MORE FUN, FEWER RISKS: DEVELOPMENT OF A GAMIFIED WEB APP FOR RISK MANAGEMENT

STUDIENARBEIT

of the course computer science at the

Cooperative State University (Duale Hochschule) Baden-Württemberg Karlsruhe

from

Inga Batton, Moritz Horch, Nils Krehl

Submission date:

18. May 2020

Timeframe:	TODO: XX Wochen
Matriculation number, course:	XXX, TINF17B2
Company:	dmTECH GmbH, Karlsruhe
Supervisor:	Ph.D., Prof. Kay Margarethe Berkling

Abstract

Erklärung

(gemäß §5(3) der "Studien- und Prüfungsordnung DHBW Technik" vom 29.09.2017)

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema: "More Fun, Fewer Risks: Development of a Gamified Web App for Risk Management" selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Unterschrift

Contents

List of Figures

List of Tables

List of listings

List of abbreviations

Die nach Ansicht des Autors wichtigsten Abkürzungen:

API	Application Programming Interfaces
HTTP	Hypertext Transfer Protocol
MVC	Model View Controller
REST	Representational State Transfer
SQL	Structured Query Lanuage
UC	Use Case
JSX	JavaScript XML
JPA	Java Persistence API

Glossary

REST todo

SQL todo

1. Introduction

context, motivation, aims, purpose, ..

2. Theoretical background

This chapter introduces the theory of the domain risk management (chapter 2.1), gamification (chapter 2.2) and Progressive Web Apps (chapter 2.3).

2.1. Risk Management

There are plenty of reasons for projects to fail and frequently even large companies and organizations experience costly failures of big projects [1]. Projects are often defined as failed, when they cannot meet time or budget constraints or do not fulfill the pre-defined requirements. However, this definition is not useful in every context [2]. IT projects often follow agile management techniques allowing for changes in the pre-defined scopes [3]. To allow for a wider understanding of what IT-project failure means Lyytien and Hirchheim group such failures into four different categories [4]:

- Correspondence: Not meeting the pre-defined objectives
- Process: Exceeding time or budget restrains
- Interaction: Lack of end-user engagement
- Expectation: Inability to meet stakeholder's expectations

Analogous to the variety of ways in which a project can be defined as being unsuccessful there are many reasons which can lead to any such failure. Plenty research has been done to investigate the causes of project failure [5]. Events that lead to project failure can be understood

as risks. Islam [6] provides the following definition for risks in an IT context:
"Software risk, is defined as, the possibilities of suffering a loss such as budget or schedule over-runs, customer dissatisfaction, poor quality and passive customer involvement due to an undesirable event and its consequences during the life cycle of the project."

Many such risk factors have been identified in the literature by now. The following risks are an excerpt from lists collect via literature reviews by Whitney and Daniels [7] and Tesch et al [8].

Whitney and Daniels

- Lack of top management commitment to the project
- Failure to gain user commitment
- Misunderstanding the requirements
- Lack of adequate user involvement
- Lack of required knowledge/skills in the project personnel
- Changing scope/objectives
- Introduction of new technology
- Failure to manage end user expectations
- Insufficient/inappropriate staffing
- Poor project management
- Excessive schedule pressure
- Lack of technical specifications

Tech et al.

- Personnel shortfall and straining computer science abilities
- Unrealistic schedules and budgets
- System functionality
- Requirements management
- Resource usage and performance
- Personnel management
- Unrealistic project goals and objectives
- Poor project team composition
- Project management and control problems
- Problematic technology base/ infrastructure

To counter such risk factors risk management practices can be integrated into the overall project management. Risk management serves to identify risks, analyze them and to address them to minimize the damage these risks could do to project [8].

Risk management is a process that should be initiated early in the project lifecycle to enable proactive handling of threats [6]. In general the cycle of risk management involves the following steps: Identification, Analysis, Response/Treatment and monitoring and control [6], [8], [9]. The steps should be undertaken at the beginning of the project and updated whenever changes occur. There are different and more detailed variations [8], however for the purpose of this paper the general model will be assessed in more detail.

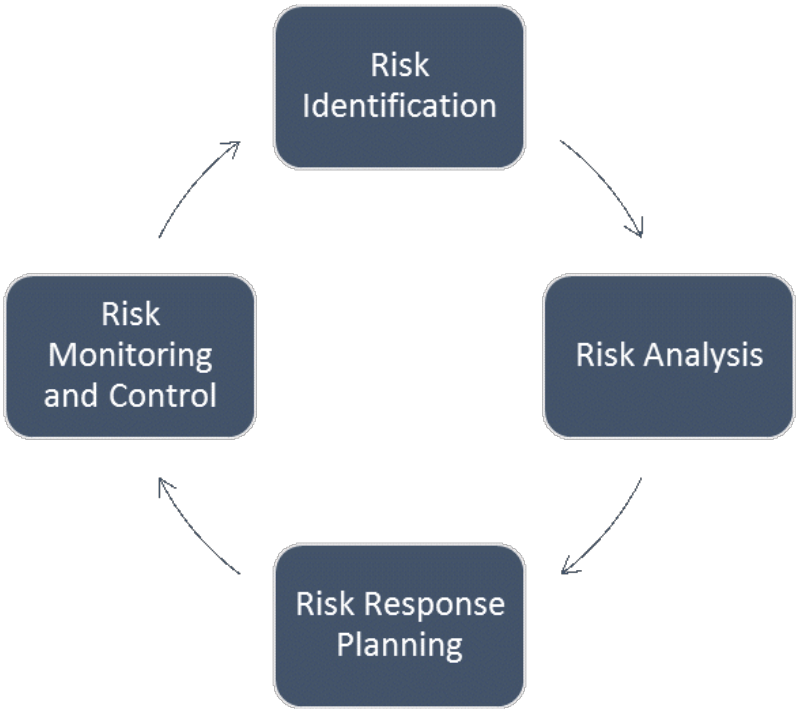

FIG. 2.1.: *Risk Management Circle*
[own representation]

The different steps of the process serve different purposes and have different side effects. Risk identification helps to create awareness and to initiate action in general. It is also a phase during which the project team and stakeholders can share their concerns regarding the project and clarify their expectations to form a common view [9].

To actually perform the identification different techniques can be used. Two commonly used ones are the checklist and brainstorming [6], [9]. Checklists rely on past experience to identify known risk factory which are applicable to the project at hand. Another variant of procedure is to use a questionnaire instead which covers characteristics of the project to find specifically corresponding risks. Brainstorming is ideally done together with project stakeholders to gain different perspectives. Risk identification techniques are not mutually exclusive and combinations may result in more comprehensive results [6].

Risk analysis serves to create acceptance of the previously identified risks as well as to indicate their impact [9]. During this phase the likelihood of risk occurrence and the impact are estimated. This can be done in a qualitative manner by assigning ordinal values for both dimensions. The scales for likelihood can for example go from rare to almost certain. Impact can be described from low to catastrophic. Such estimates are subjective and my produce unclear results however trying to apply quantitative techniques can be unreliable as well since estimations based on past data may not be applicable anymore in a rapidly changing environment such as IT [6].

Risk response planning serves to reduce threats and to enhance opportunities [8]. Dealing with risks can be approached in different manners. Measurements can be defined to either avoid or prevent the risks or to deal with the impact should the risk occur. Another alternative can be to simply accept the risks or to outsource the risks [6]. Another practice used is to assign risk owners to establish clear responsibility for later control efforts [10].

Risk control serves to initiate action on the monitored risks and to direct action [6]. Monitoring the risks enables responding to changes via new cycles of the risk management process as well as triggering the measurements defined during the previous phase if necessary [8]. Techniques employed during this phase can be risk audits, trend analysis or regular status meetings [6].

TBL. 2.1.: *Summary effects and practices of the different phases of risk management*

Phase	Effects on the project	Risk management practices
Risk identification	Initiate action Create awareness Stakeholder communication Common view on the project Clarify expectations	Checklists Brainstorming
Risk analysis	Create acceptance Identify risk impact Estimate risk occurrence likelihood	Qualitative estimation Quantitative analysis
Risk response planning	Reduce threats Enhance opportunities	Contingency plan Risk avoidance or prevention Risk acceptance Outsourcing Risk ownership
Risk monitoring and controlling	Initiate and direct actions React to changes	Risk audit Trend analysis Status meeting

Different studies have been undertaken to evaluate the effect of risk management on project success [2], [9], [11], [12]. The results vary regarding which part of risk management or which tools and techniques contribute to projects success but some sort of positive impact is reported from all of them.

However, in practice risk management is often neglected [11]. Even if there is initial investment into risk management during the planning phase of a project there are tendencies to let efforts slide once the project is running and time pressure picks up [10]. Another attitude towards risk management that has been observed is to view it as additional work and cost which can hinder the adoption of any such practices [8].

For the development of a software tool to facilitate risk management in IT projects the following conclusions can be drawn from the above presented:

- The tool should provide functionalities for all four phases of risk management
- To address the project stakeholders the tool should facilitate risk management practice and provide transparency
- The tool should be easy to use as to speed up risk management processes
- The tool should be engaging to prevent negligence in times of pressure

Further aspects to pay attention to when developing a risk management tool as presented by Keshlaf and Hashim [13] are:

- Documentation and building on that graphical preparation and usage of the documented risks
- User assistance for the risk estimation process
- Versatile applicability
- Comprehensiveness
- Automation

2.2. Gamification

The following chapter's aim is to clarify the main theory behind human motivation, gamification and the corresponding patterns and methods. Therefore first of all the term Gamification is defined and explained (chapter 2.2.1), furthermore there is an introduction to human motivation (chapter 2.2.2) and motivational design patterns (chapter 2.2.3). Moreover the Gamification Design Process is introduced (chapter 2.2.4). Finally the effectiveness of gamification in terms of business software is discussed. (chapter 2.2.5).

2.2.1. Definition

The term gamification is defined by Kumar and Herger as follows:

> *"Gamification is the application of game design principles and mechanics to non-game environments. It attempts to make technology more inviting by encouraging users to engage in desired behaviors and by showing the path to mastery. From a business viewpoint, gamification is using people's innate enjoyment of play."*
>
> GAMIFICATION [14, P. 8]

Based on the above definition gamification aims to motivate the user to do something [14, p. 8]. That is why the next chapter provides a more comprehensive introduction on motivation.

2.2.2. Human Motivation

The game design principles and mechanics which are used in the context of gamification are a specialization of motivational design patterns used in Human Computer Interaction. [14, p. 59]

Therefore this chapter provides an introduction to the ground of gamification from the areas psychology of motivation, behavioral psychology and behavioral economics - all three dealing with Human motivation.

Psychology of motivation Human motivation is one of the main topics of psychology. Some questions which arouse are: What motivates humans for doing something? What intentions do they pursue with their doing? Which activities are a pleasure for them? [15, p. 1]

There are two types of motivation: extrinsic and intrinsic motivation. On the one hand intrinsic motivation is based on an internal drive to do something. Humans are doing this task for their own. Possible motivational factors are gained autonomy, mastery or freedom. [15, p. 2, 3, 4], [14, p. 60, 61]

Deci describes intrinsic motivation as follows: "One is said to be intrinsically motivated to perform an activity when he receives no apparent rewards except the activity itself." [16, p. 105]

On the other hand extrinsic motivation is based on motivational factors from the outside, such as money, throphys or the comparison with others (for example with points, levels or leaderboards). [15, p. 2, 3, 4], [14, p. 60, 61]

One theory dealing with the core psychology behind motivation is the self-determination theory by Ryan and Deci. According this theory human motivation depends on the satisfaction of the three psychological basic needs:

1. Autonomy
2. Competence
3. Relatedness

Based on Deci and Ryan whenever humans feel autonomous, competent and related then motivation arises. [17, p. 416-432]

Flow is another concept based on intrinsic motivation. It describes the situation when different actions steps run and merge smoothly without any problems. The entire attention belongs to the current task and no concentration is necessary to focus on the task. The basis for being able to experience flow are a clearly defined aim, concrete action steps and the tasks submit feedback regarding their correctness. [15, p. 19, 20, 21]

Interest describes a current state of mind supporting knowledge building. It can be explained by a general preference for specific topics (e.g. specific school subjects), or by situational factors (e.g. interesting educational topics). Interest can be a catalyst for intrinsic motivation. [15, p. 22, 23, 24]

Behavioral psychology Behavioral psychology studies the way how humans behave and tries to find underlying patterns which trigger a specific behavior. There is a constant stream of inputs (stimuli) to our body. In the field of behavioral psychology human behavior is seen as a response to these inputs. [18, p. 10]

