical environment, several decisions were made:

- Version control and source control: Team Foundation Server
- Build services: MongoDB, Microsoft SQL Server 2014 and Visual Studio 2015
- Test Framework: NUnit test
- Libraries: jQuery and Knockout

One of the pre-agreed decision was also establishing an email which will be used for testing purposes of some use cases:

- Account name: thespots.testapp@gmail.com
- Password: thespots

For the process part, during Sprint zero, user stories were written down and organized into a product backlog, the plan for next iteration (Sprint one) was made as well as decision how the progress will be measured.

4.1.1 User stories

To be able to start the project, it is important to know the system requirements. Functional requirements for this project are described by the user stories. They describe only the functionality valuable for the user or product owner. They are written using business terms by the product owner. A user story consists of several parts: ID, story name, actor (who/what is doing the action), conversation (I want to do something so that there is a result of something else), acceptance criteria, priority and estimate. The estimate is added to the user story by the team in the planning poker phase.

After all of the user stories are ready, it is time to perform the user story ranking procedure. This is simply creating a list of all of the user stories ranked from the highest, the most important, to the lowest, the least important. To do prioritization easier, the MoSCoW prioritization was used, where it was determined which user stories MUST be a part of the final solution, SHOULD be a part of the final solution, COULD be a part of the solution if there will be enough time) and WON'T be a part of this solution, but could be implemented in the future solutions.

MUST	SHOULD	COULD	WON'T
Register user	Password forget	Delete profile	Additional fields in the spots
Log in	Profile confirmation	Add a friend	Additional fields in the profile
Log out	Automatic deletion of the profile	Log in via FB or Google +	Captcha
Edit profile	View other user profile	Register via FB or google +	Log out timer
Create business	View event page	Edit location	Log out confirmation
Creating location	Comparing availability (users)	List events (for a spot)	Edit profile confirmation
Creating a spot	Edit the event	List events (for a user)	Confirming location
Search a spot availability by city and date	Edit a spot	Send invitation for the event (friends)	Search location by city
Comparing availability (spots)	Calendar (spots)	Send invitation for the event (employees)	
Create an event	Calendar (users)	Visible profile	
	Delete a spot		
	Delete a location		
	Delete the event		
	Add an employee		
	Remove employee		

Figure 55 User stories MoSCoW prioritization

The list was created with the top 10 user stories, which must be implemented during this project, because they are crucial for the base of the product to be functional. Others were added accordingly to complete the product backlog, which concluded this phase.

Next step needed to be done is the estimation of the user stories and for that the story points system was used. Story point is actually a measure of complexity and is used in relative estimation techniques[102]. To be more precise, the value of the story point is not necessarily same as the value of time, for example, 1 story point does not need to be equivalent to 1 hour. Instead, it means that if there are user story with 1 story point and user story with 2 story points, the latter one is two times more complex than the first one.

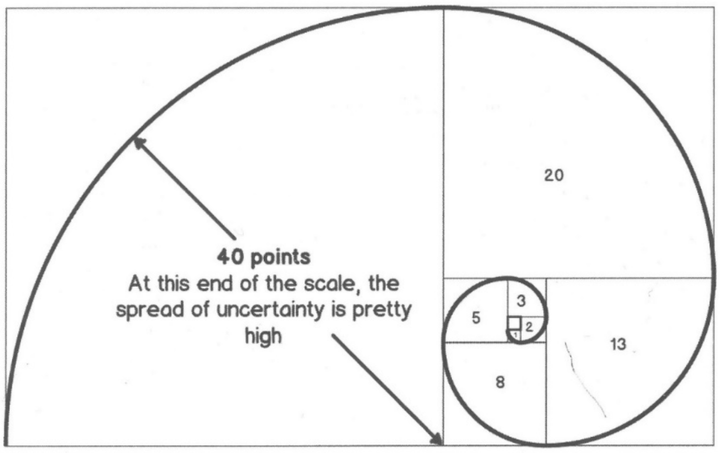

40 points
At this end of the scale, the spread of uncertainty is pretty high
20
13
8
5
3
2
1

Figure 56 Modified Fibonacci sequence as a "golden spiral"[103]

Estimating is a difficult job to do, because it requires to literally guess how much time and effort needs to be invested for a specific user story. If done incorrectly, it can influence the time of the project completion and its quality. In this project, the estimation technique called planning poker was used. Planning poker is done by cards with modified Fibonacci numbers of 0.5,1,2,3,5,8,13,20,40,100, ∞ (infinity) are used.

[102] Quote from Klaus Nielsen: *I Am Agile – Knowledge That Sets You Apart*, page 168
[103] Figure from Ilan Goldstein: Scrum shortcuts without cutting corners: agile tactics, tools & tips, Page 70

As it can be seen in the figure[104] to the right, this approach works in the following way:

- Product owner presents the user story
- After discussion (where developers ask questions to the product owner), each developer chooses from his own deck a card that represents their estimate of how much work should be invested in the story
- All estimates are kept private until each participant chooses a card
- Finally, all estimates are revealed and if the point value differs, the discussion between the developers is held. Then they choose the cards again and repeat the process until the point value for the user story is in the agreement and they can continue to the next user story.

It was decided to estimate only the top user stories in the product backlog that must be in the solution. There is no point in estimating all user stories before it is known whether they will be developed or not. Next step was to plan the first iteration.

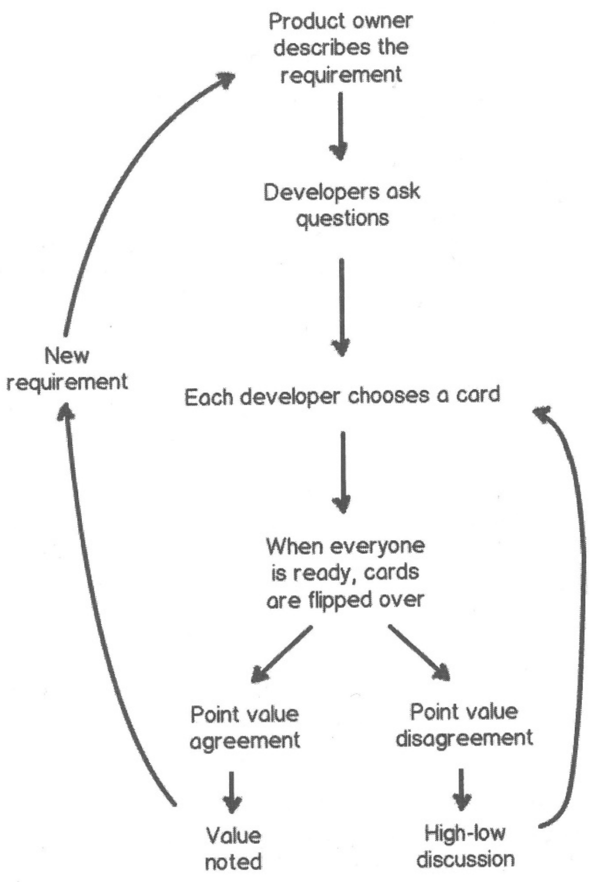

Product owner describes the requirement
Developers ask questions
Each developer chooses a card
When everyone is ready, cards are flipped over
Point value agreement
Point value disagreement
Value noted
High-low discussion
New requirement

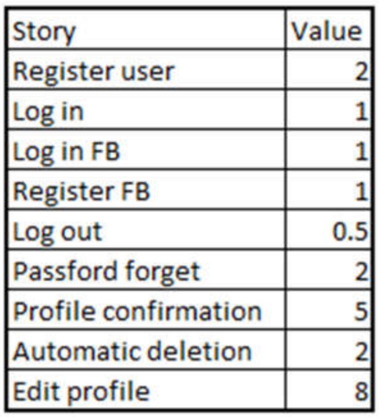
Figure 57 Planning poker circle

4.1.2 Iteration planning

At the end of each iteration, the product owner decides which user stories should be added to the next iteration. On the other hand, the team decides the number of the story points they can take into the iteration, based on the efficient time (time without breaks, illness and every other activity not relevant for the project) they can put into the iteration.

Because this project is being done by a two-member team, the decision was made that 35 hours of the efficient work can be put into the project by each of them, based on the 5-day working week. Additional 5 hours a week will be allocated for the stand-up meetings, regular meetings, sprint review meeting and sprint retrospective meeting, which rounds up to a maximum of 40-hour working week for each member.

The next step for the iteration planning was to decide how many story points could be done in the first sprint. It was decided that around 20 story points could be done.



Story	Value
Register user	2
Log in	1
Log in FB	1
Register FB	1
Log out	0.5
Passford forget	2
Profile confirmation	5
Automatic deletion	2
Edit profile	8

Figure 58 Sprint backlog for sprint 1

[104] From Ilan Goldstein: *Scrum shortcuts without cutting corners: agile tactics, tools & tips*, Page 72

That is why for the sprint one, after reviewing the user stories priorities, nine user stories with a total value of 22.5 story points were planned to be done. The user stories have been split into 43 tasks.

