Herbert Fuchs, BSc

Mobile Interactive Recommender Framework

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Denis Helic

Co-Supervisor

Dipl.Ing. Lukas Eberhard, Bsc

Institute of Interactive Systems and Data Science

Graz, February 2019

Affidavit

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

Date

Signature

Abstract

Nowadays, a large amount of information is made available to everyone in a wide variety of areas. In order to customize this information, recommendation systems have been developed. *rbz.io* is such a system, which creates personalized recommendation for movies, board games and video games by just telling the system what things you like or dislike. This system is currently only available through a bot on Reddit and must be addressed with a specific bot-language.

The aim of these Master thesis is to create a mobile application for the recommender engines of *rbz.io* and to make these engines available for everybody without knowing the specific bot-language. Additionally, the application should reach many people, therefore, the operating systems Android and iOS should be supported.

At the first part of the thesis, different technologies about mobile applications are described. Additionally, basic knowledge about web technology is provided and different frameworks for hybrid mobile development and API development are illustrated. At the second part, requirements and use cases are collected, which are decisive for the selection of the used technology. Afterwards, the basic architecture of the system is explained and each individual component is described. In particular, the API and the mobile application are shown in more detail. To get an insight of the page design of the mobile application, each page gets explained and illustrated separately. In the evaluation chapter, a case study will be conducted to assess the usability and to identify strengths and weaknesses of the mobile application. Finally, the result is analysed and the different outcomes are discussed.

Contents

List of Figures

1. Introduction

Nowadays, almost 2.71 billion people, which is almost 35.9% of the world's population, use a smartphone and this number will increase further in the coming years as illustrated in Figure 1.1 [14]. 99.9% of these smartphones use the operating system Android and iOS [18]. In order to reach as many users as possible, it is important to be able to provide an application for both operating systems. This can be achieved with different approaches. Either with a native application for each operating system, or with a hybrid application, which works for both operating systems.

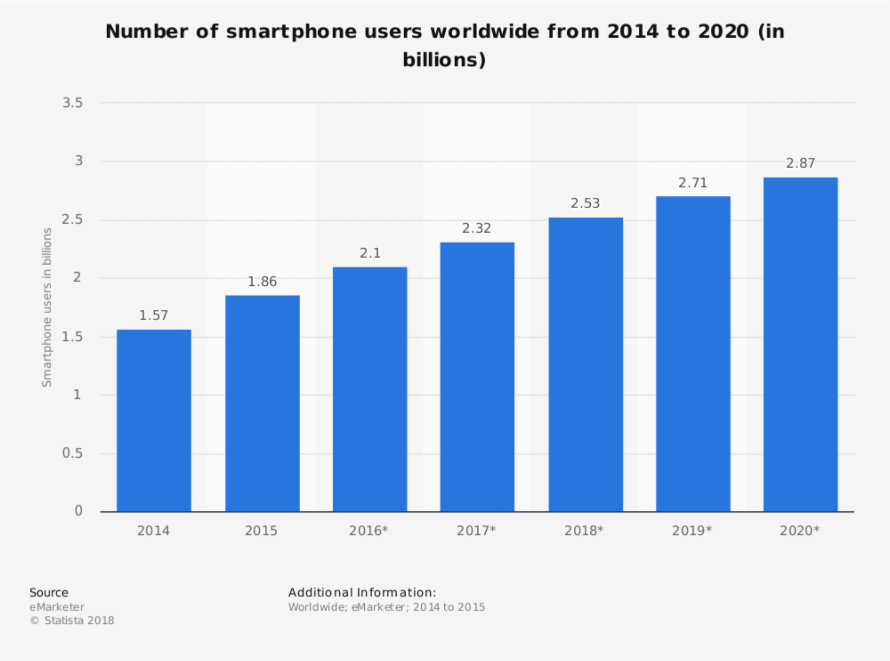


Figure 1.1.: Number of smartphone users worldwide from [14]

Another industry that is constantly growing is the entertainment industry [24]. Whether games or movies, the number of available products is steadily increasing. This makes it very difficult to have an overview of the available products. A solution for this problem are recommender systems. These systems recommend movies or games according to the personal preferences, and helps the user not to lose sight of the big picture. The big players like Netflix[1] and Amazon[2], have been using this technology for a while now to increase customer satisfaction by offering personalized recommendations.

In this master's thesis these two aspects are combined and a mobile application for movie, board game and video game recommendations is developed. This application uses the bot and the recommendation framework from a project of the *Institute of Interactive Systems and Data Science* (ISDS)[3] of the *Graz University of Technology Graz*[4] and is developed as a hybrid application with the Ionic Framework[5].

The implementation of the application and the associated systems are described in Chapter 4. The mobile application, as well as the API are described in detail. Chapter 2 provides the background for the used technologies. Furthermore, some related work is presented in this chapter. To understand the user's needs, use cases are defined in Chapter 3 and requirements are extracted from them. A case study was performed to evaluate the application. Details of this study are shown in Chapter 5.

1.1. Motivation

Several mobile applications for movie recommendations are available on the market. The main task of most applications is to provide an overview of movies or to search for specific movies and get information about them. The recommender function is usually only integrated as an additional feature. Many applications require that you first rate some movies in order to get a recommendation. This is often annoying, because you have to rate movies, which you probably do not know, to get a result. In most cases, exact specifications for the recommendations cannot be made. This leads to imprecise recommendations, which makes the user unhappy and as a consequence the user may delete the application.
For other domains like board games and video games, only a limited number

[1] https://netflix.com
[2] https://amazon.de
[3] https://isds.tugraz.at
[4] https://tugraz.at
[5] https://ionicframework.com

of mobile applications are available, although for video games there are more applications than for board games.