A concrete application, where behavioral psychology can be observed are learned processes, also known as operant conditioning. Prominent Experimental Research in the area of operant conditioning was done by Skinner and his experiments known as Skinner box. For a deeper insight into his experiments, his book "The behavior of organisms" [19] is referred. By rewarding desired behavior and punising undesired behavior humans get conditioned to display specific desired behaviors. Rewards and punishments are the stimuli causing responses. [18, p. 11]

Moreover the timing when rewards are provided, influences how the interaction works. Based on Lewis [18, p. 10] there are four different strategies:

1. Fixed Ratio: After a fixed number of responses rewards are provided (e.g. coffee card: the tenth coffee is for free)

2. Variable Ratio: Reward frequency is not firmly defined, the reward is offered on average after a couple of responses (e.g. gambling machine)

3. Fixed Interval: Rewards are provided after a fixed period of time (e.g. coffee machine)

4. Variable Interval: The interval in which rewards are offered is variable (e.g. fishing)

The highest response over time is generated by variable ratio strategy. In case of designing engaging applications, connecting the user with this application one should consider the use of rewards in a variable ratio. [18, p. 11]

So large parts of the gamification principles are based on rewards (e.g. increasing points, levels) and punishments (e.g decreasing points and levels). However the application of these principles should always be done carefully. This can be illustrated by a thought experiment by Schell called "chocofication". First of all there is the fact that chocolate tastes good. Adding chocolate to peanut butter makes it tasting good. But nevertheless the conclusion that everything tastes good with chocolate is wrong. For example hot dogs with chocolate are a disaster. To conclude you can say, that based on the thought experiment chocolate is not the magic bullet for food, alike gamification is not the magic bullet for application design. [18, p. 12]

Behavioral economics Behavioral economics explores, which effects affect economic decisions. In general whenever a resource (e.g. time, money) is gained or lost it is the consequence of a decision. So behavioral economics could also be seen as the theory behind decision making. Moreover in the context of Human Computer Interaction whenever a user interacts with an application lots of decisions are made. Engaging application design tries to include aspects of behavioral economics to influence the users decision to spend more time with the application. Human decisions could be rational or irrational. Rational decisions are made to reach a concrete aim such as happiness and can be logically explained. Irrational decisions are not necessarily comprehensible. Nevertheless irrational decisions can be triggered by external influences. For example people tend to use memberships, even if they don't profit (e.g. injured people going to the gym to make use of the membership). Referring to the relationship between behavioral economics and application design the application can be designed to trigger the user to make an irrational decision (e.g. spend more time with the application than needed). [18, p. 19]

Patterns which motivate the user to do something by using the theoretical background of motivation, behavioral psychology and behavioral economics are described in the following chapter 2.2.3.

2.2.3. Motivational Design Patterns

The theoretical concepts above are used in various motivational design patterns. In Lewis [18] and Kumar and Herger [14] motivational design patterns are described. In the following some patterns which may be relevant for the conception of the risk management application are introduced. The selection criteria was the applicability of the pattern in the context of a business application for risk management. For more comprehensive insight into motivational design patterns please refer to [18] and [14].

Based on [18] patterns can be classified. The presented patterns are out of the classes: Gameful Patterns, Social Patterns, Interface Patterns and Information Patterns. Gameful Patterns focus on operating methods known from games, Social Patterns, enable the users to satisfy their social contact needs, Interface Patterns are dealing with the influence of the interface on the user's behavior and Information Patterns study the way how content and information can be presented. [18, p. 4, 5, 6]

Gameful Patterns

- Collection: Collecting and owning virtual items (e.g. Forza Horizon, Pokémon). [18, p. 4, 35]
- Specialization—Badge: The user has reached a goal which is now visible through a badge (e.g. Xbox 360). [18, p. 4, 37]
- Growth: User owns something which was reached over time (e.g. SimCity). [18, p. 4, 40]
- Increased Responsibility: Trust in a user is the underlying basis for getting responsible tasks (e.g. Stack Overflow). [18, p. 4, 41]
- Leaderboard: Ranking users based on specific metrics (e.g. Doodle Jump). [18, p. 4, 44]
- Score: Based on the reward principle. By performing desired behavior the user normally achieves points, presenting his/her achievement level (e.g. Pac-Man) [18, p. 4, 46]

- Challenge: Challenges motivate users by giving them the feeling of reaching something great (e.g. Runkeeper) [14, p. 77, 78]
- Constraints with urgent optimism: Urgent Optimism combined with deadlines leads to a motivational effect. It is an extreme form of self-motivation combined with the belief in the reachability of the aim. [14, p. 78]
- Journey (Onboarding, Scaffolding, Progress): Journey describes the adaptability of the application based on different usage phases. One can think about a specific onboarding process providing an introduction and help regarding the application. The next phase after onboarding is scaffolding. The user is still inexperienced leading to a risk of operating errors. By providing support and constant feedback the bounce rate is minimized. Finally the user is onboarded and knows the main concepts of the application and is able to use them. Nevertheless constant user engagement is still desirable. It can be implemented with elements clearly showing users their current progress and feedback loops. (e.g. Setup process for LinkedIn) [14, p. 80, 81, 82]

Social Patterns

- Activity Stream: Representation of current events as never ending stream of news (e.g. Facebook). [18, p. 4, 52]
- Broadcast: Information can be shared between different users (e.g. Facebook, Twitter). [18, p. 4, 53]
- Social Feedback/Feedback loops: Users are able to easily feedback something. Furthermore multiple feedback loops are possible (e.g. Facebook). [18, p. 4, 54]

Interface Patterns

- Notifications: The user can be alerted by the application when a change occurs (e.g. Android, iOS) [18, p. 5, 70]

- Praise: Rewards for performing desired behavior (e.g. FarmVille) [18, p. 5, 72]
- Predictable Results: The results of an action are clearly predictable for users. (e.g. Google Search always provides search results) [18, p. 5, 74]
- State Preservation: The current state of the application is stored at any time, no matter when the application is left (e.g. Google Docs) [18, p. 5, 75, 76]
- Undo: The user is able to revert actions (e.g. Google Docs) [18, p. 5, 79]

Information Patterns

- Organization of Information: When information is presented ordered and organized the retrieval afterwords is simpler (e.g. Outlook) [18, p. 6, 85, 86]
- Personalization: Based on the individual user preferences the application adapts itself (e.g. Amazon) [18, p. 6, 87]
- Reporting: Reporting inappropriate content by users is possible (e.g. Facebook) [18, p. 6, 90]
- Search: Huge content is easily searchable (e.g. Google Search) [18, p. 6, 90, 91]
- Task Queue: Presents tasks which can be done next by a user trying to keep the user using the application (e.g. Setup process for LinkedIn) [18, p. 6, 93]

2.2.4. Gamification Design Process

According to [20, p. 5, 6] and [14, p. 27, 28] a well established design philosophy is User Centered Design. The center of the whole design and development of the application is the user. With this approach it is getting possible to match the users needs. The developed application is intuitively operable for the user and increases the user's productivity.

In the context of gamification the User Centered Design Process can be adapted to be a Player Centered Design Process.

Based on [14, p. 29-32] it consists of five steps:

1. Player

   Firstly it should be clearly defined who is the user, respectively the player. Based on a profound knowledge of the player and his needs the application can be designed. Therefore user/player personas are created, describing different user/player types, interacting with the application. The following user/player persona template is based on [14, p. 38-45]:

2. Mission

   Secondly the main goal of the gamification process is identified, the so called mission. Figure 2.3 represents the S.M.A.R.T Mission process to identify the mission. First of all the current situation is analyzed and the target business outcome is studied. Based on the gained knowledge a mission for the gamification process is set. It should be specific, measurable, actionable, realistic and time-bound. [14, p. 49-52]

3. Human Motivation

   Thirdly based on the theory behind human motivation (chapter 2.2.2) the concrete motivational factors for the different user personas are defined. [14, p. 59-67]

4. Game Mechanics

   Game mechanics represent the area of adding concrete gameful patterns to a non game environment. As part of motivational design patterns gameful patterns are described in chapter 2.2.3. While implementing gameful patterns in non game environments one

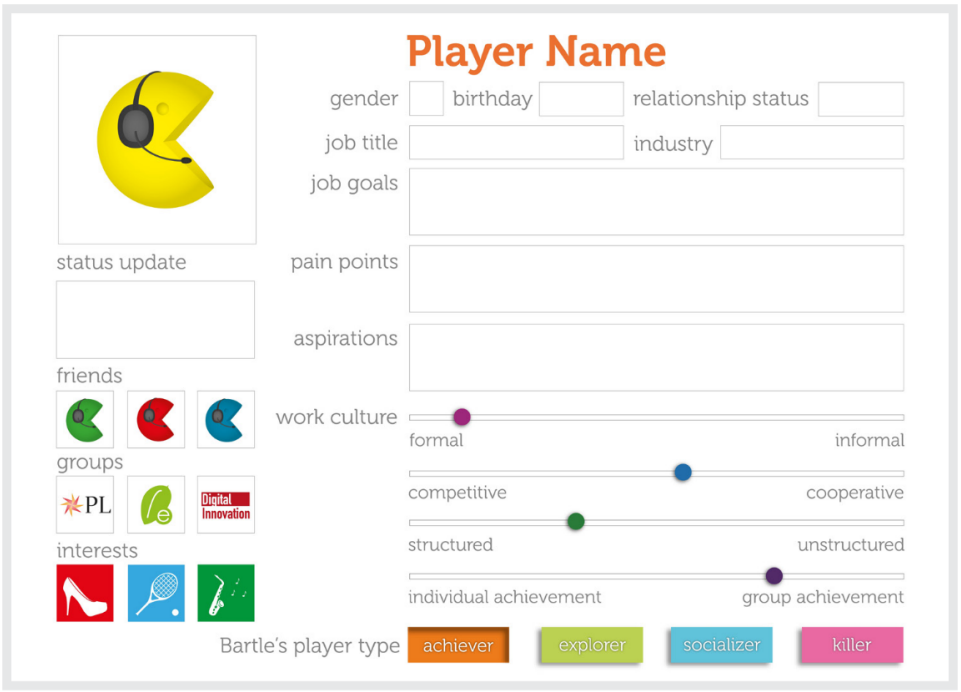

FIG. 2.2.: *Player Persona Template*
[14, p. 46]

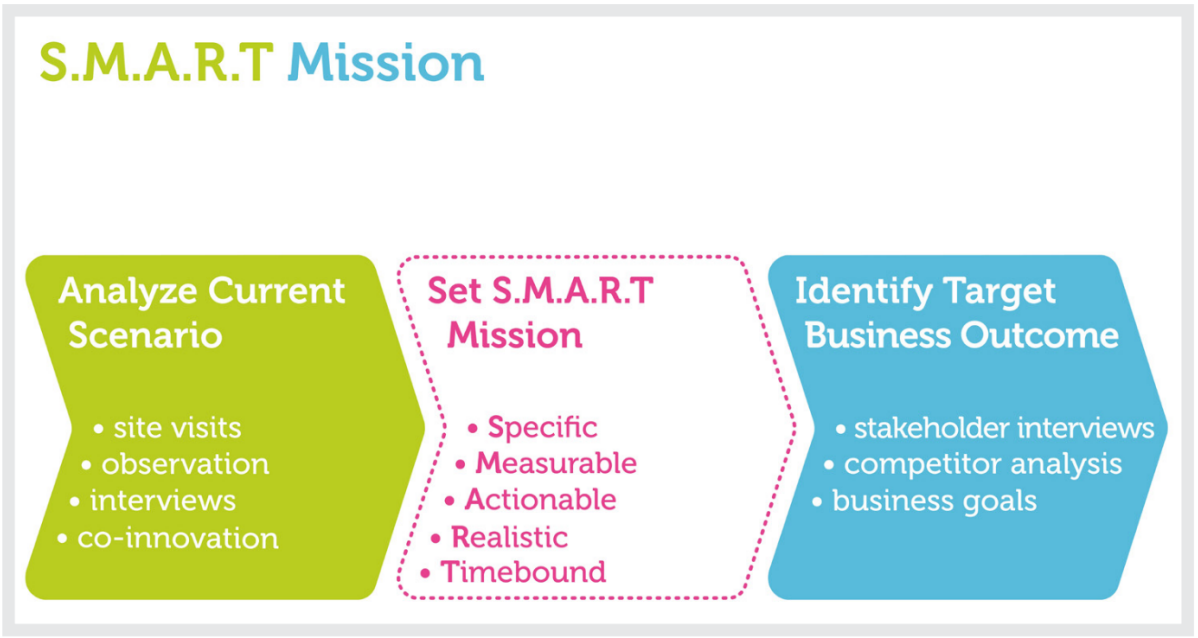

FIG. 2.3.: *S.M.A.R.T. Mission*
[14, p. 50]

should take into account that adding all patterns to an application normally doesn't reach the resumed aim. Hence the selection of fitting patterns must be adapted to the prescribed context. The main aim behind adding gameful patterns is to build a positive engagement loop centering the user/player. Figure 2.4 shows the four main steps of the engagement loop, starting with a motivating emotion. [14, p. 69-71]