4.1.3 Burn down chart and velocity

One of the most common ways to demonstrate progress in the agile software development is the burn down chart. Burn down chart shows the amount of work remaining across time[105]. On the chart, it is possible to see the total number of story points in the sprint compared with the actual number of story points achieved during the sprint. It is a great tool to illustrate the progress by looking at the chart, and additional user stories could be added or removed.

The velocity is a way to measure progress – usually done by calculating how many story points have been successfully finished in the iteration. To assist monitoring the progress, the velocity can be observed by following the line of the story points done on the burn down chart in the sprint. It is usually used to better plan future sprints.

4.2 Sprint 1

Sprint 1 started on the 21st of November and ended on the 25th of November. Sprint one was used as a base test for all of the future Sprints. Daily schedule could have three items: stand-up meeting, regular meeting (only held if needed) and working. To put it as a time schedule, it looked like:

- 09:00-09:15 Stand-up meeting
- 09:15-10:00 Regular meeting (if needed)
- 10:00-17:00 Working

The working time will be more flexible (e.g. it can be from 10-14 and then from 17-20) but it cannot exceed more than 7 hours every day. Sprint retrospective meeting and sprint planning meeting are held on Sunday.

As this was the first Sprint, there were small issues from the technical point of a view, which should have been resolved during the Sprint zero. One of them was that Andrej's laptop could not connect to the SQL server and he had to invest some time into resolving this issue. The second issue was that some administration rules in the VSTS were not set up properly.

During the 3rd day's Stand-Up meeting it was noticed from looking at the burn down chart (figure 59) that it will not be possible to finish all of the planned user stories from the sprint backlog. That is why it was decided to postpone user story "edit profile" for the Sprint 2 and completely remove "automatic deletion" user story. "Automatic deletion" does not have the business value, which would justify putting in the working hours during the project. Therefore, it was returned at the bottom of the product backlog.

[105] From Ken Schwaber: *Agile Project Management with Scrum*; Page 11

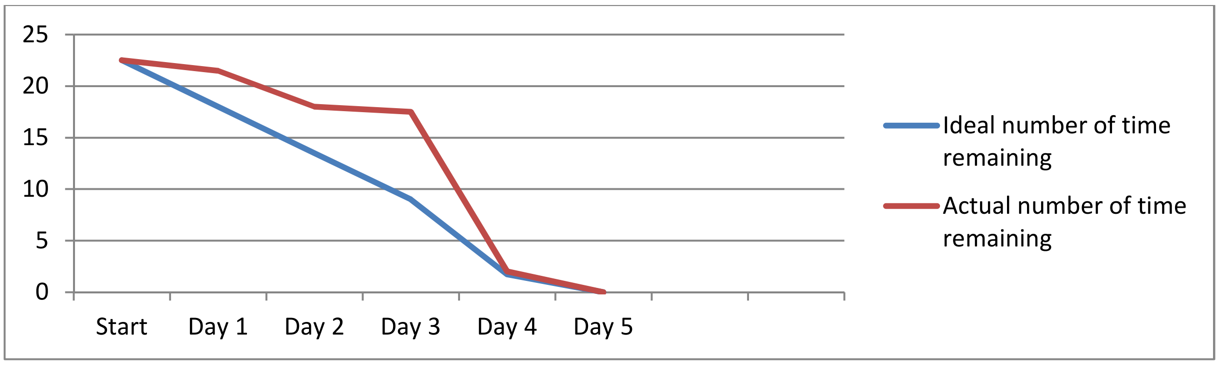


Figure 59 Sprint one – Burn down chart

During the 5th day's stand-up meeting, unit and integrated testing from the remaining two user stories that were planned were dropped after a short discussion that the time needed to solve these tasks can be used in different ways. If these tasks are implemented to every user story, in a long-term overview of the process it would require at least one more person on the team.

As planned, on Sunday the sprint retrospective meeting was held. The only conclusion was that the approach during the planning meeting for sprint one was overoptimistic – the amount of story points that could be accomplished during the first Sprint was simply too high. This conclusion was important for the upcoming sprint planning meeting. The velocity for the sprint was 336 minutes per story point, but the main reason for that speed could be due to the technical issues that occurred during the sprint. By removing the testing tasks from the future sprints, it should be possible to do around 18 story points in next one.

During the sprint planning meeting held after the retrospective meeting, a decision that two additional user stories are to be written down – "create business" and "view other user profiles". These stories were skipped from the initial requirement planning, but are important for the product. The first one is needed for the system to work properly (magical triangle of business – location – spot), while the second one has important business value for the system.

It was also decided that for the second sprint, four user stories with 21 tasks and a total value of 18 story points are planned to be done. This was decided based on the knowledge from the first sprint.

4.3 Sprint 2

Sprint two started on the 28th of November and ended on the 2nd of December. This Sprint was more stressful as Andrej was unfortunately unavailable for a couple of days during the week as he was travelling. This also influenced the number of the tasks that he was doing per day, like creating the use case descriptions for user stories that are being developed.

During the sprint it was obvious that the use case description tasks are important for the software development, because they help the developer to visualize how the code should look like. Since they were not written down earlier in the sprint, one user story ("create business") was postponed for the next sprint. The decision was made on the 3rd stand-up meeting as a precaution to make sure that all other user stories would be done on time.

From the programming point of view, some complications occurred during the development of the Edit profile user story. Firstly, additional research had to be done to correctly integrate the

Model-View-ViewModel pattern using Knockout library. It was a completely new approach, which substituted the standard usage of JavaScript and jQuery. The time investment was costly, however the usage of MVVM led to a huge reduction of the JavaScript code and simplification of the markup.

Another problem that was faced in this sprint was communication. So far the communication between the view and model was accomplished by form submission. The edit profile page required the usage of AJAX, in order to update the view without page redirection. In addition a deeper understanding of how to serialize model data was required. Once more, the research cost a lot of time, but it was beneficial in a long run, because same techniques were applied to a number of other user stories.

Due to the fact that additional time was spent on the technological research, and that use case testing for some user stories was written later, it was surprising that the velocity actually improved and now was 323 minutes per one story point.

The main decision of the retrospective meeting was that the requirement tasks need to be written down as soon as possible as they are important for the creation of the code. That is why all of the tasks labelled requirements would from then on be done first. Using the knowledge collected in the first two sprints the plan was to do only two user stories with a total value of 13 story points for Sprint 3. User stories have been split into 14 tasks.

4.4 Sprint 3

Sprint three started on the 5th of December and ended on the 9th of December. Learning from the previous sprints, first tasks done were creating the requirements and then all of the other tasks of the user story.

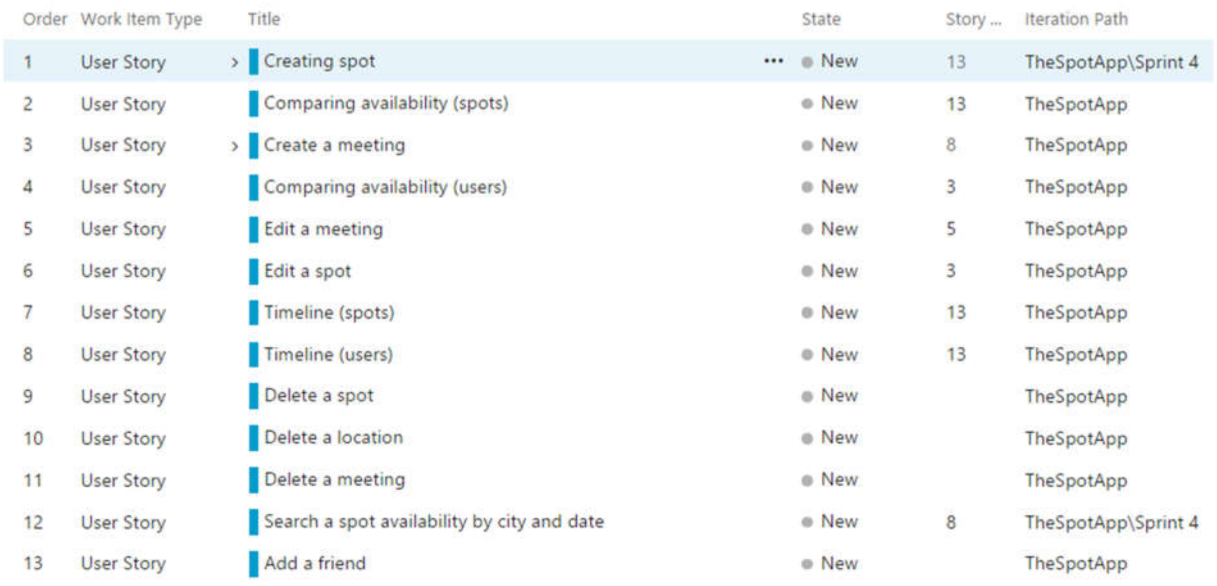
This is the first sprint where it was managed to finish everything that was planned. It was becoming obvious that the team capacity is around 13 story points per iteration with a tendency to improve – in every sprint there were some additional tasks that were not planned. Therefore, future iteration planning would slowly keep increasing the amount of the story points for the sprints.