The ISDS provides recommendation engines called rbz.io[6] for the three different domains:

- Movies,
- Board Games,
- Video Games,

but they are only available through Reddit[7], by calling it with a specific bot-language[8] as illustrated in Listing 1.1.

```
/u/rbzio **m:Rain Man** **green lantern** **Thriller**
**g:Action** *comedy* **y:2005** *a:Cameron diaz*
**<2016** **a:leo dicaprio**
```

Listing 1.1: Example of the recommender engine bot-language to get a recommendation. These example includes 2 movies, 3 genres (two desired, one undesired), two actors (one desired, one undesired), a lower and an upper bound

The aim of these Master thesis is to create a mobile application for the provided recommender engines and to make them available for everybody without knowing the specific bot-language. All three domains should be available within one application and it should be easy to use.

[6] http://www.rbz.io

[7] https://reddit.com

[8] http://www.rbz.io/#botlanguage

2. Background and Related Work

This chapter describes basic technologies and different types of mobile applications. Especially web technologies and frameworks for hybrid application development, as well as the definition of REST APIs and its underlying frameworks are in focus. Furthermore, I show related work about mobile applications for recommendations and I illustrate some Ionic applications.

2.1. Types of mobile applications

Nowadays, different types of mobile applications exists. The appropriate type depends on how an application should work, how complex the development is, on which operation system the application should run, how skilled the developers are and much more.
In the following section I will give a description of the main types of mobile applications: native, hybrid and web applications. Later on I present a brief comparison between those types and related work.

2.1.1. Native Applications

Native applications are restricted to their platform. They are directly using the provided platform API. Therefore, Android applications can only be implemented with Java or Kotlin. iOS applications can only be created with Swift and Objective-C. Other platforms can be dropped, because according to the statistics of Gartner [18], almost all smartphones worldwide operate with Android (88%) or iOS (11.9%). Also the respective native software development kit (SDK) of these two systems are different. Android uses Android Studio as its SDK, whereas iOS uses Xcode [22]. Because of the direct API access, native applications are very powerful. GPS and other sensors, as well as the camera, the address book and other features of the operating system can be accessed and used with a high performance. Also computationally expensive advanced graphics and high performance operations can be processed efficiently. A big advantage of using native applications is the provided user interface of the platform, which everybody can use. This supports the creation of user-friendly applications, because the user usually already knows the interface element and can interpret it correctly. The main disadvantage of native applications occur if an application has to be developed for multiple platforms. Due to the non-existent reusability of the code, the application has to be developed multiple times. This causes longer development times, higher costs, and bigger maintenance effort.

2.1.2. Web Applications

Web applications are hosted on web servers and are accessible via a URL[1] in the web browser. They are developed in HTML5, CSS and Javascript. It is not necessary to install these applications. This is the reason why they are able to run on all different platforms. These applications try to act like a native application, but that is only possible for the view. Expensive operations and intensive graphical animations are not working as efficient as on native applications. The reason is that the application has only as much access to the resources as the used web browser. Besides, the application is only reachable if the device is online. It is also not possible to distribute this applications via an app store, since they are hosted and served like usual websites [25]. A big advantage of web applications is the easy maintainability, since the application can be updated on the server at any time and the user is obliged to use the new version. This prevents that a faulty version is used over a longer period of time.

2.1.3. Hybrid Applications

Hybrid applications combines the advantages of native- and web applications. This means that this kind of application type is mostly platform independent. The application uses the browser engine of the platform to render itself in a webview [2]. It is able to access platform features like sensors, the storage and device information. The complete set of native features is not available. One of the biggest development platform for hybrid applications to access native features is called Apache Cordova (See Section 2.3.1 for a detailed description) [15]. The used programming languages are HTML5, CSS and Javascript. Lots of different libraries are available and therefore the technology can be used

[1] Uniform Resource Locator

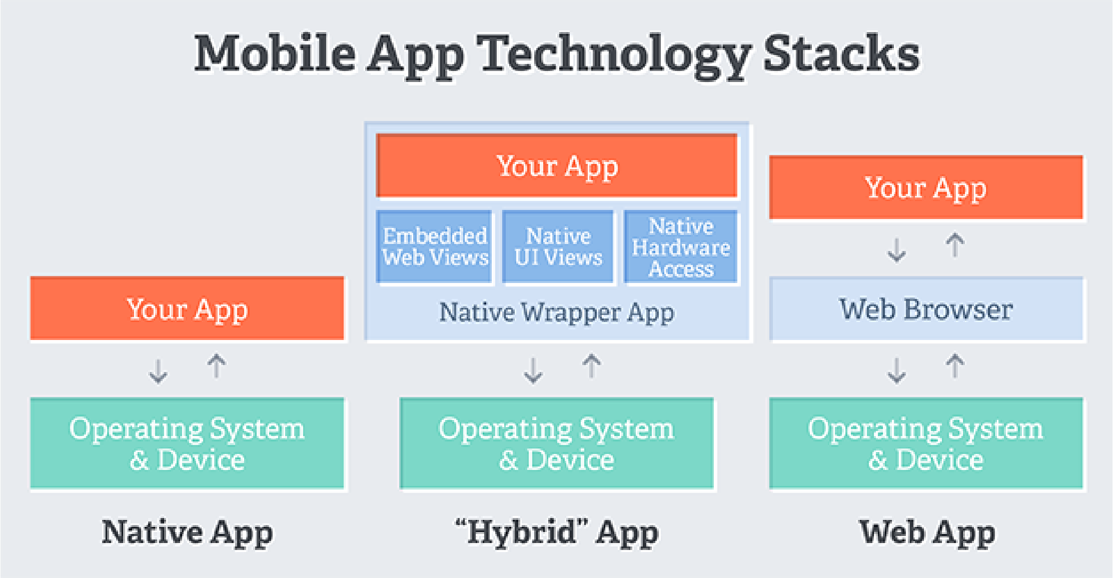


Figure 2.1.: Schematic architectural representation of the different types from [33]

in many different areas. As with web applications, computationally expensive operations and complex user interface animations are not performed as quick as on native applications. Applications of this type can be published in an app store and an installation file can be provided for a installation without a store.