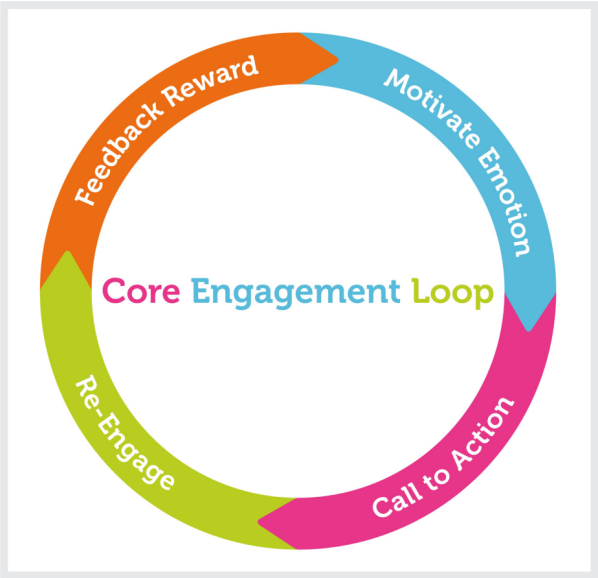

FIG. 2.4.: *Engagement Loop*
[14, p. 88]

5. Manage, Monitor and Measure

   After applying specific game mechanics to an application there are few points left, which should be observed in production. On the one hand the mission should be managed. Based on the S.M.A.R.T. Mission process the identified mission should be checked frequently and if needed adapted. On the other hand the user/player behavior should be monitored and measured to evaluate the effectiveness of the implemented patterns. This can be done qualitative by surveys and interviews and quantitative by tracking and data evaluation. Based on the acquired knowledge the application can be enhanced in the future. [14, p. 92-96]

2.2.5. Discussion

A literature review from Hamari, Koivisto and Sarsa [21] tries to answer the question if gamification works. Therefore quantitative and qualitative studies on this topic had been analyzed, resulting in the statement that quantitatively there are positive effects of gamification, but the gamification elements are only partly responsible for these effects. The analysis of qualitative studies resulted in the statement that gamification is more versatile than often assumed. [21, p. 3029, 3030]

The next question arising from this is: What are the reasons for these results and which disruptive factors harm the effectiveness of gamification? Therefore the study's conclusions are analyzed resulting in two aspects: Influence of the gamified context and user qualities. [21, p. 3029, 3030]

Influence of the gamified context The context which should be gamified influences the prospects of success. Hamari, Koivisto and Sarsa name three contextual factors [21, p. 3029, 3030]:

1. Social environment:
   In order to form behaviors one key for success is the voluntariness of doing something. [21, p. 3030]

2. Nature of the system:
   Systems which should be gamified can be hedonic or utilitarian. Hedonic systems support their users reaching desire and pleasure. [21, p. 3030] They are based on the philosophical concept of hedonism, which centers the human pursuit of desire and pleasure. Only the steady pursuit can reach intrinsically motivation. [22, p. LV]

   On the contrary utilitarian systems are purpose-oriented. The underlying philosophical concept is called utilitarianism. It is based on the principle that an action is morally correct when it maximizes the aggregated overall benefit, that is the sum of the welfare of all concerned. [22, p. 3]

3. Involvement of the user:
   Depending on the application's context there are two types how a user can be involved: cognitive or affective. [21, p. 3030] Cognitive involvement describes the user's interest respective an application. When being affectively involved, one evolves specific feelings regarding the application. In the context of business application normally the user's are involved cognitively. [23]

The overjustification effect describes the consequences how intrinsically motivated users change their behavior when extrinsic incentives are added. By adding extrinsic motivation the intrinsic motivation decreases. [15, p. 9-13]

Moreover there is the risk of false incentives. When applying gamification patterns without really thinking about the consequences it can lead to misguided behavior. E.g. when every user who contributes a risk to the project gets points, that will create a false incentive, leading to lots of contributed risks, but little attention to the risk management of each risk. [14, p.69]

User qualities The different abilities and qualities of users have a decisive influence on the users behavior while using the application and thus the success of gamification. Each user interacts differently with the application. E.g. positive gamification effects where only measurable inside a specific context or with specific users. [21, p. 3029, 3030]

2.3. Progressive Web Apps

As of today, devices such as mobile phones and personal computers come with their own app store. Microsoft offers their own Store, Google their Play Store and Apple the App Store. Users often find themselvews worrying about an app they once saw running on another platform not being available for their platform (e.g. Apples iOS and Googles Android) [24, p. 3].

Progressive Web Apps (PWAs) approach these concerns by trying to move away from app stores onto a platform which is available on most devices – the web browser. This means that PWAs are regular web apps at their core but can progressively leave the web browser. For example, PWAs can be installed on the underlaying operating system and be accessed from the app switcher or the taskbar and be executed in full screen mode without the browsers interface being visible [25, p. 26]. Further characteristics of PWAs and how exactly a PWA can leave the web browser are granularly described in the following chapter.

2.3.1. Characteristics of a Progressive Web App

To transform an existing web app into a PWA or build one from scratch one must implement different criteria instead of including a new framework or library [24, p. 6]. While the following eight characteristics represent Mozilla's (Firefox) ideas of a PWA, other web browser manufactures such as Microsoft (Edge) or Google (Chrome) have roughly the same idea about PWAs and describe eight or ten characteristics respectively within their developer documentations [25, p. 90], [26].

1. **Progressive**: The first characteristic defines that a PWA should not exclude the user from using the core functionality but extend the user experience by embracing new features implemented by the web browser manufactures [25, p. 100], [27, p. 2]. For example, a web app to check mails should not exclude the user from checking their inbox or sending mails but could provide push notifications to inform about incoming mails. In this example the user is not excluded from using the core functionality of a mail app (checking the inbox and sending mails) and user experience is enhanced by push notifications. To avoid

unexpected failures a developer should follow the *Feature Detection* principle which says an application should not blindly use a non-standardized feature before checking its existence [25, p. 101].

2. **Network Independent**: Using a regular web app on the go can be a problem, especially in regions with little to no mobile reception or no stable Wi-Fi being around. Thus, the dynamic content of a web app does not load within a tolerable timeframe or a user is inhibited to perform actions like sending a mail [25, p. 106]. On these grounds a technology called *Service Worker* has been established and implemented by many browser manufactures. In short, a service worker is a script that is able to listen to the network traffic caused by a PWA and therefore is able to cache possible answers fetched from a server and serve them to the web app when no stable network connection is available or do background syncing by running code even when the web app isn't in use. For more information about the service worker see chapter 2.3.3 [24, p. 43].

3. **Safe**: As mentioned in the previous paragraph, service workers can run code independently from the PWA. To avoid harmful service workers from running malicious code, browser manufactures expect PWAs to be served by a trusted host over a secure connection. To be more precise, over an HTTPS connection [24, p. 24]. *HTTPS* stands for Hypertext Transfer Protocol Secure and is based on *TLS* (Transport Layer Security). Once the host has obtained a digital certificate for its domain, this certificate is being transferred to the client where it can be verified by the web browser. On success an HTTPS connection can be established and every upcoming network traffic will be encrypted [25, pp. 112-113].

4. **Re-engageable**: A feature which native apps are using for a while now are push notifications. Push notifications are a common way to inform users about the newest events such as a new mail in the user's inbox. Thanks to the Push API that is implemented on top of the service worker, just like native apps, PWAs can keep the users engaged by sending notifications as can be seen in figure 2.5 [27, p. 201].

5. **Responsive**: This characteristic specifies that the PWA render its user interface corresponding to the devices used to access it. This is necessary as the available space and

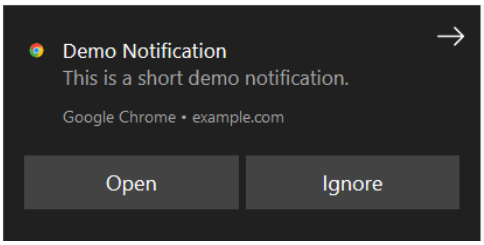

FIG. 2.5.: *Push notification to keep users engaged*

input method can change from device to device. The screen of a phone on average is way smaller than the screen of a notebook. Furthermore, using fingers to interact with a phone is less precise than using a mouse on a notebook [25, pp. 115-116]. In figure 2.6 for example one can see how the content and navigation arrange differently on each device. Due less space the navigation bar on the mobile version (extract on the right side) is completely collapsed and can be accessed by clicking the so-called burger-like icon while on the desktop version the whole navigation bar is visible.

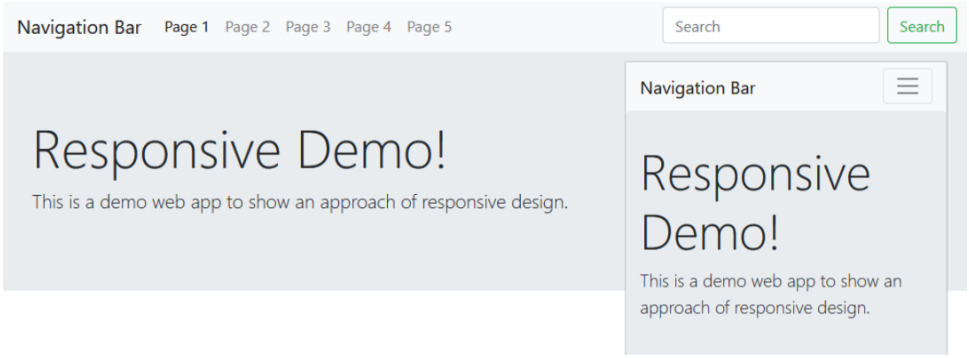

FIG. 2.6.: *Responsive design*

6. **Discoverable**: As PWAs are not a new framework or library but regular web apps at their core, there needs to be a method to distinguish between a PWA and a regular web app. This is necessary for web browsers to provide additional features to PWAs such as an option to install (see next paragraph). To make a PWA discoverable a "Web Manifest" file (see chapter 2.3.2), which contains information like the name of the PWA, needs to be provided [25, p. 118].

7. **Installable**: To take things even further, besides offline functionality, a PWA should be installable to the user's device. In detail, a user should be able to install the PWA from within the web browser to the underlaying operating system like Android or iOS. From this point on the user can launch the PWA directly from the devices home screen like it is shown in chapter 2.3.2. Different browser manufactures expect different requirements to be fulfilled before they provide an option to install. Mozilla's Firefox for example expect that the PWA is network independent, safe and discoverable [28].

8. **Linkable**: The last characteristic implies that a PWA is referable by a *URL* (Uniform Resource Locator, e.g. "www.example.com") instead of requiring to be installed via any app store. Ideally, the URL should also point to different views of a PWA like a profile page for a specific person. Hence, the current view can be easily shared between users. As PWAs are being run by a web browser, which needs a URL to access the web app in first place, this characteristic, in its fundamentals, does not require any further attentiveness by the developer [25, pp. 126-127].

2.3.2. Web Manifest

The web manifest is a *JSON* (JavaScript Object Notation) file. Its primary task is to make a PWA discoverable and installable (see chapter 2.3.1, p. 22) by providing descriptive information, like a short app name and paths to icons, about the PWA. The following listing shows a minimal web manifest:

LISTING 2.1: *Example Web Manifest*

```
{
  "short_name": "PWA Demo",
  "name": "Progressive Web App Demo",
  "description": "A simple Progressive Web App Demo.",
  "icons":
    [{"src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon" },
    {"src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192" },
    {"src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512" }],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#dddddd",
  "background_color": "#ffffff"
}
```

The `(short-)name` represenst the name of the PWA which is used on the app switcher or home screen, depending on how much space is available. `icons` contains various file paths to app icons with different sizes which are used in different scenarios like the app switcher, home screen or the apps splash screen. For each use case the most appropriate size is chosen automatically. `start_url` defines the entry point of the PWA, `display` holds information about how the PWA will be displayed once it is installed (e.g. `standalone` for no web browser elements) and finally `theme`- and `background_color` which determine the primary color of the user interface and the background color of the splash screen respectively [28].

Figure 2.7 shows the effects of this web manifest on an example PWA. On the left one can see the use of the icon and name field in Googles Chrome "Add to Home Screen" prompt. In

the middle the short name is used due the given space. Once the PWA is launched, like on the right, the standalone display mode is used which hides all elements of the web browser.