In this sprint there were some unpredictable issues from the technology point of view. The biggest problem that had to be overcome were dependencies. For example the controller classes were dependent upon the model classes. This lack of abstraction caused problems when unit tests were written. It was difficult (or impossible) to mock object and actual instances had to be created instead. Besides, the fact that application components were directly dependent upon each other, was bad by itself.

In order to overcome problems, Dependency injection pattern was used. There were questions on how to use it, as the choice was between Management Extensibility Framework or Inversion of Control (IoC) container. Additionally, there are a number of IoC containers that could have been selected (Unity, Ninject, etc.) and it had to be considered, which to use. Finally, in order to implement dependency injection, Unity for MVC IoC container is the best fit. 10% of the total sprint time was used to adapt the application to these changes.

During the retrospective meeting, it was noticed that a lot of time was wasted for technology discussions. That is why the more time will be reserved for it in the next sprint, because there is still a lot of necessary researching and experimenting with the technologies. Besides, after implementation sometimes it is noticed that it could be done better and it is needed to refactor the code to hold that better solution.

For the fourth sprint something new will be done –a user story that is down the line by priority will be put higher and put on the sprint backlog. The user story was "Search a spot availability by city and date". This can be seen in the figure below (it was 12th in the planned order).



Order	Work Item Type	Title	State	Story ...	Iteration Path
1	User Story	Creating spot	New	13	TheSpotApp\Sprint 4
2	User Story	Comparing availability (spots)	New	13	TheSpotApp
3	User Story	Create a meeting	New	8	TheSpotApp
4	User Story	Comparing availability (users)	New	3	TheSpotApp
5	User Story	Edit a meeting	New	5	TheSpotApp
6	User Story	Edit a spot	New	3	TheSpotApp
7	User Story	Timeline (spots)	New	13	TheSpotApp
8	User Story	Timeline (users)	New	13	TheSpotApp
9	User Story	Delete a spot	New		TheSpotApp
10	User Story	Delete a location	New		TheSpotApp
11	User Story	Delete a meeting	New		TheSpotApp
12	User Story	Search a spot availability by city and date	New	8	TheSpotApp\Sprint 4
13	User Story	Add a friend	New		TheSpotApp

Figure 60 Product backlog at end of Sprint 3 (VSTS view)

The reason to pick an item that it is so low, is that both the "comparing availability (spots)" and "search a spot availability by city and date" are almost the same user story. The main difference is that the first one is done completely in the back-end and is, in fact, a part of the latter user story. Instead dividing them in two separate user stories, they will be implemented as one user story.

Since the sprint four will be focused on the manual testing, only two user stories with a total value of 16 story points will be developed. This is higher than in the previous sprints, but the team was confident that they can improve. There are 14 tasks to develop these user stories.

4.5 Sprint 4

Sprint four started on the 12th of December and ended on the 16th of December and it focused on building up two user stories, system analyses and testing. Although there were only two user stories, there was a lot of work that needed to be done. In the first user story, it was important to connect the businesses, locations and spots. The second user story, required a search tool, for finding available spots that could be booked. This was a huge challenge, because it required to figure out the algorithm for checking spots available at a certain time and location.

One of the issues that hit the team immediately on the first day was the way how the businesses and locations are stored in the database. Originally, locations were completely embedded inside a business. The fact that this constitutes a problem was discovered during the implementation of the search for the available spots. In order to find what spots are available, a city (location's field) had to be entered. With the original schema, this was impossible to do, because businesses were not known and selecting all of them would be a tremendous performance drawback.

Due to this reason, it has been decided, that embedding is not a suitable option. Locations were moved into a separate document, while business document contained their reference. This unplanned refactoring took one day that were unplanned for this sprint. Luckily, both user stories went really smooth, so they were implemented in the next three days.

Although it was managed to develop the user stories planned for this sprint, the time that was planned to be invested into the system analysis and testing was not done. This was not a crucial part of the sprint, but the main goal was to see should some of the items in the fifth sprint be refactored.

In the sprint retrospective, there was satisfaction with the process of this sprint. The only hiccup was that at the end approximately 20 % of the sprint planned for the functionality review was spent in fixing a connection that was supposed to be sorted out in the sprint before. Regardless of this small issue, the velocity of the project improved and it was 262 minutes per story point in this sprint. This was incredible improvement, because it is better for over one hour per story point compared to the previous sprint. Following this, it has been decided that in the fifth sprint two bigger user stories will be developed with a total value of 21 story points.

In this sprint planning meeting, for the first time, a deeper look was made into the product backlog as there were three main reasons to do it:

- Issue with the naming – it was noticed that there is some confusion with the naming for some user stories
- Adding additional user story
- Some of the user stories were overestimated at the beginning and they should be re-estimated

First part of the discussion was an issue with the naming and it was connected with the process for booking a spot. Originally it referred as the “meeting”. As meeting is “a gathering of people for a particular purpose[106]” due to the much broader possibilities of the system, it was decided to changed it to “event” which by itself has a broader meaning – “something important or notable that happens[107]”. According to this decision, the naming of user stories was corrected from “meeting” to “event” and all of the fields in the code will be checked in the upcoming sprint.

The second thing on the agenda was adding a new user story – view event page. From the product owner point of view, having the access to the event page has a big business value – it provides the possibility for users to overview the event information – the description, who created it, who is invited, location of the event and much more. It was estimated with 8 story points and it will be added to the next sprint.

Finally, the last and the longest part of the discussion was the re-estimations. It has been decided to go through the next upcoming user stories that should be done in the last two sprints and also potential user stories that could be added to these sprints. User stories in the question are:

- Create a meeting – this user story is quite complex, so it was re-estimated to 13 instead of 8 story points that it originally had
- Timeline (users) – this user story was pushed higher in the product backlog, because there will be a lot of unknown issues how exactly will the information (about the events and participants) be presented on the timeline. Since this is one of the most important functionality of the project, it was re-estimated to 20 story points (originally 13)

[106] From https://www.merriam-webster.com/dictionary/meeting (Retrieved on 12/2016)
[107] From https://www.merriam-webster.com/dictionary/event (Retrieved on 12/2016)

- Edit a meeting and edit a spot were classified of being the same complexity, so the new story point value for them is now 5
- Comparing availability and timeline for users were dropped from the product backlog for this project as were classified as too big to be added in the upcoming sprints

After careful consideration, “create the event” user story, alongside with the new user story “view event page” were added to the sprint backlog with a total number of nine tasks.

4.6 Sprint 5

Sprint five started on the 19th of December and ended on the 23rd of December and it evolves around slowly closing down the first part of the business value for the system – possibility to create the events.

This was finally a sprint that went completely as planned. There were no negative feedback on the retrospective meeting and it was the biggest achievement during this process – total value of 21 story points with a record velocity – 200 minutes for one story point.

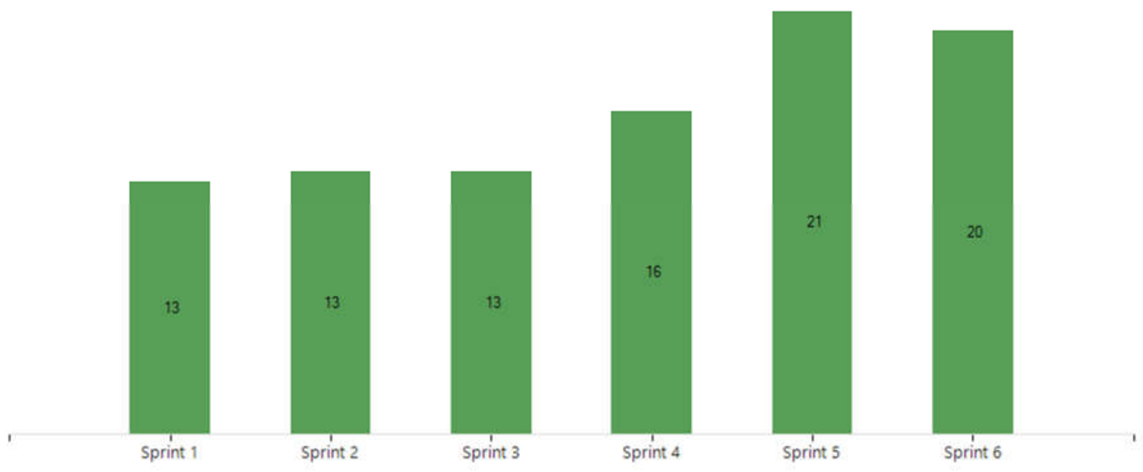


Figure 61 Velocity chart for the project

At the planning phase for the final, sixth sprint, it was decided that only one user story with the biggest total value of 20 story points would be done. Alongside with doing this user story, the focus will be on improving the design part to increase the business value of the product. The final user story was split in only four tasks.