2.1.4. Comparison between the different types

The main difference between the types is how the application uses the device. While native applications directly access the platform API, hybrid applications use the provided web view and web applications work via the provided browser. This architecture is schematically shown in Figure 2.1. For high performance operations like games, native applications are definitely recommended. If only data is displayed e.g. in newsreaders, web- or hybrid applications are sufficient. It should be determined whether the app should be provided in an app store or not. Another aspect is how often the application is used. If it is used more often, a hybrid or native solution is preferable. If an application is rarely used, the web approach is a better choice. A main factor of choosing a type is the cost. Native development is much more costly as developing a hybrid application. Table 2.1 shows the main differences according to the types. Based on these differences, I have selected the appropriate technologies for the application in this master thesis.

Aspects	Native Application	Hybrid Application	Web Application
Programming Language	Native only: Java or Kotlin - Android, Swift or Objective C-iOS	Web technologies-JavaScript HTML, CSS	Web technologies-JavaScript, HTML, CSS
User Interface (UX/UI Design)	Completely platform-specific UI. Native app design - rich, customizable, great visual effects and animation possible	Uses highly similar design interface to a native app. Cross-platform app design. Limited customization possibilities	Common for all platforms. Limited customization
Platforms	Native Android, Native iOS	Multiple	Multiple
App Stores Distribution	Google Play Store, Apple's App Store, Windows store	Google Play Store, Apple's App Store, Windows store (if guidelines met)	No stores
Performance	Faster based on embedded connection with OS and the device	Moderate response	Slower
App Ecosystem	SDKs and other tools for any technical implementation	Limited to the framework and to available 3rd party services	Limited
Feature Set	Wide access. Any device APIs used. Offers solutions for unique and specific features (VR, AR, IoT, etc.)	Moderate access. Some APIs are closed for hybrid mobile apps (e.g. gyroscope or accelerometer)	Limited. Some of the device APIs can be used (e.g. geolocation)
Navigation	App has embedded and intuitive navigation	WebView connects the web content with native app functioning	WebView connects the web content with native app functioning
Hardware Capabilities	Uses all capabilities of the mobile device	Less access to the device	Minimum access to device hardware
Development Costs	Higher than development for multiple platforms	Moderate	Minimum due to single code base
Development Timeline	Longer timeline	Moderate	Short
Expertise of Developers	High expertise (learn different languages)	Moderate expertise	Moderate expertise

Table 2.1.: Comparison between Native-, Hybrid- and Web Applications from [22]

2.1.5. Related Work

In this section I present mobile applications, which create personal recommendations for movies, video games and board games. There exists a variety of applications that provide personalized recommendations. A distinction must be made between applications in which the recommender function is the main task and those in which this is just a feature. The following applications show a selection of the most popular applications and their advantages and disadvantages.

IMDb

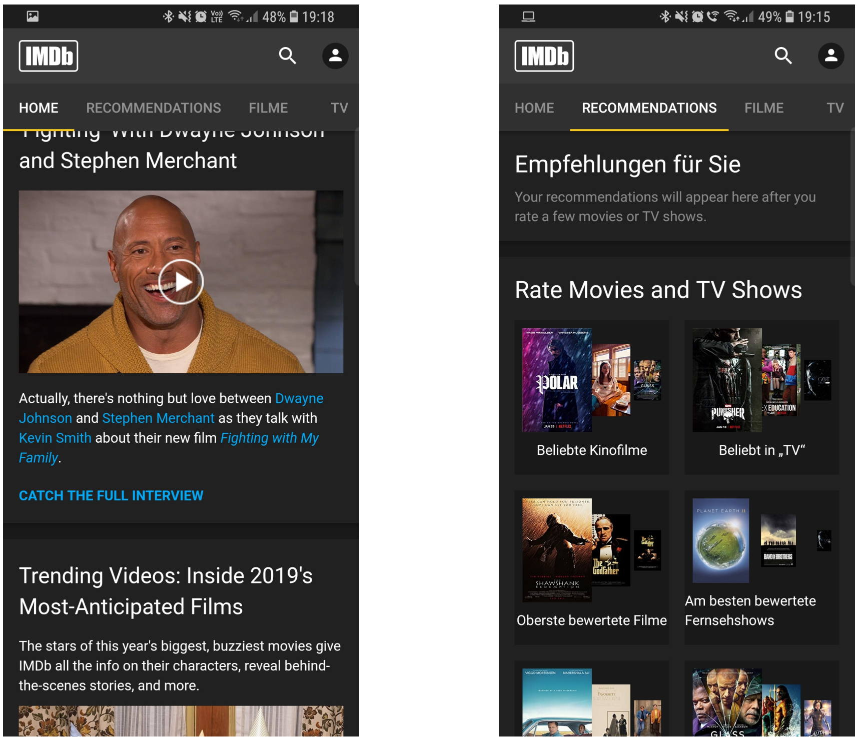
The mobile application of IMDb[2] mainly provides information about movies, actors and current news from the movie industry as shown in Figure 2.2. The recommendation is only implemented as a feature and it refers to movies that have already been rated. IMDb has over 100.000.000 downloads in the Google Play Store[3] and has a rating of 4,2. It is available for Android and iOS.