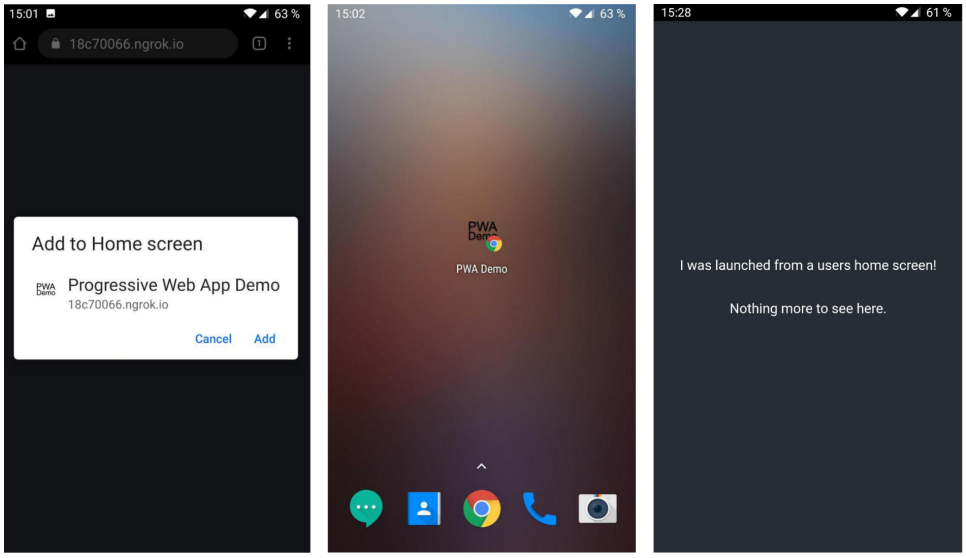

FIG. 2.7.: *Outcome of the web manifest*

2.3.3. Service Worker

A service worker is a script written in JavaScript, running in the context of the web browser. Its primary purpose, as mentioned in chapter 2.3.1, is to cache content and provide it to the PWA whenever a slow or no connection to the internet exists. Background syncing as well as sending push notifications can also be realized with a service worker. To achieve this, a service worker has three specific traits: being *controller*, *interceptor* and a *proxy* [25, p. 176].

After being registered by the web browser, which means that a HTTPS connection is established and it has been requested by the PWAs source code, it has full control to all in- and outgoing network traffic, hence the trait of a controller. Interceptor means that the service worker can manipulate and inspect the network traffic and proxy as the service worker is able to decide if the outgoing traffic should be redirected to the requested resource or completely avoid any outgoing traffic by answering with data it stored in a cache previously. The latter is the exact reason why a service worker is indispensable for the Network Independent characteristic (chapter 2.3.2, p. 25) of a PWA [25, p. 176-177].

The upcoming listing demonstrates how a service worker can manipulate and proxy outgoing network traffic. Before, lets assume that a cache called `pwa-demo` was already created by the service worker.

LISTING 2.2: *Cache First Service Worker*

```
1 self.addEventListener('fetch', event => event.respondWith(
2    caches.open('pwa-demo')
3      .then(cache => cache.match(event.request))
4      .then(response => response || fetch(event.request))
5   )
6 );
```

In the first line one can see the service worker is referencing itself and adds an event listener for all fetch-events (requesting data from outside of the web browsers context). The event listener gets passed the fetch `event` as its second argument on which it calls the `respondsWith` method. Within that method, it opens the cache called `pwa-demo` in line two to then check if the requested event matches the data stored in the cache in line three. If it does match, it then directly returns the data back to the PWA. Otherwise it executes the right part of the conditional OR expression `||` and calls the `fetch` method to get the data from the requested resource [25, p. 60].

This strategy is called *Cache First* as the service worker will look up the cache before requesting resources outside of the browser's context. Once cached, the data will available much faster and offline but if the resource changes its content, the service worker will still return the old, cached version. On the other side the *Network First* strategy exists that will always fetch data when a connection to the resource (e.g. the internet) can be established. Thus, the user will always receive the newest content and has a slightly older version available when being offline. On the downside, if the connection is slow the content may take a while to be fetched. Therefore, the *Cache and Network* strategy combines both, the Cache- and Network-First, strategies. To bridge the time of fetching new content, the user is presented the cached content until the result

is available and then be presented by refreshing the user interface. These are just a few but popular strategies and each has their own scenarios where they work best [27, pp. 109-111].

2.3.4. Compatibility

As with every new specification introduced it takes some time for every manufacturer to fully implement it in their products. In this chapter a short overview is given over which feautures web browsers currently support, in terms of PWAs. The following figures are taken from www.CanIUse.com (17/10/2019). They show to which grade a web browser version supports a given specification. Red means no support, dark green fully supported and light green supported to a degree.

In figure 2.8 one can see the web browser compatibility of the web manifest for richer offline experiences - like being installable.

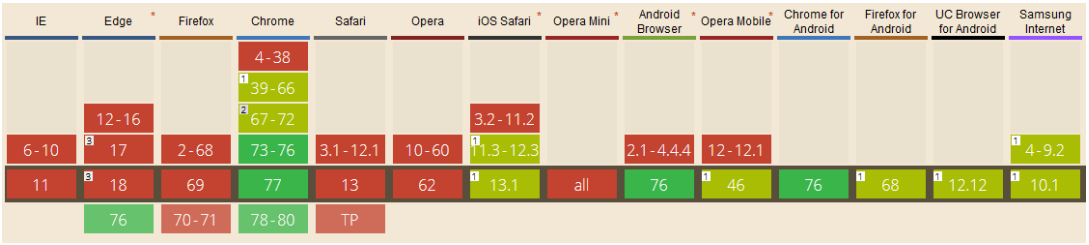

FIG. 2.8.: *Compatibility of the web manifest*

The web manifest is currently not widely supported on the desktop versions of many browsers. Currently only Googles Chrome fully supports it, in return though, most major mobile browsers (except Opera Mini) at least partly support the web manifest.

Figure 2.9 shows the web browsers support for the service worker whose primary goal is the offline functionality. Regardless of desktop or mobile platform, it is currently supported by every major web browser except Internet Explorer and Opera Mini.

If the PWA should use push notifications to inform its users about new occurrences and fulfill the re-engageable characteristic (see chapter 2.3.1) it can use the Push API which is another specification that needs to be implemented by the web browsers manufacturer.

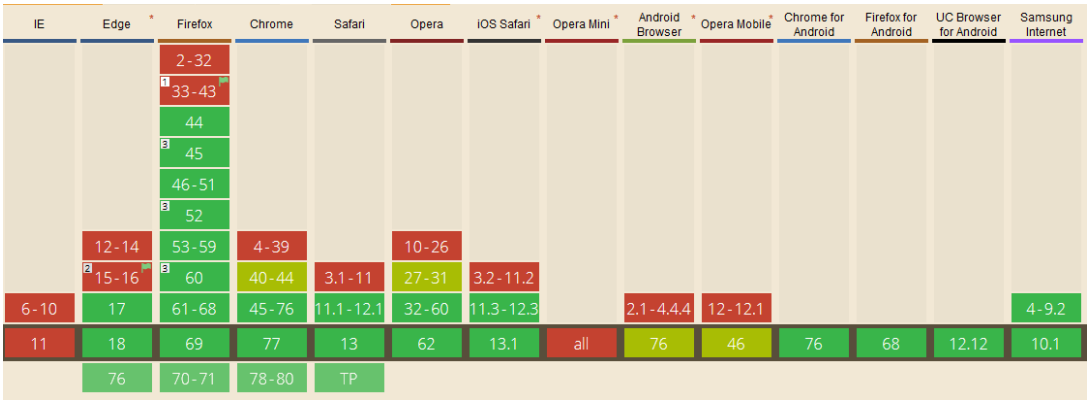

FIG. 2.9.: *Compatibility of the service worker*

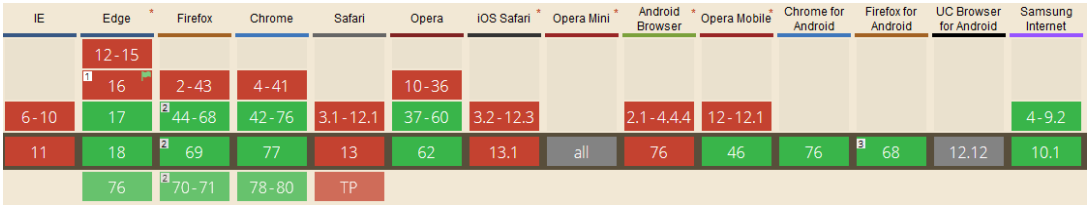

FIG. 2.10.: *Compatibility of the Push API*

As seen in figure 2.10, most manufactures have implemented this feature, Apple falls behind with their desktop and mobile web browser Safari as well as Microsoft's Internet Explorer.

Thinking back on the first characteristic of a PWA, being progressive, a non-supported feature (e.g. the web manifest in Mozilla's Firefox) won't lock the user out of the app as all these specifications should only enhance the user experience and not lock the user out from using the core functionality of a web app.

3. Conception

text...

3.1. Survey

3.2. Software Specification

3.2.1. Purpose and Scope

What is the software for and what is not part of the project (I'm integrating the vision part here because I don't think we need to state why we describe requirements)

3.2.2. Functionalities

Basically just the UC

3.2.2.1. Overall Use Case Diagram

3.2.2.2. Use Case Specification: UC1 User CRUD

Description

This use case specifies how user accounts are being created, updated, deleted and are used for login and logout. These fields are required for the user account creation:

- e-mail adress (String)
- password (String)

Additionaly the user can set the following field while updating his account:

- language (enum)

Screenshots

Insert screenshots and shortly explain what can be seen

FIG. 3.1.: *Use Case X: Detail*

Basic Flow

New Account:

- The visitor clicks the "register" button and fills in the fields mentioned in the brief description.
- Then clicks on the "register" button below the form to send the input to the server.
- If the visitors input fulfills the criteria of a username, email and password the account is being created on the server and a conformation e-mail is send to the given e-mail address.

The account itself can only be used after by clicking on the provided link in the conformation e-mail.

Update Account:

- The user is logged in and clicks on the "settings" button.
- After updating the the fields the user wants to change, the input is sent to the server.
- If the updated fields fulfill their criteria, those fields are updated on the server.

Delete Account:

- The user is logged in and clicks on the "settings" button.
- The user then clicks on the "delete my profile" button.
- After typing in the users password, the request to delete the profile can be send to the server by clicking on the "delete" button.
- If the given password match the user's password the account is being deleted on the server and user will be logged out.

Login:

- The visitor already created an account and wants to log in.
- Therefor the "login" button must be clicked and the e-mail and password fields have to be filled in.
- The visitor can then click the "login" button below the form.
- If the e-mail and password match an entry of the list of registered users, the visitor will be logged in and can act as an user.

Logout:

- The user is logged in and wants to log out.
- By clicking the "logout" button the client removes all information used to authenticate to the server and forces the user interface to render without any user specific information fetched from the server.

The user is now able to login again.

Activity Diagram

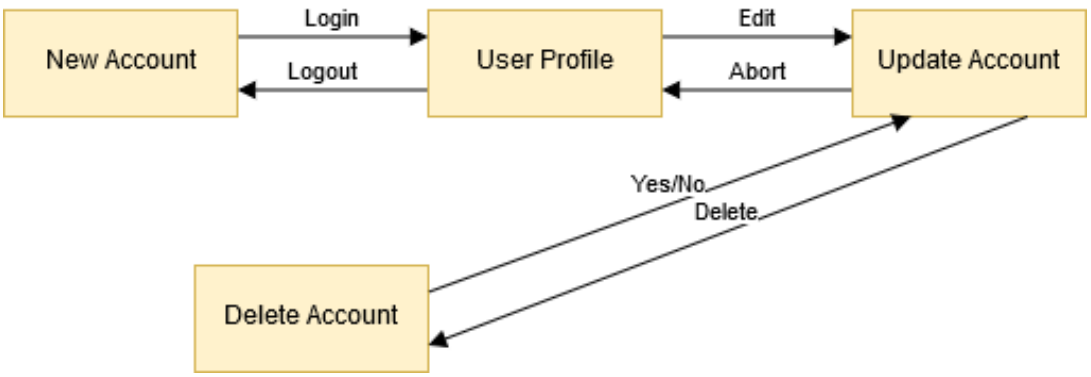

FIG. 3.2.: *Activity Diagram UC1 User CRUD*

Alternative Flows

Invalid input:

- The user / visitor fills in the required fields (username, e-mail, password, language).
- If one field does not meet its criteria (e.g. an e-mail should contain the @-sign), which is validated by the server after submitting the input, the user / visitor will be informed next to the input fields.

In this case the user account won't be created nor updated.

Invalid user credentials:

- If a visitor wants to login into an existing user account or if an account should be deleted by a user, the user accounts credentials have to be submitted (e-mail and password for login, password only for deletion).
- If the account credentials do not match any entry on the server, the user / visitor will be informed next to the input fields.