4.7 Sprint 6

Sprint six started on the 26th of December and ended on the 30th of December. For the last sprint the main focus was finishing the timeline, refactoring the code to make it “cleaner” and improving the design. The reason behind these latter tasks were that they have the most business value to the product owner, as these changes will directly affect the usability and performance of the system.

Although the final sprint ended by successfully doing the final user story, the velocity dropped to 210 minutes per user story. The reason behind this drop is that more time for the design and refactoring was spent than planned.

It can be questionable why refactoring and design were not added to the product backlog and respectively sprint backlog as new user stories, but as usually 10-15 % of the sprint time is spent for extraordinary tasks in the sprint, it was deemed to put them in that category.

At the end of the sprint, the final and complete demonstration of the software was performed, starting from the “first user story”, registration and login and ending with “the final user story” – presenting the timeline. The conclusion is that every user story which was marked as a MUST was implemented at the end.

5 Future steps

As mentioned in the chapter 2.2.2, this project is an end to its first phase, which was referred to as the initial phase. In this phase, the main goal was to set up the core of the software, but it does not end there. After successfully testing the parts that have been created, the focus is going to be on the upcoming phases:

- Second phase – design and release phase
- Third phase – further development phase
- Fourth phase – a distant future

5.1 Second phase

The second phase will directly continue after the first phase. All of the secondary goals of the first phase will be reviewed. The same way as product backlog, if the goal is considered important, it will be moved on the top of the "backlog". In other case, it will be pushed down. The second phase has main and the secondary goals.

The main goals are:

- Improving users
- Improving businesses
- Improving spots
- Improving meetings

The secondary goals are:

- Payment option for events
- Launching new platforms
- Check in option for the events

The second phase will be done as a part of UCN's Next Step project during a period of 10 weeks, starting on 23/01/2017 and ending on 31/03/2017. The second phase will consist of several stages that are intertwined together:

- 1st stage - Starting a company (23/01/2017-03/02/2017)
- 2nd stage - Improving the solution (23/01/2017-17/02/2017)
- 3rd stage - Hiring new people (06/02/2017-15/02/2017)
- 4th stage - Implementing new functionality (20/02/2017-31/03/2017)

Each stage has more details behind it. The main reasons behind the first one, creating the company, are: being able to release the product and being able to hire people. Reasons are connected, as to be able to hire people it is obligated to have a company. The stage consists of gathering the knowledge about rules and regulations for the IT companies in Denmark, finalizing the business plan for the company, filing the paperwork to open the company, and of course starting working.

Second stage, improving already existing solution, is connected with this project. After the project exam, the next natural step in the software development would be to improve already implemented functionality. To be able to know which functionality to improve the feedback will be collected that will be obtained during the first phase.

The second stage acts as a mini-project of four weeks where the product backlog from the first phase will be reviewed and if some of the items from it could be reused, they will be added to the new product backlog, but prioritized differently. There is already some new functionality that could be added, for example to the: users (friend lists, improved user profile pages), businesses (linking to their websites), spots (equipment list), and meeting (joining/quitting process, comment section, check in option), but the focus will also be on improving the speed and the security of the system.

Third stage, hiring new people is directly connected to the first and fourth stage, where the successful first stage is a precondition that needs to be done so that interns from the different courses could be hired. With hiring new people several mini-projects can be started and initial hiring plan is:

- One graphic design intern who will be responsible for promo material for fairs, social media, digital media and non-digital media in the branding project
- One financial management intern who will be responsible in assisting to make the financial plan for the upcoming periods and who will help in the research for the payments over Internet from abroad (paying by card options on the website)
- Two marketing management interns who will be responsible to set up and maintain the social media and create and maintain the content for the product own media (website, apps, etc.)
- Three multimedia design interns who will be responsible for website design; video promotion, creating other multimedia material
- Four computer science interns who will be responsible for creating apps for other platforms and help improving the product

And finally, the fourth stage, implementing new functionality will focus on the research and implementation of payment options, researching and preparing different platforms to expect the product on them, research possibilities for hardware options for check-in options, and of course, implement secondary goals and continue to improve already existing functionality.

The fourth stage will be the longest part of the second phase and it will last for 6 weeks and also the most complex phase part as it will involve working on several different projects at the same time, while supervised by an ultimate Scrum master. Projects in question are:

- Payment options
- Platform development
- Hardware extensions
- Branding
- Improving the existing product

The fourth stage will also have a goal to prepare the launch of the beta version of the product, as well to collect the information from it and prepare a list of new functionalities that could be added in the third phase.

5.2 Third phase

The third phase is planned to start a couple of weeks after the second phase ends and should finish with the actual release of the product. The initial estimate for the third phase is that it will start at the end of April and end sometime in August 2017, while the goals of the phase will be:

- Collect information from the beta version
- Create new product backlog
- Do comprehensive testing on all platforms
- Prepare everything for the big launch

It is impossible to know what feedback will be gathered from the beta version, but as this will be the first time collecting information from a group of users, it is crucial that all of the suggestions are heard and if possible implemented during this phase. It is hard to predict what users will prefer to be implemented so far ahead, but the first and second goal could be implementations of:

- Statistics
- Ratings
- Automatic confirmations of the events
- Floor plans

The third goal will focus on the manual testing to see if everything is looking as it should look, while the fourth goal will be to prepare all of the needed prerequisites for launching a website.

5.3 Fourth phase

Although the fourth phase is a long time away, it is important to stress out that the product needs to continue to evolve and the maintenance and new implementations will continue on. It is important to make sure that the feedback is continuously collected from the users. Small projects to improve the functionality are being done and that the new functionality is added to the system. In addition, parts that will be considered useless will be removed.

It is impossible to predict how will this phase look at this moment, but the only real hope is that it will be possible to actually reach this phase at the end of 2017.

6 Conclusion

The project was challenging as the idea was developed while both of the team members were on their internships in different countries. Developing an idea for the project while being thousands of kilometres apart made the discussions difficult due to the limitation of using basic tools like whiteboard and drawings, and sharing the knowledge. Nevertheless, different tools were used to communicate (Trello and Google Drive for exchanging documents, Skype and Google Hangouts for discussions) and the idea was steadily developed. When team members met in November 2016 and sort out all of the potential risks that could endanger the creation of the product, the final idea was formulated. Unfortunately, working at the same spot did not last for long, as a couple of days later, members split.

This brought an interesting twist to the fact that for this final project agile methods will be used in a small team working on long distance. Following SCRUM workflow (daily meetings with weekly review and retrospective) helped to maintain the momentum of the project. The biggest issue was not the dislocation of the members, it was the overconfidence in the ability to absorb new technologies and follow the procedures. For example, from the procedural point of view, the challenge was to keep daily meetings short and on the topic. By continuously learning during the process, working as a team by dividing and explaining the completed tasks, the work process became better and better to culminate with a perfect sprint at the end. That brought out the outcome that the best combination of methodologies and technologies were used in this project. Regardless, the initial sprints pointed out that better test strategy was needed and additional time for researching technologies is required.

It is important to state that during the sprints, the system was being designed according to the requirements agreed upon at the beginning of the project. Even so, there was a positive attitude towards both new ideas and changes, in case they would lead to make a better solution at the end. All of the activities during the project had both positive and negative aspects. Positive aspects of working in a small team were easy distribution of the tasks and fast feedback inside the team. In addition, the most interesting was combing members with a different background in knowledge who ended up with a lot of new learning points for each other. On the other hand, there were also some negative aspects of working in a small team. For instance, the fear that someone would do less than expected or that some parts of the product would not be created due to the lack of the expertise. Both cases could endanger the project, which was quite a risk as there are only two members. All in all, methodologies used were a good choice, but the team size should be at least four to five members. This would reduce any size related risks and ensure that the quality of the product is high.

From the technical view, this project was a great learning point. It was the chance to learn a number of new things, techniques and practises, that together widely extended the quality of the product. It also provided a great opportunity, to apply the knowledge gained during the internships. Overall, the project was a great practise of creating the product, with a focus on all - planning, implementation, testing and development.

To conclude, it is important to say that the final product at the end is a workable software solution with the primary function of time and event management. Users are divided in two categories: businesses and private user. This was done so that the business owners can manage their business, locations and spots, while the private users can create events on that spots. The solution at the end is simple and user-friendly, so users are able to create and view the events which they prefer. Although this project finishes today, the product is still going to be developed in the future and the real challenge of releasing the product to the market is just starting.

7 References

[1] Alistair Cockburn (2007): Agile Software Development, second edition, published by Addison-Wesley

ISBN: 978-032-148275-4

[2] Ian Sommerville (2016): Software Engineering (10th Edition), published by Pearson

ISBN: 978-1-292-09613-1

[3] Ilan Goldstein (2014): Scrum shortcuts without cutting corners: agile tactics, tools & tips, published by Pearson Education Inc.