Advantages:
A major strength lies in the diversity of information about movies. Furthermore the application is clearly designed, which makes it very comfortable to use. It has a rich set of features, for example a rating list, a watchlist, a favourite cinema list and it is able to provide recommendations. But for these functionalities a user account is required.

Disadvantages:
As mentioned before, to use the functionalities, a user account is required. This makes the quick use of this application infeasible. In addition, it is not possible to create different movie requests, because the recommendations refers to the movie ratings. Furthermore the application only covers the movie domain.

[2] https://imdb.com

[3] https://play.google.com/store



Fighting With Dwayne Johnson and Stephen Merchant

Actually, there's nothing but love between Dwayne Johnson and Stephen Merchant as they talk with Kevin Smith about their new film *Fighting with My Family*.

CATCH THE FULL INTERVIEW

Trending Videos: Inside 2019's Most-Anticipated Films

The stars of this year's biggest, buzziest movies give IMDb all the info on their characters, reveal behind-the-scenes stories, and more.

Empfehlungen für Sie

Your recommendations will appear here after you rate a few movies or TV shows.

Rate Movies and TV Shows

(a) Landing page, which displays different information about the movie industry

(b) Recommendation feature, which is only accessible with a user account.

Figure 2.2.: Screenshot of the mobile application of IMDb

Movledge

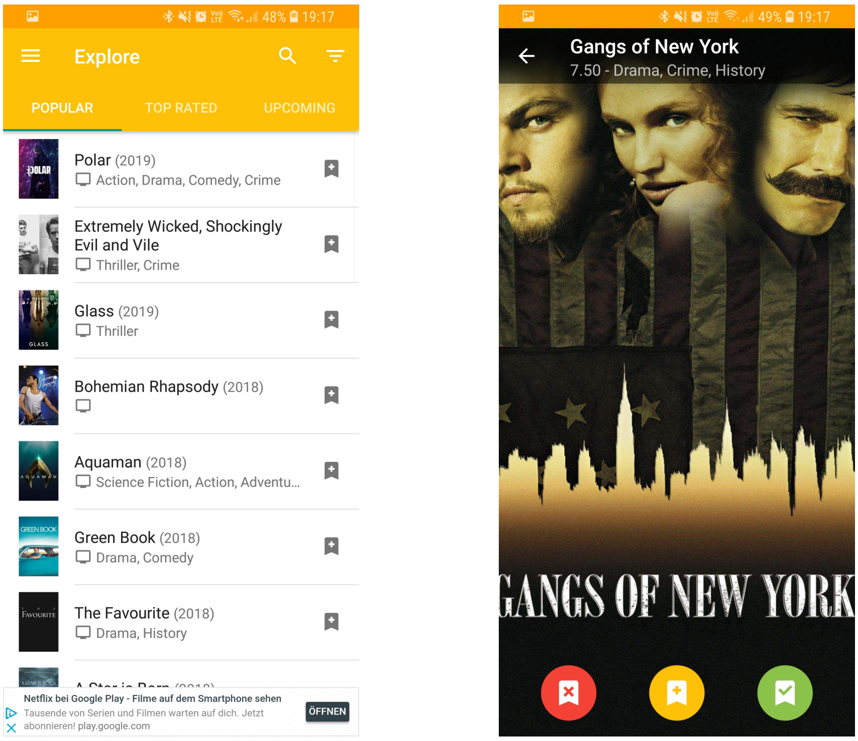
Movledge[4] is a free application to create personal movie collections and to browse through different movies as shown in Figure 2.3. It retrieves the data from IMDb and TMDb[5]. A recommendation functionality is provided as a feature which is based on the movie history of the user. Movledge has over 10.000 downloads in the Google Play Store and has a rating of 4,1. It is available for Android and iOS.

Advantages:
It provides different personal lists to categorize the films according to the

[4] https://movledge.com
[5] https://themoviedb.org

preferences. In addition, the user can find other users, and share these lists with them.

Disadvantages:
To use the application, a user account is required. Furthermore the recommendations are imprecise, because only the history of the user is taken into consideration. In addition only the film domain is supported.



48% 19:17

Explore

POPULAR TOP RATED UPCOMING

Polar (2019)
Action, Drama, Comedy, Crime

Extremely Wicked, Shockingly Evil and Vile
Thriller, Crime

Glass (2019)
Thriller

Bohemian Rhapsody (2018)

Aquaman (2018)
Science Fiction, Action, Adventu...

Green Book (2018)
Drama, Comedy

The Favourite (2018)
Drama, History

A Star is Born (2018)

(a) List of movies, which can be discovered.

49% 19:17

Gangs of New York
7.50 - Drama, Crime, History

(b) Recommendation feature, where the user can add the recommended movie to his personal list.