In this flow the visitor won't be logged in and neither will the account be deleted.

Special Requirements and Preconditions

Different flows require different preconditions.

1. If a visitor wants to log into an user account has to exist or created beforehand.

2. If a user wants to log out, the user has to be logged in.
3. If a user wants to update or delete the account,the user has to be logged in, too.

Postconditions and Persistance

Mentionable postconditions are:

1. After deleting an account, the account can not be restored and therefor not being used anymore.
2. Furthermore logging in allows a user to then access user related content from the server and logging out hides all user related content respectively.

The persistence guidelines are:
Creating, updating and deleting a user account or logging in by filling in the required fields and then submitting the input leads to a POST request to the server. If the fields fulfill their criteria, the information will be persisted on the server. Logging out will only delete the users authentication information on the client side and will not send a request to the server.

3.2.2.3. Use Case Specification: UC2 Project Access Management CRUD

Description

This use case specifies how projects are being created, updated, deleted and are used for grouping a pool of users and risks. The fields in the list below are required for the project creation:

- projetc name (String)
- project description (String)
- start date (Date)
- end date (Date)

Besides the required fields the project owner can set the following field while updating a project:

- project members (Strings: Usernames, E-Mails)

Screenshots

Insert screenshots and shortly explain what can be seen

FIG. 3.3.: *Use Case X: Detail*

Basic Flow

New Project:

- The user is currently on the project page and clicks the "new project" button and fills in the required fields.
- Then clicks on the "create" button to send the input to the server.
- If the users input fulfills the criteria of a name, description and time span, the project will be created on the server and be available in the project list.

Update Project:

- The user opened the detailed project page by clicking on it on the project list and then clicks the "edit" button.
- After updating the fields the user, to be more precise the project owner, intended to change, those updated fields are send to the server by clicking the "save" button.
- If the updated fields fulfill their criterias, those fields are updated on the server and in the user interface.

Adding users while updating a project is the basis to work on risks together as described in chapter 3.2.2.4

Delete Project:

- The project owner opened the detailed project page by clicking on it on the project list and then clicks the "edit" button.
- The owner then clicks on the "delete" button.
- After typing in the users password, the request to delete the project will be send to the server once the "delete" button is clicked.
- If the given password match the owner's password the project, including all related risks, is being deleted on the server and removed from the project list in the user interface.

Activity Diagram

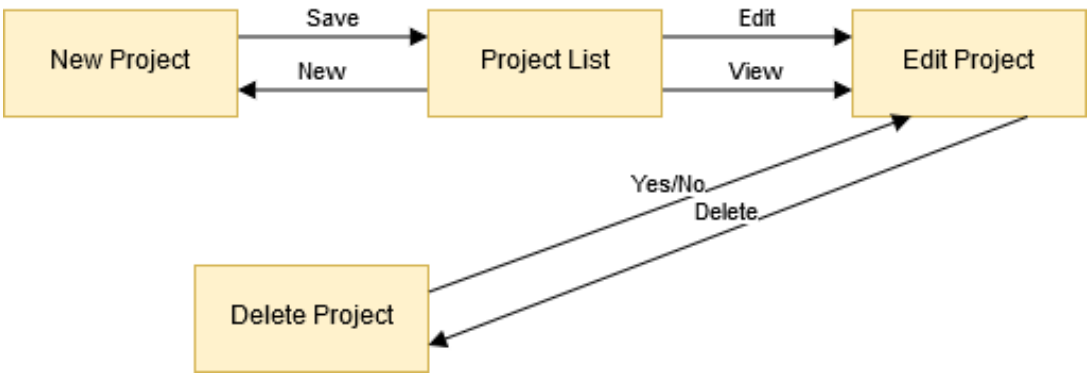

FIG. 3.4.: *Activity Diagram UC2 Project CRUD*

Alternative Flows

Invalid input:

- Like chapter 3.2.2.2 "Invalid Input", but for project creations and updates.

Invalid user credentials:

- Like chapter 3.2.2.2 "Invalid user credentials", but for project deletion.

Special Requirements and Preconditions

Generally, the user who created the project is automatically the project owner.

1. The user has to be logged in to create a project.
2. The user has to be the owner of a project to edit it.

Postconditions and Persistance

Mentionable postconditions are:

1. When deleting a project, the project including all related risks can not be restored.
2. After creating a project and adding users, users will be able to access it and create risks within.

The persistence guidelines are:
All mentioned basic flows above create POST requests to the server and additionally, if the fields meet their criterias the data will be persisted.

3.2.2.4. Use Case Specification: UC3 Risk CRUD

Description

This use case allows users to create, read, update and delete risks. A risk consists of the following fields:

- name (String)
- description (String)

Following fields are filled later and are not part of the input form:

- probability of occurence (Enum)
- impact (Enum)
- risk factor (Enum)
- response (Objects)
- person in charge (User)
- public risk (boolean)

Screenshots

tbd: Insert screenshots and shortly explain what can be seen

FIG. 3.5.: *UC3 Risk CRUD: TODO*

Basic Flow

Creating a risk:

- When the user clicks the "+" button at the project overview page.
- Then the screen for adding a new risk is opened.
- When the risk form is filled by the user.
- And the user clicks on the "Propose risk" button.
- Then the risk is synced with the server.

Reading a risk:

- The user is on the project overview site with all project risks.
- By clicking on a risk a detail risk view is opened.
- For exiting the risk detail view a return button ("Close" button) is clicked.

Updating a risk:

- The user is on the project overview site with all project risks.
- By clicking on a risk a detail risk view is opened.
- On the detail view there is a pen button, enabling editing and changing the "Close" button to a "Save" button.
- When clicking the "Save" button the changes are syncronized with the server.

Deleting a risk:

- The user is on the project overview site with all project risks.
- By clicking on a risk a detail risk view is opened.
- By clicking a "Delete" button the risk is deleted. This behavior is changed in UC6 Risk Discussion described in chapter 3.2.2.7.

Activity Diagram

Alternative Flows

n/a

Special Requirements and Preconditions

The preconditions for this use case are:

1. A project exists.
2. The user is member of the project.

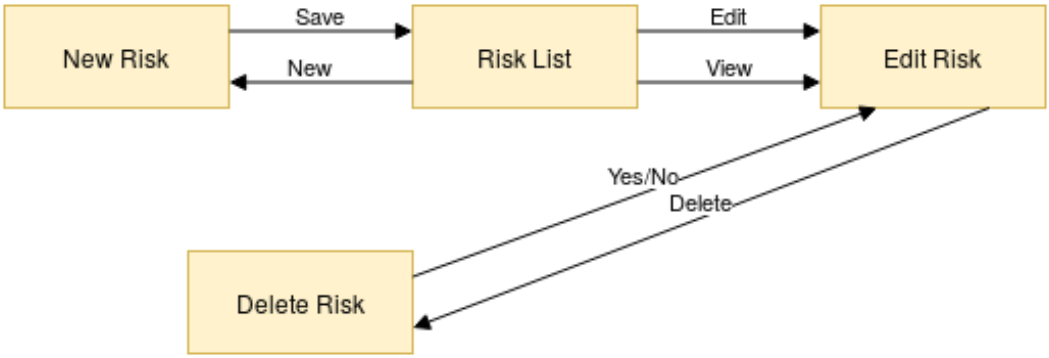

FIG. 3.6.: *Activity Diagram UC3 Risk CRUD*

3. The user has clicked the "+" button at the project overview page to add a new risk.

Postconditions and Persistance

The postconditions for this use case are:

1. The risk is immediately part of the projects risk table (this behavior is changed within UC6 Risk Discussion described in chapter 3.2.2.7).

The persistence guidelines are:
The risk form was completely or partly filled by the user. When the user tries to leave the page now, there should be a prompt for exiting. When the risk form is filled out and the button "Propose risk" is clicked a POST request syncs the status with the server.

3.2.2.5. Use Case Specification: UC4 Project risk adjustment

Description

After a risk has been submitted and evaluated by the team it is grouped into three categories. However, within these categories the initial order is simply that the risks submitted first will be placed at the top. This use case allows the teams to prioritize risks according to their perceived risk urgency. On the project risk overview page the after the initial risk evaluation users will find a button they can use to adjust the ranking. They will then enter adjustment mode where they can arrange the risks according to their perceived urgency and submit their ranking. Then they will return to the overview where the risks will be displayed according to their average relative ranking positions. The adjust button will be disabled until new risks are added to prevent manipulation. The user will be notified that they cannot rank multiple times when they first go through the adjustment process.

Screenshots

tbd: Insert screenshots and shortly explain what can be seen

FIG. 3.7.: *Use Case X: Detail*

Basic Flow

- The user is on the project overview site with all project risks.
- By clicking on a "Rank Risks" button the ranking view is enabled and the button becomes a submit button.
- The user can order the risks according to his personal priority.
- The user submits their preferred order.
- This order is submitted to the server and the average rank for every risk according to the current submissions is calculated.

- The risks on the risk overview page are displayed accordingly.
- The adjust button is disabled for users who have already submitted a ranking.

Activity Diagram

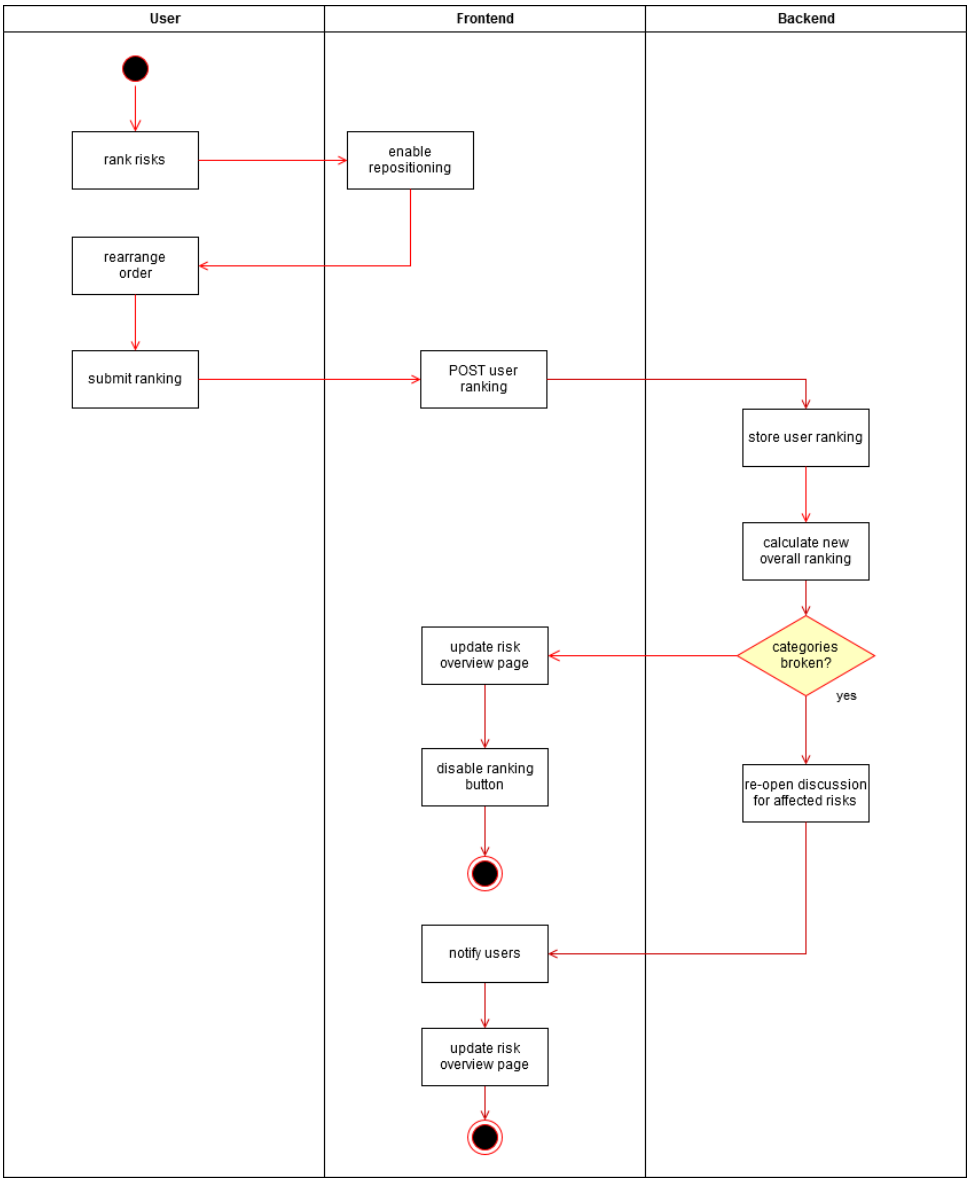

FIG. 3.8.: *Activity Diagram Use Case 4*

Alternative Flows

The following steps will be added if the user enters the ranking process for the first time or has not participated in a ranking process for a prolonged period of time:

- When the user presses the ranking button before the ranking mode is enabled an explation prompt will be shown.
- The user will be informed that they can submit their ranking only once.
- The user can either choose to confirm or they can choose that they do not want to be notified again doubling the time before the notification is shown again.