ISBN: 978-0321-82236-9

[4] Ken Schwaber (2004): Agile Project Management with Scrum, published by Microsoft Press

ISBN: 0-7356-1993-X

[5] Klaus Nielsen (2013): I Am Agile – Knowledge That Sets You Apart, first edition, published by Nyt Teknisk Forlag

ISBN: 978-87-581-2803-1

[6] Martin Fowler (2003): Patterns of Enterprise Application Architecture, published by Addison Wesley

ISBN: 978-0321127426

[7] Mike Cohn (2010): Succeeding with Agile: Software Development Using Scrum, published by Addison-Wesley

ISBN: 978-0-321-57936-2

[8] Mike Cohn (2004): User Stories Applied: For Agile Software Development, first edition, published by Addison-Wesley

ISBN: 978-0321205681

[9] Pieter Jongerius (2012): Get Agile: Scrum for UX, design and development, published by BIS publishers

ISBN: 978-90-6369-302-2

[10] Ramez Elmasri (2011): Fundamentals of Database Systems, published by Addison Wesley Pearson

ISBN: 978-0136086208

[11] Material from the course folders on Canvas

[12] Different websites (information retrieved in November and December 2016):

- https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx
- https://www.psychologytoday.com/blog/wired-success/201204/why-meetings-kill-productivity
- http://www.agile-process.org/

- http://c2.com/cgi/wiki?ExtremeProgramming
- https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch
- https://www.merriam-webster.com/dictionary/event
- https://blogs.msdn.microsoft.com/saraford/2004/10/28/developing-a-test-specification/
- http://istqbexamcertification.com/what-is-use-case-testing-in-software-testing/
- https://msdn.microsoft.com/en-us/library/ee658117.aspx
- https://www.quora.com/What-are-the-advantages-of-ASP-Net-MVC-over-ASP-Net-Web-Forms
- https://addyosmani.com/blog/understanding-mvvm-a-guide-for-javascript-developers/
- https://www.tutorialspoint.com/knockoutjs/knockoutjs_mvvm_framework.htm
- https://msdn.microsoft.com/en-us/library/ff649977.aspx
- https://www.youtube.com/watch?v=IKD2-MAkXyQ
- http://tutorials.jenkov.com/dependency-injection/dependency-injection-benefits.html
- https://www.youtube.com/watch?v=7Pj5kAhVBlg&t=1163s
- https://www.youtube.com/watch?v=t_frFWqrnJE
- http://www.dofactory.com/net/command-design-pattern
- https://msdn.microsoft.com/en-us/library/ff921087.aspx
- https://www.codeproject.com/articles/560798/asp-net-mvc-controller-dependency-injection-for-be
- https://developer.chrome.com/apps/app_frameworks
- http://neo4j.com/blog/why-nosql-databases/
- http://www.vtagion.com/scalability-scale-up-scale-out-care/
- https://www.youtube.com/watch?v=XPqrY7YEs0A
- http://robertgreiner.com/2014/08/cap-theorem-revisited/
- http://www.c-sharpcorner.com/blogs/sql-server-acid-properties1
- http://stackoverflow.com/questions/3342497/explanation-of-base-terminology
- http://neo4j.com/blog/acid-vs-base-consistency-models-explained/
- http://www.jamesserra.com/archive/2015/07/what-is-polyglot-persistence/
- http://www.zdnet.com/article/rdbms-vs-nosql-how-do-you-pick/
- https://www.youtube.com/watch?v=X1yNTq_R2JA
- https://docs.mongodb.com/manual/core/data-modeling-introduction/
- https://www.asp.net/mvc/overview/older-versions-1/views/asp-net-mvc-views-overview-cs
- https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview
- https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor
- https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial
  https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/
- https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/
- http://www.azquotes.com/author/38434-Jeffrey_Veen
- http://getbootstrap.com/css
- http://knockoutjs.com/documentation/observables.html

- http://knockoutjs.com/documentation/introduction.html
- https://www.codeproject.com/Articles/826757/Understanding-OWIN-and-Katana
- http://www.w3schools.com/jquery/jquery_ajax_intro.asp
- http://www.w3schools.com/xml/ajax_intro.asp
- https://www.tutorialspoint.com/ajax/what_is_ajax.htm
- https://www.thoughtworks.com/insights/blog/introducing-software-testing-cupcake-anti-pattern
- https://watirmelon.blog/tag/software-testing-pyramid/
- https://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx
- http://stackoverflow.com/questions/1554018/unit-test-nunit-or-visual-studio
- https://www.nunit.org/
- https://msdn.microsoft.com/en-us/library/hh694602.aspx
- https://www.jetbrains.com/resharper/features/unit_testing.html
- https://msdn.microsoft.com/en-us//library/bb383977.aspx
- http://www.seguetech.com/the-benefits-of-unit-testing/
- http://stackoverflow.com/questions/7321407/what-is-sut-and-where-did-it-come-from
- https://smartbear.com/learn/automated-testing/
- http://www.guru99.com/introduction-webdriver-comparison-selenium-rc.html
- http://www.guru99.com/manual-testing.html
- https://www.tutorialspoint.com/software_testing_dictionary/manual_testing.htm
- https://www.youtube.com/watch?v=gwIS9cZlrhk
- http://williamdurand.fr/2013/07/30/from-stupid-to-solid-code
- http://searchdatacenter.techtarget.com/definition/back-end
- http://blog.digitaltutors.com/whats-difference-front-end-back-end/

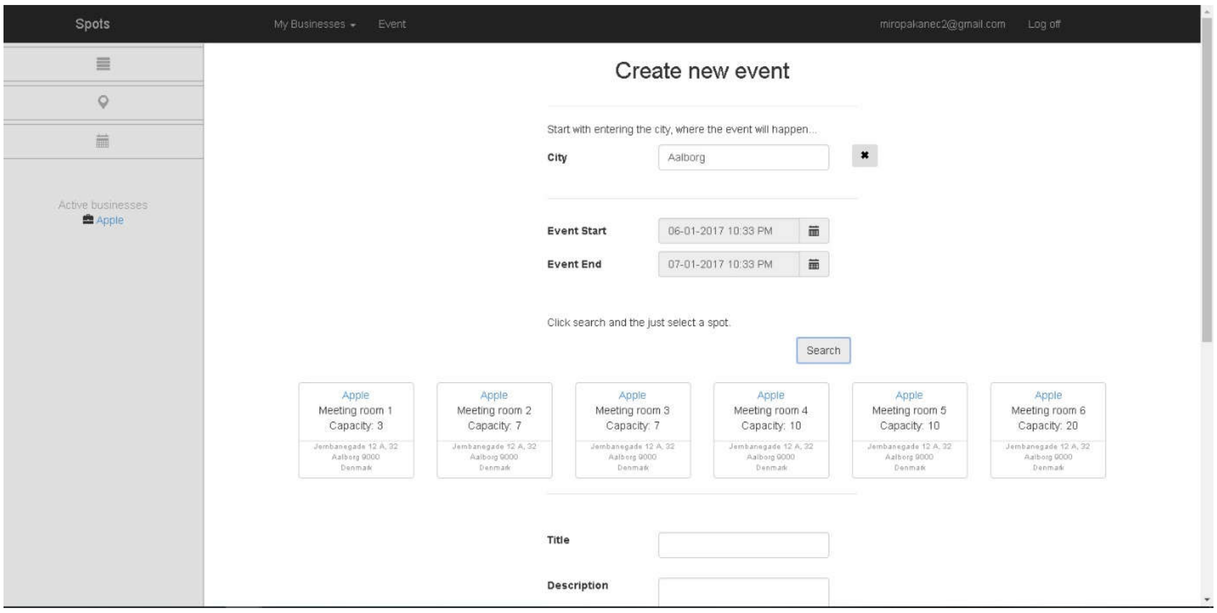
7.1 List of abbreviations and acronyms used in the report

ACID - acronym from Atomicity, Consistency, Isolation, and Durability
AJAX - abbreviation for Asynchronous JavaScript and XML
BASE - acronym from Basically Available, Soft state, Eventual consistency
CAP - acronym from Consistency, Availability and Partition tolerance
CEO - abbreviation for chief executive officer
CRUD - acronym from Create, Read, Update, Delete
CSS - abbreviation for Cascade Style Sheet
DAL - abbreviation for Database Access Layer
DB - abbreviation for Database
DI - abbreviation for Dependency Injection
DIP - abbreviation for Dependency inversion principle
GUI - abbreviation for Graphical User Interface
HTML - abbreviation for Hyper Text Markup Language
ID - short for Identificator
IIS - abbreviation for Internet Information Services
INVEST - acronym from Independent, Negotiable, Valuable, Estimable, Small, Testable
IoC - abbreviation for Inversion of Control
ISP - abbreviation for Interface Segregation Principle
JSON - abbreviation for JavaScript Object Notation
LSP - abbreviation for Liskov substitution principle
MVC - abbreviation for Model-View-Controller
MVVM - abbreviation for Model-View-ViewModel
NoSQL - abbreviation for Not only Structured Query Language
OCP - abbreviation for Open/closed principle
OWIN - abbreviation for Open Web Interface for .Net
RAM - abbreviation for Random Access Memory
RDBMS - abbreviation for relational database management system
REST - abbreviation for representational state transfer
SoC - abbreviation for Separation of Concerns
SEO - abbreviation for Search Engine Optimization
SOLID - acronym from SRP, OCP, LSP, ISP, DIP
SRP - abbreviation for Single responsibility principle
SQL - abbreviation for Structured Query Language
SUT - abbreviation for System Under Test
TDD - abbreviation for Test Driven Development
UI - abbreviation for User Interface
URL - abbreviation for Uniform Resource Locator
VSTS - abbreviation for Virtual Studio Team Services
WIP - abbreviation for Work In Progress
XML - abbreviation for Extensible Markup Language
XP - abbreviation for Extreme Programming

7.2 Figures inserted in the main part

7.3 Figures inserted in the appendix

8 Appendix


Spots My Businesses Event miropakanec2@gmail.com Log off

Active businesses
Apple

Create new event

Start with entering the city, where the event will happen...