Figure 2.3.: Screenshot of the mobile application of Moveledge

Itcher

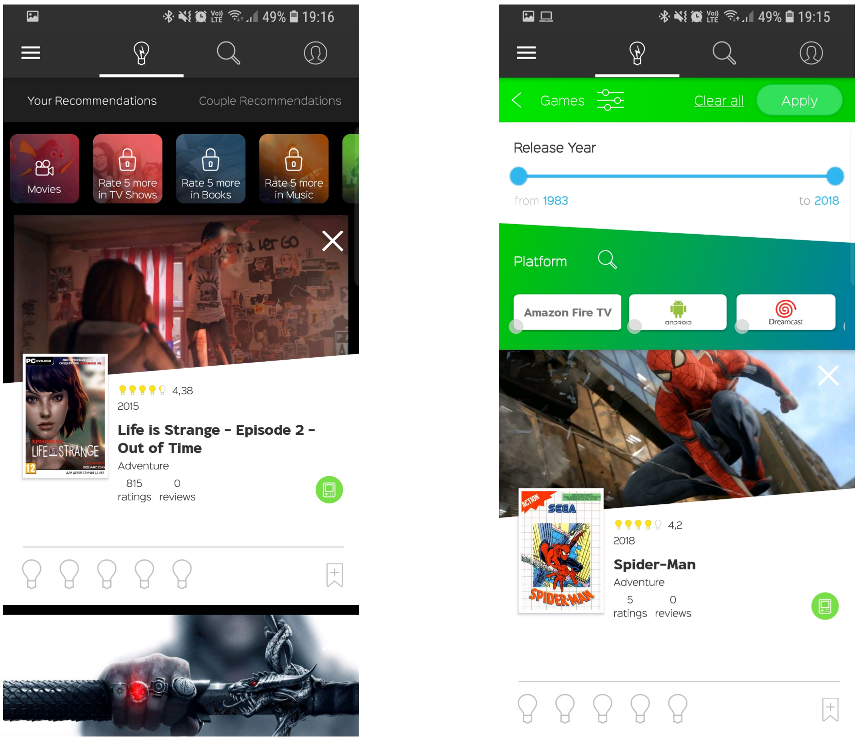
Itcher[6] is a recommendation applications for movies, tv shows, books, music and games. It creates recommendations after the user rated five entries of the domain. The user interface is clearly designed and makes a very structured

[6] http://itcher.com/

appearance as shown in Figure 2.4. Itcher has over 50.000 downloads in the Google Play Store and has a rating of 4,3. It is available for Android and iOS.

Advantages:
Each recommendation domain has its own filter to get a more personalized recommendation list, e.g. the movie domain can filter the list by the release year, genre and streaming provider. Each element in the recommendation list has a large amount of information and additionally it shows entries which are similar to it. This extends the recommendation list, but it is very well packaged and the user does not lose the overview. Additionally, it is possible to create couple- and group recommendations, which consider the ratings of each person.

Disadvantages:
The only disadvantage is that you have to create a user account in order to use the application.



(a) Start page of itcher. At the top of the page, the domain for the recommendation can be selected.

(b) Game recommendation with the filter functionality.

Figure 2.4.: Screenshot of the mobile application of Itcher

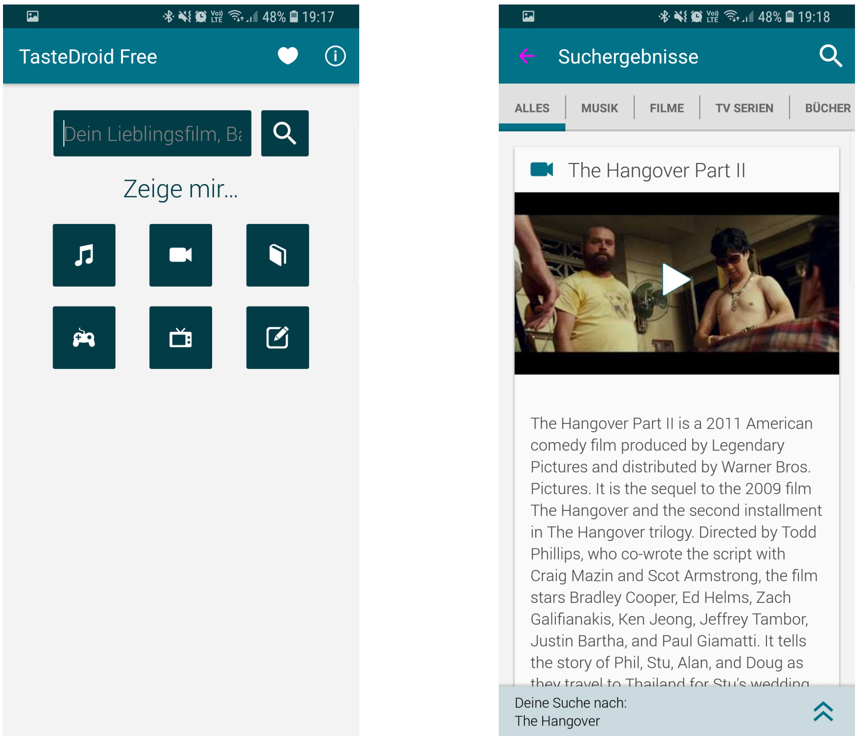
TasteDroid

TasteDroid[7] is a recommendation application for movies, music, books, authors and games for Android devices. It is illustrated in Figure 2.5. It has only 1000+ installations, but a rating of 4,0.

Advantages:
It provides a variety of domains and it is usable without a user account. It has a very simplistic user interface and it provides a mixed recommendation for the given search parameter. Furthermore, the result can be separated into the different domains, which is helpful to keep the overview.

Disadvantages:
It is only possible to specify one parameter for the computation of the recommendation. Therefore the recommendations are very general and not user related. Additionally, the automatic suggestions does not work, which makes the request creation quite hard.

[7] https://play.google.com/store/apps/details?id=com.muetzenflo.tastedroid



TasteDroid Free

Dein Lieblingsfilm, B

Zeige mir...