The following steps will be added should the risk adjustment contradict the intital risk assesment:

- A risk is ranked outside its category due to the average ranking process.
- The risk discussion on the risks which are now out of category is re-opened.
- The team members receive a notification to re-evaluate those risks.

Special Requirements and Preconditions

This use case has the following preconditions:

1. The user is part of a project which contains risks.
2. The initial risk discussion is finished.
3. The user has either not yet ranked the risks or
4. the risk adjustment has been re-opened by the project manager.

Postconditions and Persistance

The postconditions for this use case are:

1. The risk order will be updated for all project members.
2. The ranking option will be disabled until new risks are proposed or the ranking is manually reset by the project manager. The ranking results are persisted in the database.

The user ranking is transmitted to the backend via post request where the project risk ranking is handled accordingly.

3.2.2.6. Use Case Specification: UC5 Risk Pool

Description

The risk pool serves to share knowledge among different teams and to store experiences. Users can vote for project risks to be added to the risk pool. After a certain quorum is reached the risk will be moved to the risk pool and will be persisted together with its responses there. Whenever a new risk is created the user will then have the option to check the pool risks if their risk is already present. They can then add a reference to the pool risks to their project risks. Pool risk can already have responses attached to them and will display the average occurrence probability and impact severity estimates of their project references giving an indication of how other teams evaluated the risk. The individual project risk evaluation process will still be undertaken for pool risks.

To remove risks from the risk pool a user will have to request a voting process on the pool risk. They can choose to either have the risk deleted (because it is outdated for example) or to merge it with another risk. All users will then be notified that a pool risk is up for discussion and will be invited to vote. If the vote is favorable the risk will be deleted and in case of a merge request its responses and references will be moved to the merge risk.

Quorums can be adjusted by project managers however there will be a notification that the change affects all teams and other project managers will receive a notification should a quorum be changed.

Screenshots

Insert screenshots and shortly explain what can be seen

FIG. 3.9.: *Use Case X: Detail*

Basic Flow

Add flow:

- A user clicks on the "nominate for risk pool" button on a project risk's detail page.
- Other users on the project team will be notified that a risk has been nominated.
- The nomination button will be replaced by a support button for these users.
- Once a pre-defined quorum of users has supported the nomination the risk will be moved to the risk pool and the project risk will become a reference to the pool risk.

Remove flow:

- A user nominates a pool risk for being removed on the pool risk's detail page.
- All users who are part of a project that references the pool risks will be notified.
- The nomination button will be replaced by a support button for those users.
- If a pre-defined quorum of users supports the nomination the project references will be turned into copies of the original risk and the copies risk will be removed from the risk pool.

Duplicate flow:

- A user marks a pool risk as being a duplicate.
- The user will be shown a list of pool risks to mark the corresponding risk.
- All users which are part of projects that reference either risk will be notified.
- The nomination button will be replaced by a confirmation button for those users.
- If a pre-defined quorum of users confirms the duplication the risks will be merged.
- All references of the second risks will be redirected to the first, the response list will be merged and the duplicate will be deleted.

Activity Diagram

FIG. 3.10.: *Activity Diagram Use Case X*

Alternative Flows

The last steps of the above described flows will change if the quorum is not met within a pre-defined timeframe.

- The risks will not be changed and the status from before the nomination will be restored.

Special Requirements and Preconditions

This use case has the following preconditions:

1. The user is part of a project which contains risks. Or
2. There is a risk in the risk pool. Or
3. There are two similar risks in the risk pool.

Postconditions and Persistance

The changes are instantly reflected in the database. Once the changes are confirmed corresponding PUT and DELETE requests update the database.

3.2.2.7. Use Case Specification: UC6 Risk Discussion

Description

The process of adding risks to a project is based on three steps:

1. A project member proposes a risk (name and description). (UC3 Risk CRUD defined in chapter 3.2.2.4).
2. A specified number of project members review the proposed risk. After the review process the risk is ready for discussion.
3. The whole project team discusses and defines:
   - probability of occurence
   - impact
   - risk factor (defined by probability of occurence and impact)
   - response(s)
   - person in charge

Screenshots

tbd: Insert screenshots and shortly explain what can be seen

FIG. 3.11.: *UC6 Risk Discussion: TODO*

Basic Flow

1. A project member proposes a risk (name and description as defined in UC3 Risk CRUD chapter 3.2.2.4)
   - The proposed risk is visible in the section proposed risks at the project overview page ("Proposed risks").
   - All members of the project receive a notification (in their activity stream) about the new proposed risk and the request to review it.
2. A specified number of project members review the proposed risk. After the review process the risk is ready for discussion.
   - When the specified number of project members have positively reviewed the risk, it is visible in the section ("Risks open to discussion").
   - The project owner is able to manually start an estimation session for the risk (not recommended). The recommended way is to wait until a specified number of risks open for estimation are collected. Then an estimation session for all risks is automatically started.
3. The risk is estimated by the whole project team:
   - All project members receive a notification (in their activity stream) to estimate the risk in terms of (probability of occurrence, impact) and to add response(s). Probability of occurrence and impact are mandatory fields, response(s) are optional.
   - The risk factor is determined by the probability of occurrence and the impact.
   - It is checked if at least one response is defined. If not the project members are notified to add a response.
   - In the last step the person in charge is defined. Therefore a notification where the user is able to sign up for being responsible for the risk is sent to all project members.
4. Finally the risk appears at the project risk table ("Project risks").

Activity Diagram

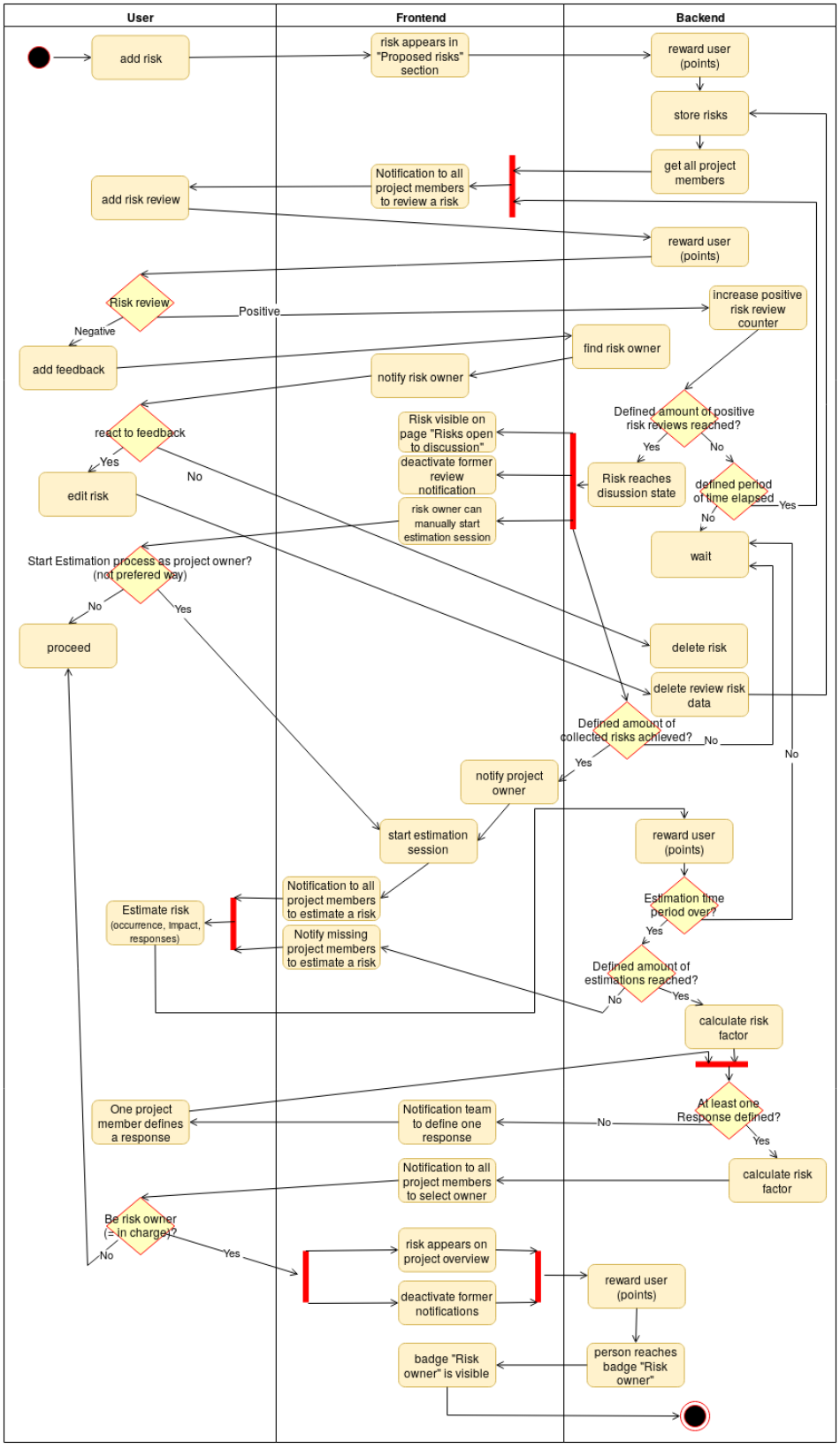

FIG. 3.12.: *Activity Diagram UC6 Risk Discussion*

Alternative Flows

Proposed risk receives negative reviews:

- A project member negatively reviews a risk.
- The reviewing person adds feedback for the risk which is sent to the risk owner.
- The risk owner can edit or delete the proposed risk.
- When the risk is deleted it is removed from the section proposed risks.
- When the risk is edited the review process data is removed and the message for reviewing is sent again.

Project members doesn't react to their notifications to:

1. review a risk: When the needed amount of reviews is not achieved the notification is sent again.
2. estimate a risk: When the needed amount of estimations is not achieved the notification is sent again.
3. define a person in charge: When no person volunteers the notification is sent again. For finding a risk person in charge fast the Gamification concept Challenge is used.

Special Requirements and Preconditions

The preconditions for this use case are:

1. A project exists.
2. The user is member of the project.
3. The user has proposed a risk.

Postconditions and Persistance

The postconditions for this use case are:

1. The proposed risk was reviewed and estimated.
2. The risk contains all relevant data.
3. The risk is visible as part of the projects risk table.

3.2.2.8. Use Case Specification: UC7 Risk Monitoring

Description

Describe the functionality

Screenshots

Insert screenshots and shortly explain what can be seen

FIG. 3.13.: *Use Case X: Detail*

Basic Flow

Describe the most common path through this use case

Activity Diagram

FIG. 3.14.: *Activity Diagram Use Case X*

Alternative Flows

What can go wrong :D

Special Requirements and Preconditions

Where does the user come from, what does he have to do before he gets here

Postconditions and Persistance

What has changed and how do we make sure the change persists?

3.2.2.9. Use Case Specification: UC8 Risk Response Management

Description

Responses are ways to react to a risk. A new response can be added to a risk from the risk detail page. A response contains a name, a type and a description as well as the information whether the described steps have already been undertaken or not. Depending on the Type of risk a reminder can be set to remind the risk owner to repeat the response tasks.
The risk owner can remove a response on a project risk if the project manager concurs.
If the risk is a pool risk the response can also be a pool response. Pool responses can only be removed if all project managers who currently reference the risk concur. However, pool responses can be deactivated for the current project with only the current project manager agreeing to prevent situationally unsuitable response reminders.
A new response to a pool risk can be turned into a pool response via a voting process among team members. This is a CRUD use case.

Screenshots

Insert screenshots and shortly explain what can be seen

FIG. 3.15.: *Use Case X: Detail*

Basic Flow

Creating a response:

- When the user clicks the "+" button at the risk detail page in the response section.
- Then the screen for adding a new response is opened.
- When the response form is filled by the user.
- And the user clicks on the "Add response" button.
- Then the risk is synced with the server.

Reading a response:

- The user is on the risk detail site for a project risk.
- By clicking on a response a detail view is expanded.
- For exiting the detail view a minimize button is clicked.

Updating a response:

- The user is on the risk detail site for a project risk.
- In the risk response section there is a pen button, which opens an editing form.
- By clicking the "Save" button the risk owner is notified of the changes.
- If the owner concurrs the changes are syncronized with the server.

Deleting a response:

- The user is on the risk detail site for a project risks.
- By clicking a "Delete" button the project manager receives a notification.
- If the project manager concurrs the response is deleted.