City: Aalborg

Event Start: 06-01-2017 10:33 PM

Event End: 07-01-2017 10:33 PM

Click search and the just select a spot.

Search

Apple
Meeting room 1
Capacity: 3
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 2
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 3
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 4
Capacity: 10
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

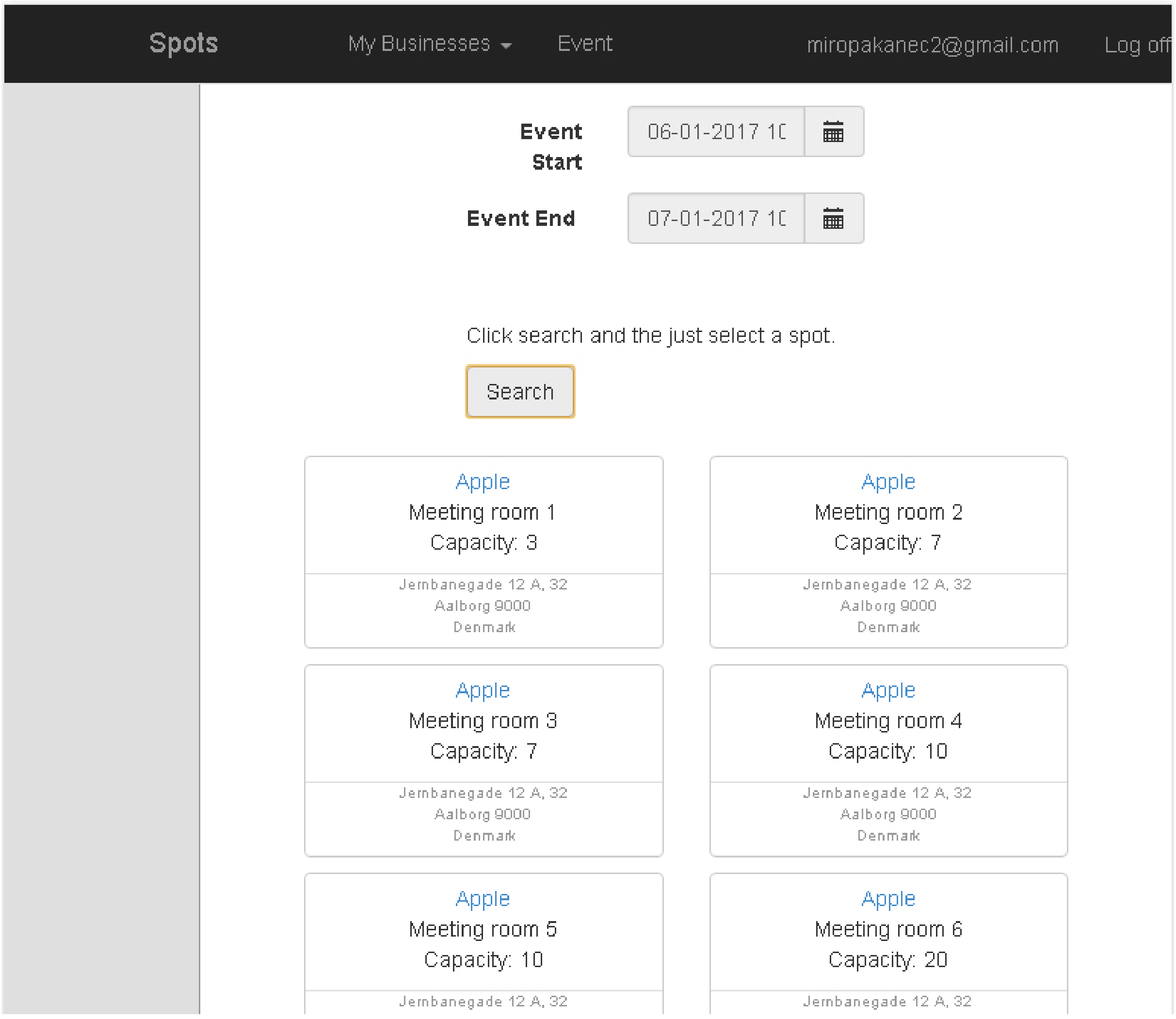
Apple
Meeting room 5
Capacity: 10
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 6
Capacity: 20
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Title

Description

Figure 62 Web page opened on a large desktop device


Spots My Businesses Event miropakanec2@gmail.com Log off

Event Start: 06-01-2017 10

Event End: 07-01-2017 10

Click search and the just select a spot.

Search

Apple
Meeting room 1
Capacity: 3
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 2
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

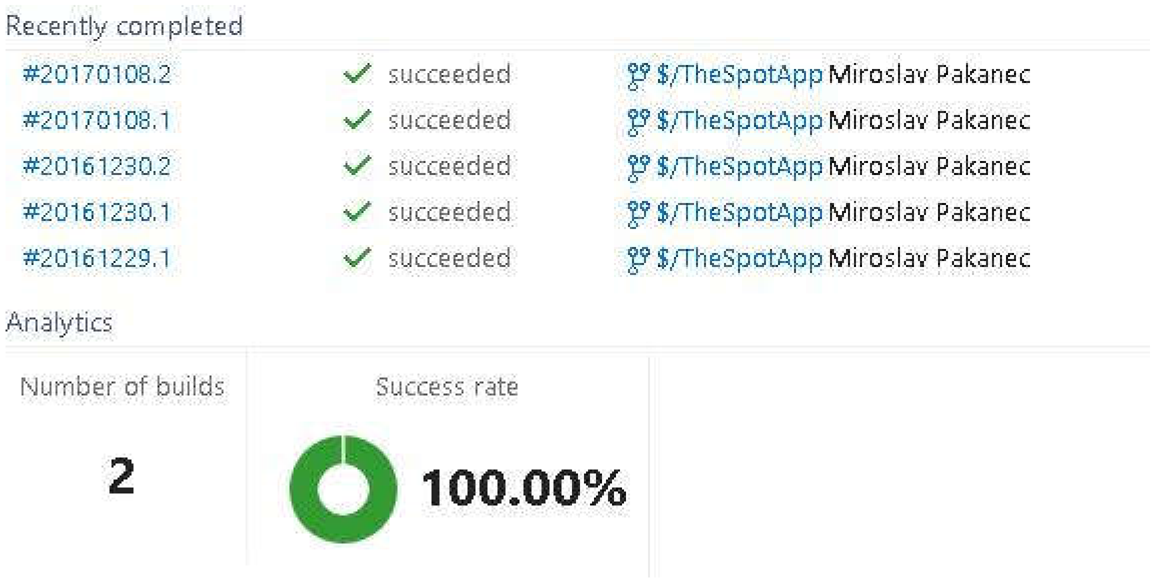
Apple
Meeting room 3
Capacity: 7
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 4
Capacity: 10
Jernbanegade 12 A, 32
Aalborg 9000
Denmark

Apple
Meeting room 5
Capacity: 10
Jernbanegade 12 A, 32

Apple
Meeting room 6
Capacity: 20
Jernbanegade 12 A, 32

Figure 63 Web page opened on iPad (small device)