Suchergebnisse

ALLES | MUSIK | FILME | TV SERIEN | BÜCHER

The Hangover Part II

The Hangover Part II is a 2011 American comedy film produced by Legendary Pictures and distributed by Warner Bros. Pictures. It is the sequel to the 2009 film The Hangover and the second installment in The Hangover trilogy. Directed by Todd Phillips, who co-wrote the script with Craig Mazin and Scot Armstrong, the film stars Bradley Cooper, Ed Helms, Zach Galifianakis, Ken Jeong, Jeffrey Tambor, Justin Bartha, and Paul Giamatti. It tells the story of Phil, Stu, Alan, and Doug as they travel to Thailand for Stu's wedding

Deine Suche nach:
The Hangover

(a) Start page of TasteDroid, which is very simplistic.

(b) Recommendation list, which displayes the recommendation for all domains.

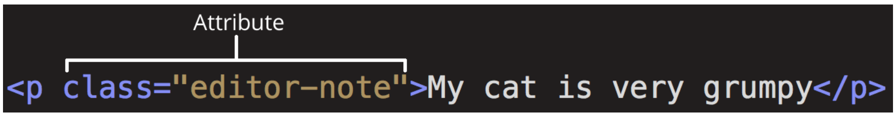
Figure 2.5.: Screenshot of the mobile application of TasteDroid

2.2. Web Technologies

Web technologies are used to give web pages a structure and a design. In addition, they ensure that interactive elements are offered in order to increase the usability. This section provides a short overview of the core web technologies.

2.2.1. Hypertext Markup Language - HTML

HTML is a standard markup language that defines the structure of web pages and web applications. It describes how the content of a page should be interpreted using HTML elements. These elements are designed as follows: as Figure 2.6 shows, the element starts with an opening tag, which consists of the element name (<p>). This tag can contain one or more attributes like id, class or style.



```
<p class="editor-note">My cat is very grumpy</p>
```

Figure 2.6.: Anatomy of an HTML element from [27]

The element ends with a closing tag, which is the same as the start tag, except that it includes a forward slash before the name (</p>). Between the tags, an arbitrary content can be inserted [27]. HTML supports nesting, so the content can be another HTML element.

The newest version of HTML is called HTML5. It reduces the need of external plugins, provides more elements and it has a better error handling. Furthermore the development has been simplified. As illustrated in Figure 2.7, data can now be stored directly on the mobile device or the browser using a local storage. This storage removes the need of Cookies and the security increases, because not every server request includes data. Besides, videos can be directly embedded and audio can be played in a web page without a plugin [30, 20].

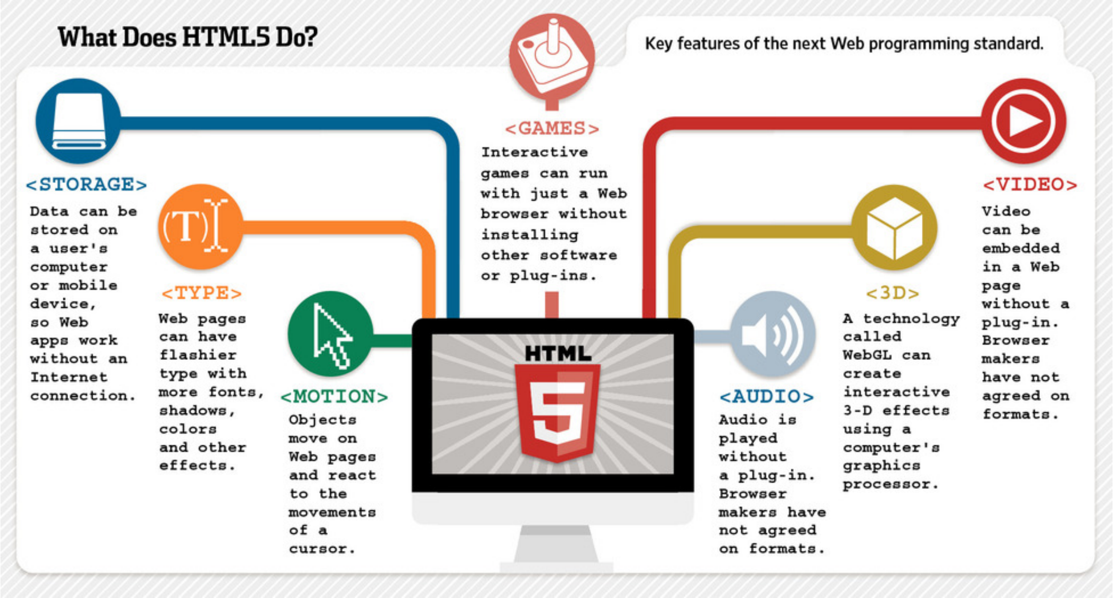
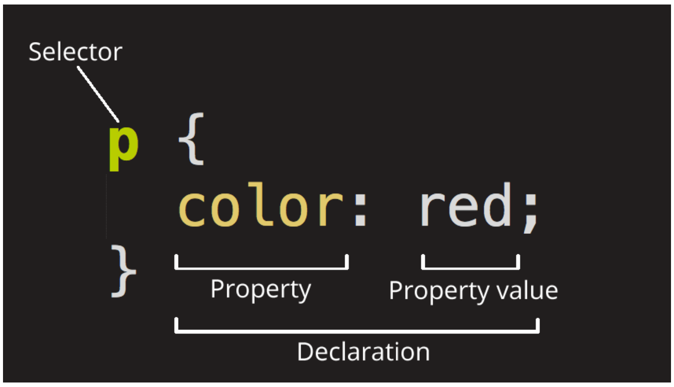

What Does HTML5 Do?
Key features of the next Web programming standard.
<STORAGE>
Data can be stored on a user's computer or mobile device, so Web apps work without an Internet connection.
<TYPE>
Web pages can have flashier type with more fonts, shadows, colors and other effects.
<MOTION>
Objects move on Web pages and react to the movements of a cursor.
<GAMES>
Interactive games can run with just a Web browser without installing other software or plug-ins.
HTML
5
<AUDIO>
Audio is played without a plug-in. Browser makers have not agreed on formats.
<3D>
A technology called WebGL can create interactive 3-D effects using a computer's graphics processor.
<VIDEO>
Video can be embedded in a Web page without a plug-in. Browser makers have not agreed on formats.