Activity Diagram

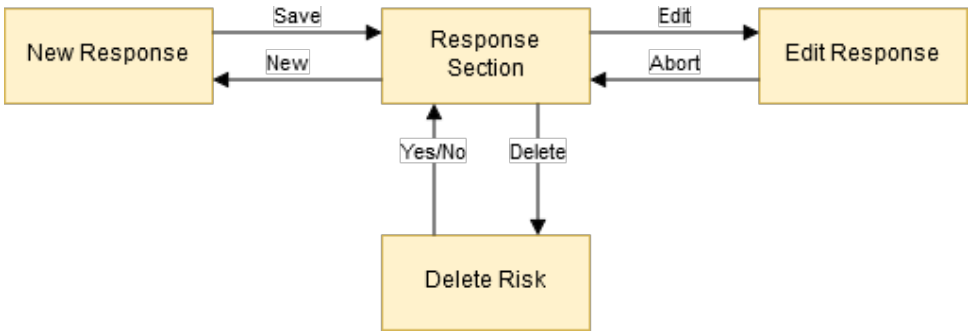

FIG. 3.16.: *Activity Diagram Use Case 8*

Alternative Flows

In addition to the crud functionalities responses attached to a risk that references a pool risk can be nominated for becoming a pool response and thus being attached to pool risk permanently. The voting process is the same as for pool risk nomination described in UC 53.2.2.6. The update and delete flows change if the response is a pool response attached to a pool risk.

- Updating a pool response requires project manager concurrance.
- Deleting a pool response will trigger a voting process as described in UC 53.2.2.6. However it will be limited to project managers to prevent users from deleting unpleasant responses.

Special Requirements and Preconditions

The preconditions for this use case are:

1. A risk exists.
2. The user is member of the project containing the risk.
3. Deleting requires the user to also be the owner of the risk.

For pool responses the following preconditions replace the above described:

1. The response is attached to a pool risk.

Postconditions and Persistance

Changes are directly reflected into the database via corresponding POST, PUT and DELETE requests.

3.2.2.10. Use Case Specification: UC9 Project Initialization

Description
Describe the functionality

Screenshots
Insert screenshots and shortly explain what can be seen

FIG. 3.17.: *Use Case X: Detail*

Basic Flow

Describe the most common path through this use case

Activity Diagram

FIG. 3.18.: *Activity Diagram Use Case X*

Alternative Flows
What can go wrong :D

Special Requirements and Preconditions
Where does the user come from, what does he have to do before he gets here

Postconditions and Persistance
What has changed and how do we make sure the change persists?

3.2.2.11. Use Case Specification: UC10 Activity Stream

Description
A user should be able to see the latest activity (see chapter 2.2.3) in the given enviroment.

Screenshots
Insert screenshots and shortly explain what can be seen

FIG. 3.19.: *Use Case X: Detail*

Basic Flow

Activity Stream:

- The user opens the home page where the activity stream can be found.
- Within the activity stream the user can then find the latest activities (examples mentioned above).
- Depending on the activity the user has the chance to interact with it and will be redirected to the specific item (e.g. a user can click on a new risks shown in his activity stream to open it).

Activity Diagram

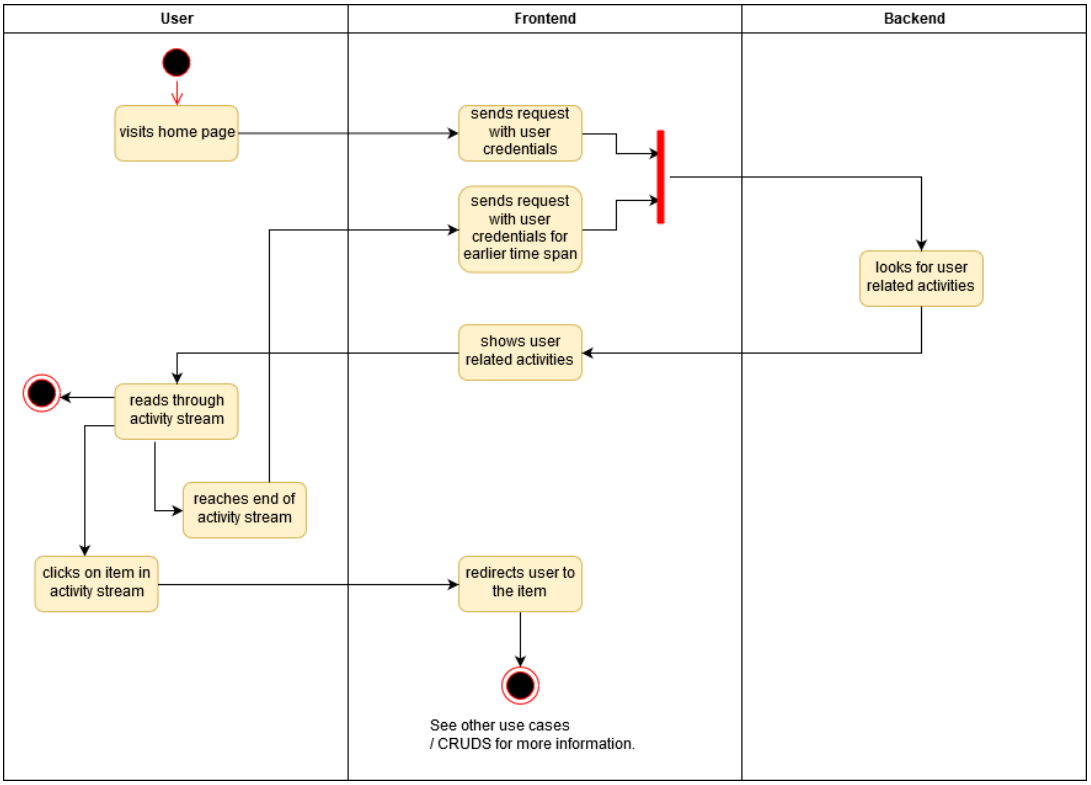

FIG. 3.20.: *Activity Diagram UC10 Activity Stream*

Alternative Flows

N/A

Special Requirements and Preconditions

As the activity contains personalized information, the following preconditions have to be fulfilled.

1. The user has to be logged in.
2. Alternatively a visitor can login to be redirect to the personal home page.

Postconditions and Persistance

N/A

3.2.2.12. Use Case Specification: UC11 Progress Indicator

Description

Describe the functionality

Screenshots

Insert screenshots and shortly explain what can be seen

FIG. 3.21.: *Use Case X: Detail*

Basic Flow

Describe the most common path through this use case

Activity Diagram

FIG. 3.22.: *Activity Diagram Use Case X*

Alternative Flows

What can go wrong :D

Special Requirements and Preconditions

Where does the user come from, what does he have to do before he gets here

Postconditions and Persistance

What has changed and how do we make sure the change persists?

TODO: INSERT USE CASES CONTEXT

3.2.3. Requirements

Couldn't find a better translation for "Anspruch" :/ Describes which 'Ansprüche' we have regarding the following categories:

3.2.3.1. Usability

3.2.3.2. Reliability

3.2.3.3. Performance

3.2.3.4. Supportability

3.2.4. Design Constraints

Where does our software run? Where does it not? Which functions cannot be covered and why? Which conditions are required to use the software? Stuff like that

3.2.5. Interfaces

3.2.5.1. User Interfaces

Describe the views the software will have

3.2.5.2. Further Interfaces

Software, Hardware and Communcation Interfaces... expand subsubsections if necessary

3.3. Architecture

In this chapter the used architectural patterns, technologies, frameworks and libraries are briefly explained. First, an overall composition is given whose individual layers will then be explicitly described.

3.3.1. Overall composition

Based on the Model-View-Controller (Model View Controller (MVC)) architectural pattern the application is split into three major parts: the frontend, backend and the persistence layer. As seen in figure 3.23 the frontend is in charge of the view and the backend of the model and controller.

In terms of the MVC pattern, the view / frontend is a presentational layer accessible by the user – traditionally a user interface. The model is equivalent to the data of an application which can be persisted with any appropriate technology (e.g. a database). The data itself can be accessed, updated or deleted by the controller. Furthermore, the controller is in charge of the logic of the application [29, p. 7].

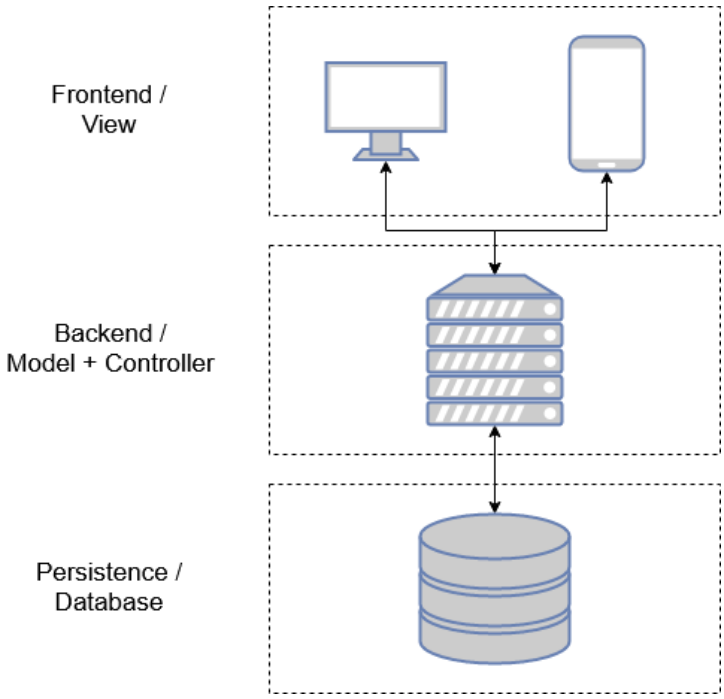

FIG. 3.23.: *Overall Composition*
[own representation]

3.3.2. Frontend / View

The view of the application is primarly developed with React (Version 16.8.5, 19/11/2019) and Redux (Version 6.0.1, 19/11/2019).

React is a JavaScript library written by Facebook Inc. for developing user interfaces (typically web apps). The major benefits from using React are components and their states. Components (e.g. an alert) are written in JavaScript XML (JSX) and reusable which means when written once they can be reused anywhere else in the user interface, thus duplicate source code can be avoided. In React a component can hold its own state (e.g. the alert message or alert duration) which can be used for the user interface. If the state of a component changes and is involved in the user interface React automatically updates the user interface with the new state, making the development of the user interface easier as elements do not have to be addressed manually and then adjusted with the new content [30, p. 7-8].

Even though components can pass their states in hierarchical order, having a central store for the state of the components has several benefits as simplifying the overall project structure and easier testing. Redux is a JavaScript library providing such a store and introducing further terms described in the following list:

- **Action**: An action typically leads to changes in the store but does not directly adjusts it. In fact, an action is responsible for fetching data from any resource (e.g. via the controller of the backend). The collected data will then be passed to a corresponding reducer.
- **Reducer**: A reducer contains the business logic about how exactly the data from the action should be saved to the store (e.g. filtering the action for only necessary information).
- **Store**: The store contains the state of the application and can be accessed by React components [31, p. 531-534].

Figure 3.24 visually explains the Redux pattern.

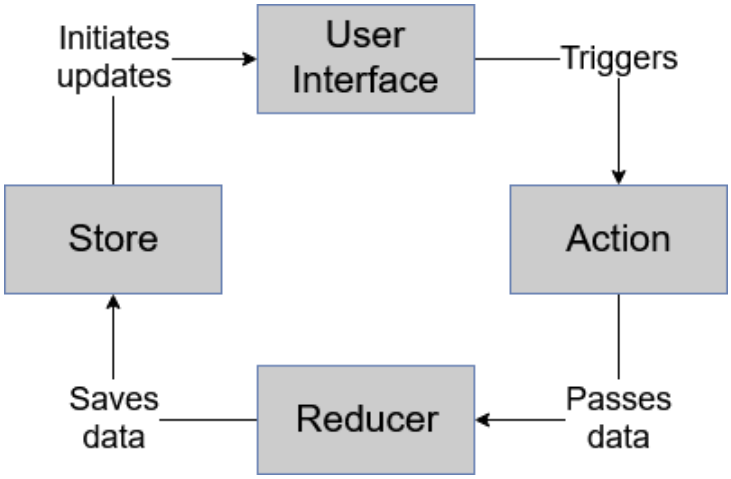

FIG. 3.24.: *Redux Pattern*
[own representation]

3.3.3. Backend / Model + Controller

The controller and model are mainly implemented with the help of Spring Boot 2 (Version 2.1.9, 20/11/2019) and the underlaying Spring Java framework (Version 5.1.10, 20/11/2019)

developed by Pivotal Software. JHipster (Version 6.5.1, 20/11/2019), a tool for generating Spring Boot and React applications, is used for a fundamental setup.