Recently completed

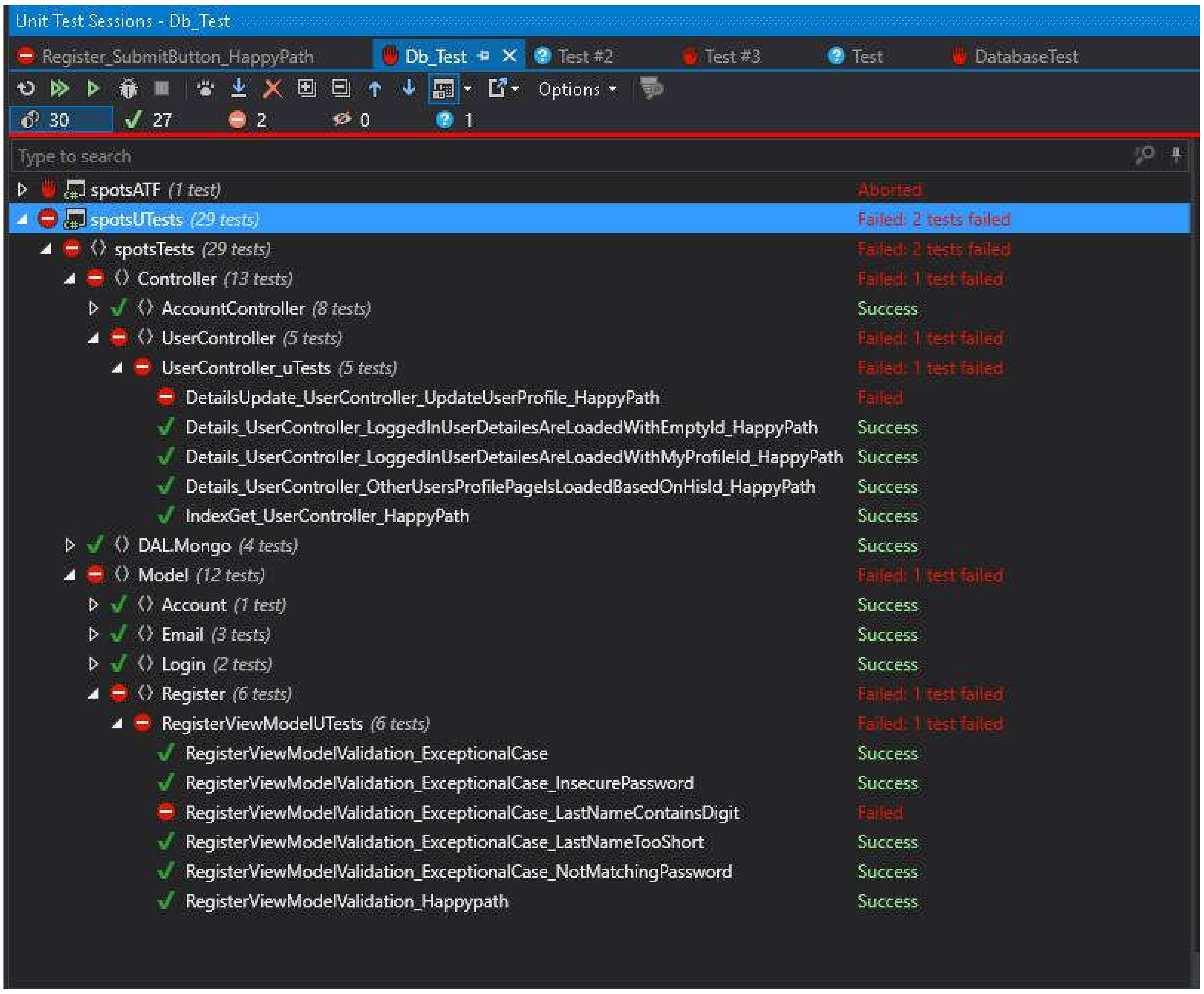
#20170108.2	✓ succeeded	$/TheSpotApp Miroslav Pakanec
#20170108.1	✓ succeeded	$/TheSpotApp Miroslav Pakanec
#20161230.2	✓ succeeded	$/TheSpotApp Miroslav Pakanec
#20161230.1	✓ succeeded	$/TheSpotApp Miroslav Pakanec
#20161229.1	✓ succeeded	$/TheSpotApp Miroslav Pakanec

Analytics

Number of builds 2

Success rate 100.00%

Figure 64 Example of continuous integration



```
spotsATF (1 test)                                                          Aborted
spotsUTests (29 tests)                                                     Failed: 2 tests failed
  spotsTests (29 tests)                                                    Failed: 2 tests failed
    Controller (13 tests)                                                  Failed: 1 test failed
      AccountController (8 tests)                                          Success
      UserController (5 tests)                                             Failed: 1 test failed
        UserController_uTests (5 tests)                                    Failed: 1 test failed
          DetailsUpdate_UserController_UpdateUserProfile_HappyPath         Failed
          Details_UserController_LoggedInUserDetailesAreLoadedWithEmptyId_HappyPath  Success
          Details_UserController_LoggedInUserDetailesAreLoadedWithMyProfileId_HappyPath  Success
          Details_UserController_OtherUsersProfilePageIsLoadedBasedOnHisId_HappyPath  Success
          IndexGet_UserController_HappyPath                                Success
    DAL.Mongo (4 tests)                                                    Success
    Model (12 tests)                                                       Failed: 1 test failed
      Account (1 test)                                                     Success
      Email (3 tests)                                                      Success
      Login (2 tests)                                                      Success
      Register (6 tests)                                                   Failed: 1 test failed
        RegisterViewModelUTests (6 tests)                                  Failed: 1 test failed
          RegisterViewModelValidation_ExceptionalCase                      Success
          RegisterViewModelValidation_ExceptionalCase_InsecurePassword     Success
          RegisterViewModelValidation_ExceptionalCase_LastNameContainsDigit  Failed
          RegisterViewModelValidation_ExceptionalCase_LastNameTooShort     Success
          RegisterViewModelValidation_ExceptionalCase_NotMatchingPassword  Success
          RegisterViewModelValidation_Happypath                            Success
```

Figure 65 Unit testing

8.1 Manual test specification

ID: TC1 (Test Case1)

Name: User registration

Description:

Test for a successful registration of the user. User inputs information to the input boxes. When all of the necessary boxes have value inputted, user clicks on the submit button.

Browser: Google Chrome

URL:

https://www.spots.com/Account/Register

https:// https://localhost:44371/Account/Register

Data:

Email: {testmail@gmail.com}

First name: {John}

Last name: {Smith}

Password: {Password0.}

Password confirm: {Password0.}

Test Steps:

#	Action	Verification
1	Go to {URL}	Browser's URL is equal to {URL}, Page header title is equal to {Welcome - Spots}, Page title is equal to {Register}, Page subtitle is equal to {Create a new account}
2	Enter {Email} to the text box with label Email	Success
3	Enter {First name} to the text box with label First name	Success
4	Enter {Last name} to the text box with label Last name	Success
5	Enter {Password} to the text box with label Password	Success
6	Enter {Password confirm} to the text box with label Confirm password	Success
8	Perform a click on the button with value Register	Browser's URL is equal to {URL}, Message to confirm Email is visible, Link to the Login page is provided

8.2 Use case testing (Example 1):

Create location

1. Description

Business admin has to enter his email and password to login to the system. After logging in, admin has to click the my business menu and choose my business. If the admin has different business, he will have to choose for which business admin would like to create location. Upon clicking the desired business, admin has to choose button create new location. Admin then inputs the address, zip code, city and country. After confirming the information by clicking the create location button, entered information are validated and either accepted or rejected. In case of an acceptance, location will be created with an unique ID.

2. Triggers

User clicks on the create location button.

3. Actors

Business admin

4. Preconditions

- User is logged in
- Business exists
- Address is entered
- Zip code is entered.
- City name is entered
- Country is entered

5. Goals

- A location is created so that new spots could be added and "rented" out.

6. Failed conclusions

- Address field is empty.
- Zip code field is empty.
- City name field is empty.
- Country field is empty.

7. Extensions

- Drop down menu for countries

8. Steps of execution

I. User enters the web page.

I.A. Web page is down - Error message is displayed

II. User inputs the login information

II.A Username is invalid; error message is displayed, input field has red outline

II.B. Password is invalid; error message is displayed, input field has red outline

III. User selects the business they want to edit

IV. User chooses create new location

V. User inputs the address

V.A. Address field is empty; error message is displayed, input field has red outline

VI. User inputs the zip code

VI.A. Zip code field is empty; error message is displayed, input field has red outline

VII. User inputs the city

VII.A. City field is empty; error message is displayed, input field has red outline

VIII. User inputs the country

VIII.A. Country field is empty; error message is displayed, input field has red outline

IX. User clicks button “Create location”

X. Location is created.

8.3 Use case testing (example 2)

Log in

1. Description

User goes to the homepage. User inputs the email and password. If the information is correct, user is redirected to the secure homepage view of his profile.