Figure 2.7.: Illustrates new key features of HTML5 from [] which should make the development easierUSC2019

2.2.2. Cascading Style Sheets - CSS

CSS is a style sheet language, which gives the possibility, to define code for changing the presentation of HTML web pages. It works directly with HTML, which means that the HTML-file must contain a reference to the used CSS file. The structure of an CSS ruleset is shown in Figure 2.8. Every set starts with a selector. There are different types of selectors, illustrated with an example in Table 2.2. After the selector, the declaration is wrapped in curved braces. It is specifying which of the element's properties are getting styled. The properties are depending on the HTML element. Each HTML element can have different properties. The big advantage of CSS is that multiple HTML pages can be styled centralized within one file [9].



```
p {
  color: red;
}
```

Figure 2.8.: Anatomy of a CSS ruleset from [9]

Selector name	**What does it select**	**Example Selector**	**Selected element**
Element selector	All HTML element(s) of the specified type.	**p**	<p>
ID selector	The element on the page with the specified ID. On a given HTML page, you're only allowed one element per ID (and of course one ID per element).	**#my-id**	<p id="my-id"> <a id="my-id">
Class selector	The element(s) on the page with the specified class (multiple class instances can appear on a page).	**.my-class**	<p class="my-class"> <a class="my-class">
Attribute selector	The element(s) on the page with the specified attribute.	**img[src]**	<img src="myimg.png"> but not <img>
Pseudo-class selector	The specified element(s), but only when in the specified state (e.g. being hovered over).	**a:hover**	<a> only when the mouse pointer is hovering over the link.

Table 2.2.: Different types of selectors with an given example from [9]

2.2.3. JavaScript

JavaScript, abbreviated as JS, is a powerful, high level, interpreted programming language which allows web pages and web applications to have interactive elements. It is defined inside a script-tag in an HTML page. JS does not require any special preparation or compilation before execution. That is a big difference to other programming languages, for example JAVA. JavaScript is not only runnable in browsers, it is also running on servers, if a JavaScript-engine is installed. This gives the possibility to create powerful APIs with this language. JavaScript has its strengths in making static websites dynamic. Some examples for that are:

- **Animations**: rotating, fading and moving of elements.
- **Interactive content**: games, videos, audio.
- **Validations**: validate forms, for example checking syntax of an email address.
- **User tracking**: analyse user behavior and sent it to the server to create personalized adds

JavaScript has a rich portfolio of third-party libraries. This makes the language usable in many fields. These libraries enables JS to easily create single page applications, to manipulate the Domain Object Model (DOM), or to create a simple asynchronous connection to a server [21].

2.3. Hybrid Mobile Application Frameworks

This section explains the Apache Cordova[8] framework which enables hybrid mobile applications to access native resources. Furthermore, I am discussing the Ionic framework in more detail. Since Ionic integrates Angular to become more powerful, I will give a brief overview of Angular the end of this section.

2.3.1. Apache Cordova

This open source framework is used for hybrid mobile application development. It enables the usage of standard web technologies (HTML, CSS, JavaScript), which makes it easier to develop applications for multiple platforms. With this framework, web applications can access native resources over plugins and the standard API. Figure 2.9 illustrates the architecture of a Apache Cordova application. The web app is the place, where the code for the application, which is mostly a web page, is located. In the WebView this application code gets executed and it combines the web page with native application components. These components are reached by the Cordova plugins. Cordova provides a rich set of core plugins, which make access to the camera, contacts, battery

[8] https://cordova.apache.org

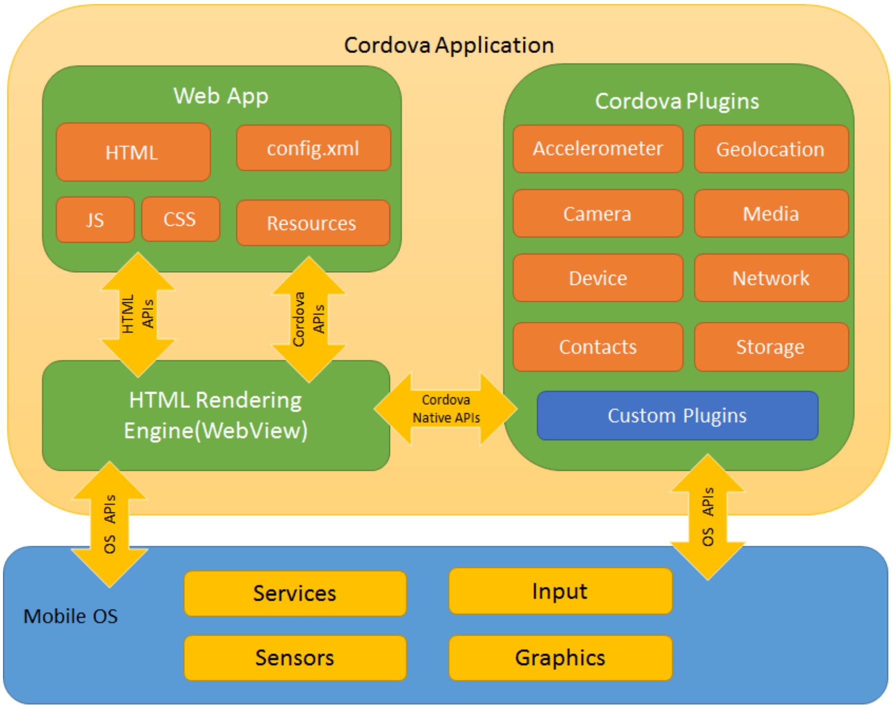


Figure 2.9.: Architecture of a hybrid mobile application with Apache Cordova from [15]

and more available. There exists several third-party plugins for specific native components which are not available on all different platforms. This fact turns Cordova into a universal framework for any kind of platform [15].