The Spring Java framework is a popular framework to develop a variety of web applications with covering the needs for security and data persistence. As the needs of business applications are constantly growing, the Spring frameworks configuration became more complex [32, p. 1]. With focus on extensibility and autoconfiguration Spring Boot was developed. The autoconfiguration is achieved by providing opinionated defaults whereas the extensibility is enabled by the possibility of adding starter modules such as Web, Data-Java Persistence API (JPA) or Security [32, p. 21-22].

All of them are used in the risk management application for the following purposes:

- **Starter-Web**: This starter modules includes the Spring MVC Java web framework providing support for the MVC pattern and Representational State Transfer (REST) Application Programming Interfaces (API)s. For example, a Java class can be annotated as controller and methods within define what kind of Hypertext Transfer Protocol (HTTP) methods are allowed as well as the REST endpoint the controller can be addressed at [32, p. 107-109].

- **Starter-Data-JPA**: The Starter-Data-JPA module allows developers to access databases in their Spring Boot application without writing Structured Query Lanuage (SQL) statements. Instead an object oriented API can be used and thus developers don't have to switch between different languages. An important part of this module is Hibernate, a technology which can map Java objects to a database. Hence the whole model of the application can be realized as Java classes and the technologies in this starter module take care of the persistence to a database [32, p. 83].

- **Starter-Security**: As not everyone should be able to access the same information or perform the same actions via the controllers (e.g. project related risks, deleting projects) the Starter-Security module enables security features such as authentication against the REST endpoints or user specific roles like being an average user or application admin [32, p. 176-176].

Figure 3.25 clarifies the functional interaction between all Spring Boot starter modules. In the case of the risk management application requests to REST endpoints, implemented with features provided by the Starter-Web module, are performed by Redux actions (see chapter 3.3.2). If the controller behind the REST endpoint has restrictions to specific roles, specified with the help of the Starter-Security module, the roles are gathered from the user by the credentials which are sent with the request and if they match, the logic of the controller gets executed. In this example data should be retrieved which can be the gathered with the help of the object oriented JPA included in the Starter-Data-JPA module and then be returned to the frontend and be used for further purposes.

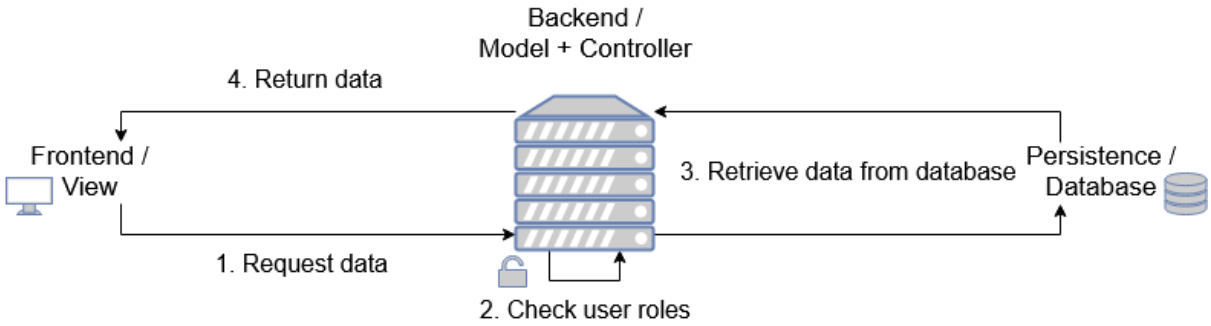

FIG. 3.25.: *Functional interaction of Spring Boot starter modules*
[own representation]

3.3.4. Persistence / Database

For persistence a relational PostgreSQL (Version 9.5, 20/11/2019) database is used and can be accessed by the backend.

The creation of necessary tables and relations is being done automatically by the technologies within the Starter-Data-JPA module. For further information about the Starter-Data-JPA module see chapter 3.3.3.

TODO: Add whole project UML here someday?

3.4. Gamification concept TBD

3.4.1. Player Personas

Player Personas based on survey -> Player Centered Design https://www.interaction-design.org/literature/boo
at-work-designing-engaging-business-software/chapter-3-58-player

3.4.2. Mission

Mission

3.4.3. Motivation + Mechanics

not only gamification patterns, but also basic motivational patterns => concrete conception of used patterns

3.4.4. Evaluation

Evaluation

4. Implementation

4.1. Unterkapitel -> Design, Evaluation, Methodisches, PM, ...

4.2. Unterkapitel2

5. Discussion

6. Conclusion and Outlook

List of references

[1] Yogesh K. Dwivedi et al. "Research on Information Systems Failures and Successes: Status Update and Future Directions". In: *Information Systems Frontiers* 17.1 (Feb. 1, 2015), pp. 143–157. ISSN: 1572-9419. DOI: 10.1007/s10796-014-9500-y. URL: https://doi.org/10.1007/s10796-014-9500-y.

[2] Karel de Bakker, Albert Boonstra, and Hans Wortmann. "Does Risk Management Contribute to IT Project Success? A Meta-Analysis of Empirical Evidence". In: *International Journal of Project Management* 28.5 (2010), pp. 493–503. ISSN: 0263-7863. DOI: https://doi.org/10.1016/j.ijproman.2009.07.002. URL: http://www.sciencedirect.com/science/article/pii/S0263786309000787.

[3] Ursula Kusay-Merkle. *Agiles Projektmanagement Im Berufsalltag: Für Mittlere Und Kleine Projekte*. Jan. 2018. ISBN: 978-3-662-56799-9. DOI: 10.1007/978-3-662-56800-2.

[4] Kalle Lyytinen and Rudy Hirschheim. "Information Systems Failures – a Survey and Classification of the Empirical Literature". In: *Oxford Surveys in Information Technology* 4 (Jan. 1988), pp. 257–309.

[5] Sandeep Gupta et al. "Systematic Literature Review of Project Failures: Current Trends and Scope for Future Research". In: *Computers & Industrial Engineering* 127 (Dec. 2018). DOI: 10.1016/j.cie.2018.12.002.

[6] Shareeful Islam. "Software Development Risk Management Model – a Goal-Driven Approach". Feb. 2011. DOI: 10.1145/1595782.1595785.

[7] Kaitlynn M. Whitney and Charles B. Daniels. "The Root Cause of Failure in Complex IT Projects: Complexity Itself". In: *Procedia Computer Science* 20 (2013). Complex Adaptive Systems, pp. 325–330. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.09.280. URL: http://www.sciencedirect.com/science/article/pii/S1877050913010806.

[8] Debbie Tesch, Timothy Kloppenborg, and Mark Erolick. "IT Project Risk Factors: The Project Management Professionals Perspective". In: *The Journal of Computer Information Systems* 47 (June 2007).

[9] Otniel Didraga. "The Role and the Effects of Risk Management in IT Projects Success". In: *Informatica Economica* 17 (Mar. 2013), pp. 86–98. DOI: 10.12948/issn14531305/17.1.2013.08.

[10] Joana Peixoto, Anabela Tereso, Gabriela Fernandes, and Rui Almeida. "Project Risk Management Methodology: A Case Study of an Electric Energy Organization". In: *Procedia Technology* 16 (Dec. 2014), pp. 1096–1105. DOI: 10.1016/j.protcy.2014.10.124.

[11] Y.H. Kwak and J. Stoddard. "Project Risk Management: Lessons Learned from Software Development Environment". In: *Technovation* 24.11 (2004), pp. 915–920. ISSN: 0166-4972. DOI: https://doi.org/10.1016/S0166-4972(03)00033-6. URL: http://www.sciencedirect.com/science/article/pii/S0166497203000336.

[12] Roque Junior and Marly Carvalho. "Understanding the Impact of Project Risk Management on Project Performance: An Empirical Study". In: *Journal of technology management & innovation* 8 (Feb. 2013), pp. 6–6. DOI: 10.4067/S0718-27242013000300006.

[13] A. A. Keshlaf and K. Hashim. "A Model and Prototype Tool to Manage Software Risks". In: *Proceedings First Asia-Pacific Conference on Quality Software*. Oct. 2000, pp. 297–305. DOI: 10.1109/APAQ.2000.883803.

[14] Janaki Kumar and Mario Herger. *Gamification at Work: Designing Engaging Business Software*. 1st ed. Denmark: The Interaction Design Foundation, 2013. 157 pp. ISBN: 978-87-92964-06-9.

[15] Hans-Werner Bierhoff (editor) and Dieter Frey (editor). *Enzyklopädie der Psychologie: Soziale Motive und soziale Einstellungen - Sozialpsychologie 2*. 1st ed. Vol. 2. 3 vols. Die Enzyklopädie der Psychologie Themenbereich C, Serie VI. Göttingen: hogrefe, 2016. ISBN: 978-3-8444-0564-0.

[16] Edward Deci. "The Effects of Externally Mediated Rewards on Intrinsic Motivation". In: *Journal of Personality and Social Psychology* 18 (Apr. 1971), pp. 105–115. DOI: 10.1037/h0030644.

[17] Edward Deci and Richard Ryan. "Theories of Social Psychology: Self-Determination Theory". In: *Handbook of Theories of Social Psychology: Volume 1*. 1 vols. London: SAGE Publications Ltd, Oct. 26, 2019, pp. 416–436. ISBN: 978-0-85702-960-7. DOI: 10.4135/9781446249215.

[18] Chris Lewis. *Irresistible Apps : Motivational Design Patterns for Apps, Games, and Web-Based Communities*. Berkeley, CA: Apress, 2014. 179 pp. ISBN: 978-1-4302-6422-4.

[19] Burrhus Frederic Skinner. *The Behavior of Organisms*. 1st ed. New York: Academic Press, 1938. 457 pp. ISBN: 978-0-9964539-0-5.

[20] Travis Lowdermilk. *User-Centered Design : [A Developers Guide to Building User-Friendly Applications]*. 1. ed. Beijing: OReilly, 2013. 135 pp. ISBN: 1-4493-5980-9 978-1-4493-5980-5.

[21] Juho Hamari, Jonna Koivisto, and Harri Sarsa. "Does Gamification Work? - A Literature Review of Empirical Studies on Gamification". In: *Proceedings of the 47th Annual Hawaii International Conference on System Sciences, HICSS 2014*. IEEE COMPUTER SOCIETY PRESS, 2014, pp. 3025–3034. ISBN: 978-1-4799-2504-9. DOI: 10.1109/HICSS.2014.377.

[22] Johannes Müller-Salo, Annette Dufner, and Henry Sidgwick. *Henry sidgwick: der utilitarismus und die deutsche philosophie*. Hamburg: Felix Meiner, Jan. 2019. 255 pp. ISBN: 978-3-7873-2996-0.

[23] Judith Zaichkowsky. "The Personal Involvement Inventory: Reduction, Revision, and Application to Advertising". In: *Journal of Advertising* 23 (June 2013), pp. 59–70. DOI: 10.1080/00913367.1943.10673459.

[24] Dennis Sheppard. *Beginning Progressive Web App Development: Creating a Native App Experience on the Web*. OCLC: 1018305973. New York: Apress, 2017. 266 pp. ISBN: 978-1-4842-3090-9 978-1-4842-3089-3.

[25] Christian Liebel. *Progressive Web Apps: das Praxisbuch*. 1. Auflage. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2019. 518 pp. ISBN: 978-3-8362-6494-5.

[26] *Progressive Web Apps*. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (visited on 10/17/2019).

[27] Majid Hajian. *Progressive Web Apps with Angular: Create Responsive, Fast and Reliable PWAs Using Angular*. OCLC: 1103217873. 2019. ISBN: 978-1-4842-4448-7 978-1-4842-4449-4. URL: http://public.eblib.com/choice/PublicFullRecord.aspx?p=5780029 (visited on 10/17/2019).

[28] *How to Make PWAs Installable*. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs (visited on 10/17/2019).

[29] Ralph Steyer. *Webanwendungen mit ASP.NET MVC und Razor: Ein kompakter und praxisnaher Einstieg*. Springer-Verlag, July 13, 2017. 116 pp. ISBN: 978-3-658-18376-9. Google Books: 09gsDwAAQBAJ.

[30] Stoyan Stefanov. *Durchstarten mit React: Web-Apps einfach und modular entwickeln*. O'Reilly, Apr. 19, 2017. 266 pp. ISBN: 978-3-96010-091-1. Google Books: 0aV4DwAAQBAJ.

[31] Adam Freeman. *Pro React 16 / by Adam Freeman*. 2019. ISBN: 978-1-4842-4451-7.

[32] K. Siva Prasad Reddy. *Beginning Spring Boot 2: Applications and Microservices with the Spring Framework / by K. Siva Prasad Reddy*. 2017. ISBN: 978-1-4842-2931-6.

Appendix

A. Anhang1