2. Triggers

User clicks on the Login button

3. Actors

Registered user

4. Preconditions

- Registration successfully done

5. Goals

- Be able to access the website as a registered user so that the features of the website can be used

6. Failed conclusions

- Email field is not filled in
- Email doesn't exist in the database
- Password field is not filled in
- Password doesn't match the password in the database

7. Extensions

- N/A

8. Steps of execution

I. User goes to the homepage and clicks on the login

II. User types in his email and password and clicks login button

II.A Email doesn’t exist in the database, error message is displayed, both fields have red outline

II.B Email field is empty, error message is displayed, input field has red outline

II.C Password field is empty, error message is displayed, input field has red outline

II.D Password field doesn’t match the password in the database, error message is displayed, both fields have red outline

III User is successfully logged in

8.4 Product backlog

ID	Story name	As	I want to...	so that...	Acceptance criteria	MoSCoW	Priority	Estimate	Comments
1	Register user	Unknown user	Register to the system by entering email, password, and other required personal information(name, surname)	I am registered as a user in the system and I can access the websites features by logging in.	Check if email is valid (has an "@" and ".") Check if email is already used in our database. Check if all fields are filled in. Check whether the password is at least 8 characters. Check whether the password contains uppercase, lowercase, digit and special character.	M	1	2	
2	Log in	Registered user	Be able to access the website as a registered user by entering a email and a password	I can use other features of the website (such as creating the event or check my future appointments)	Check if email is in the database Check if password is in the database and is matching the email If all above is correct the system should redirect to the home page	M	2	1	
31	Log in via FB or Google +	Registered user	Be able to access the website as a registered user by logging in via Facebook or gmail.	I can have the same functionality as normal log in	User can see everything as he used the normal log in procedure	C	3	1	

15	Register via FB or google +	Registered user	Connect with FB or Google +	Registration can be done by clicking one of the social media buttons	User already has a FB or Google+ account;	C	4	1	
32	Passwor d forget	Registered user	Sent an email with a reset password link	user can reset the password	Email has to be registered to the system / entering email registered to system and clicking request new passford will trigger system to send an email with instructions how to reset the password / If wrong email is enterred an error message is displayed	S	5	2	
3	Log out	Registered user	Log out from the website	My credentials are not missused	If user is logged in, logged out button is visible and clickable / After clicking log out, session ends and user is redirect to login page	M	6	0. 5	
4	Profile confirma tion	System	sent an email to every user who successfully register	they can confirm that the email is correct and the account becomes valid	If registration is successful, a confirmation email is sent. After user clicks the link, user status is confirmed as valid	S	7	5	
38	Automati c deletion of the profile	System	System automatically deletes profile that are not confirmed	to reduce the database load	Non verified accounts are deleted after XX days from the database / E-mail is sent that account has been deleted?	S	8	2	Switched to W and dropped to the bottom of the backlog

5	Edit profile	Registered user	Be able to change or update my profile attributes such as: personal information (name, surname, email, address, phone number), and profile picture	User has better control over the account and has a possibility to personalize it. User can control his calendar benefits	All of the fields except username and email are editable / Extra fields (age and middle name can be added) Updated fields have to be valid	M	9	8	
37	Delete profile	Registered user	I would like to delete my profile	I do not wanna my information shared any more	Prompt are you sure commes up and an email with 4-digit deletion code is sent / Password and code has to be entered / After confirming delete; E-mail is sent that account has been deleted	C	10	2	
42	View other user profile	Registered user	I would like to check other people profiles	I can see what are they attending	Opening a page where all of the visible information about that user is shown	S	11	3	
39	Create business	Registered user	Add my company inormation	I can manage my company information in the system	All fields are entere with next conditions: bussines name field contains ad least 3 characters, tax number field has to be exactly 10 characters, and the phone number field has to be between 10 and 30 characters	M	12	5	

6	Creating location	Business admin	Be able to create location with all of the information needed (Address, Zip code, City, Country)	I can create spots company locations and see employees connected to these locations	Name has to be unique Check if all fields are filled in.	M	13	8	
9	Creating a spot	Business admin	Be able to create a spot with all of the information needed (Name, Capacity, Visibility)	I can create spots that are available on the location	Check if all fields are filled in.	M	14	8	
11	Search a spot availability by city and date	Registered user	Search spots in a certain city	User is able to search for a spot that interests him	All of the public spots in the city are shown Fields are valid If there is no valid information, error message is shown	M	15	8	
41	Comparing availability (spots)	System	System compares available slots for the slots	the event (event) can be made	All available slots are shown	M	16	8	Duplicate of the Search a spot (ID 11)
19	Create an event	Registered user	I would like to create the event (preselected time period, availability, visibility, invitations)	so I can meet someone	Spot is choosen / The slot is available / All fields are checked	M	17	13	
43	View event page	Registered user	I would like to find more information about a certain event	I can see all relevant information about it	View information about requested event - event name, description of the event, business name, spot information, address	S	18	8	
40	Comparing availability (users)	System	System compares available	the event (event) can be made	All available slots are shown	S	19	3	

			slots for the user						
20	Edit the event	Registered user	I would like to change the information about the event	to change wrong information	Spot is choosen / The slot is available / All fields are checked	S	20	5	
10	Edit a spot	Business admin	Be able to change information (Name, Capacity, Visibility)	the information about the spot is up to date	Name has to be unique Check if actually something is changed (cannot save if there are no changes) Check if all fields are filled in.	S	21	3	
27	Calendar (spots)	Registered user	Be able to see the calendar for every spot	I can click on the calendar and check the events and free timeslots	All available slots are shown green / Are taken slots are shown yellow	S	22		
26	Calendar (users)	Registered user	Be able to see the calendar for every user	I can click on the calendar and check the events and free timeslots	All available slots are shown green / Are taken slots are shown yellow	S	23		
12	Delete a spot	Business admin	delete a spot that is not active anymore	the events in non-existing spot are created	Spot cannot have active events? Should instead delete better option be "suspend"	S	24		
13	Delete a location	Business admin	delete a location that is not active anymore	no new spots are created on the location	Spot cannot have active events?	S	25		
21	Delete the event	Registered user	I would like to cancel the event	the event is canceled	Spot is choosen / the slot for the event is available after deletion	S	26		
17	Add an employe e	Business admin	I would like to add a person to my company list	so I can invite that employee to company events and oversee his public events	Email is inserted / If email exists it is displayed with a box "add as employee" /	S	27		

					after box is clicked notification is sent to that user				
18	Remove employe e	Business admin	I would like to remove the person from my company list	as the employee is not part of the company any more	Email is inserted / If email exists it is displayed with a box "remove as employee" / after box is clicked notification is sent to that user	S	28		
16	Add a friend	Registered user	I would like to add a person to my friend list	so I can invite that friend for cool events	Email is inserted / If email exists it is displayed with a box "add as friend" / after box is clicked notification is sent to that user	C	29		Alternative - allow people to create groups and invite people to them (if the purpose is event invitation). I would call it 'connections ' if the purpose is to access profile information.
7	Confirmi ng location	System admin	Confirm that the location really exists and that the business is at that spot	the information is validated for users to have security it is not a hoax	Status of location is changed to confirmed	W	36	?	
8	Search location by city	Registered user	Search locations in a certain city	User is able to search for a location that interests him	All of the public locations in the city are shown Fields are valid If there is no valid information, error message is shown	W	37	?	
14	Edit location	Business admin	Change the address of the location in case that the office is relocated	events are not created in non-existing locations	New location has te be re-confirmed? There are no active spots.	C	30		

22	Send invitation for the event (friends)	Registered user	I would like to invite friends for the event	I can hang out with my friends	Invitiation box is checked in / notifications are sent to the "friend" list	C	31		
23	Send invitation for the event (employees)	Registered user	I would like to invite employees for the event	I can have the event with my colleagues	Invitiation box is checked in / notifications are sent to the "employee" list	C	32		
24	List events (for a spot)	Registered user	I would like to check the list of the events	I can see for what is the spot used	A list is created for the spot / private events are marked as busy (creator visible) / If there are no events, feedback that there are no events is displayed	C	33		
25	List events (for a user)	Registered user	I would like to check the list of my events	I can see my appointments and review my past events	A list is created for the user / If there are no events, feedback that there are no events is displayed	C	34		
36	Visible profile	Registered user	I would like to choose should my profile would be visible	To keep my information private	Option visible is turned off	C	35		
29	Additional fields in spots (aka equipment)	Business Admin	Add aditional information about certain location	I can track what inventory is at certain spot	All field are editable / Field needs to have an input	W	38		
28	Additional fields in profile? (aka preferences)	Registered user	Add aditional information about my wishes	I can show my preferences to other users	All field are editable / Field needs to have an input	W	39		
30	Captcha	Registered user	Increase the security of the website by implementing captcha	we prevent signups of spambots	Captcha task is completed successfully and redirect to a correct page starts / If captcha is not completed	W	40		

					correctly error message is displayed				
33	Log out timer	System	automatically log out from the website	I do not have to think about it	User is automatically logged out after 300 seconds and redirected to the home page	W	41		
34	Log out confirma tion	Registered user	have a pop up window (or notification) when I am logging out	I do not logged out by accident	Pop up window pop out after clicking the log out button (or notification pop ups in the notification bar)	W	42		
35	Edit profile confirma tion	Registered user	Confirmation prompt when I edit information on my profile	I do not save something I do not want	Confirmation label when user clicks update button on the profile	W	43		