2.3.2. Ionic Framework

Ionic Framework is a user interface (UI) toolkit, which uses web technologies to create performant, high-quality web- and mobile applications. It is an open source project, released under the permissable MIT license. The newest version is currently v4, which is still in the beta-phase. The goal of this framework is to build applications on multiple platforms with one code base. Also it provides the user a native user experience. Clean and simple user interfaces and a rich set of different functionality are the main advantages of this framework. Ionic uses Apache Cordova to access native resources. To get a more powerful set of functions, Ionic is integrating Angular. The following listing shows the core concepts of the framework:

- **UI Components**: Ionic provides a big set of predefined UI components, which are reusable and highly customizable. Hence it is possible to create an own and unique style in the application.

- **Platform Continuity**: Ionic allows the developer to use the same code for multiple platforms. Each component looks different depending on the platform, which means that the same components can look different on various platforms. The reason for that is, that Ionic takes the theme of the platform to generate a look which is as similar as the look of an native application. For example for Android devices, Ionic takes the Google's design language Material Design. For iOS devices, Ionic uses Apple's own iOS design language.
- **Navigation**: Ionic provides parallel navigation histories. This allows the user a better navigation through the app, because there is no static navigation (back- and forward button) like on conventional web pages. It is possible to create routes between each pages.
- **Native Access**: The same code base work on many platforms like desktop computers, phones and tablets. Cordova enable the application to use native resources,which enlarges the operating range.
- **Theming**: Ionic uses CSS to style the application. It centralizes the color-theming in one file using CSS variables.

Ionic supports the mobile operating systems Android 4.4+ and iOS 10+, as well as desktop browsers like Chrome, Safari, Edge and Firefox [16].

2.3.3. Angular

Angular[9] is an open source front-end web application framework based on TypeScript. The newest version of Angular was published in October 2018 under version 7. It is used to create single-page applications. Figure 2.10 illustrates the basic architecture of an Angular application. In the following list I show each part of the architecture described by Malik [26]:

- **Modules**: Modules structure the application. They link its components with related code, such as services. The root module provides the bootstrap mechanism to launch the application.
- **Components**: A component is a class with some attached metadata and it includes the application- and data logic. Every Angular application has one component called root component. The metadata of a component defines the view. It is usually an HTML template.

[9] https://angular.io

- **Templates**: A template combines HTML markups with Angular markups. Angular markups are able to modify HTML elements before they are getting displayed, for example hide or show a specific element.
- **Metadata**: By attaching a decorator, the metadata can configure the expected behaviour of a class.
- **Data binding**: Defines the communication between the application data and the DOM. There are two different kinds of bindings:
  - Event Binding: Application data gets updated by user input.
  - Property Binding: HTML template gets updated with values computed by the application.
- **Directives**: They provide logic for the templates. Before displaying the view, Angular checks the directives (program data and logic) and modify the DOM and HTML elements according to this.
- **Services**: These are classes, which provide data and functions that are shared across components.
- **Dependency Injection**: It enables component classes to be efficient an lean, by delegating tasks such as fetching data or validate user input to services.

Another key concept of Angular is routing, which creates a navigation path between many application states and view hierarchies. Furthermore routing enables lazy loading. This allows the user to run through the application very smoothly, by only loading the displayed resources [1].

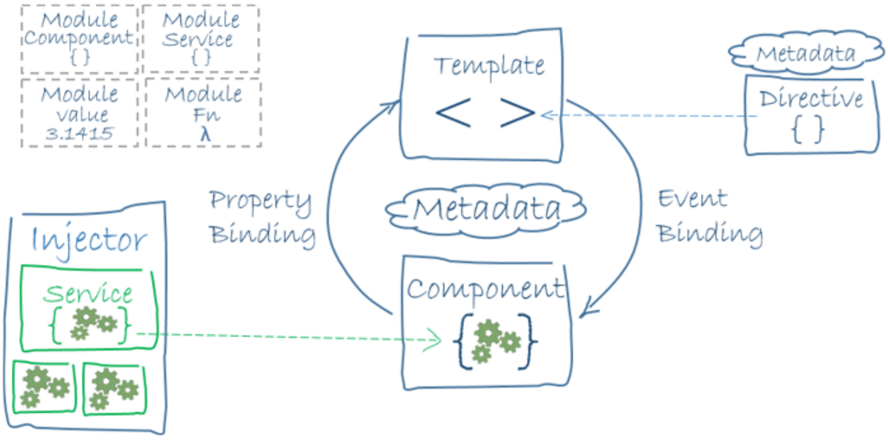


Figure 2.10.: Architecture of Angular from [1]

2.3.4. Related Work

Ionic provides a showcase[10] of the most beautiful applications in all different areas. In the following list I will give a brief overview of the different applications.

- **JustWatch[11]**: It is a movie guide, where the user can easily navigates through his favourite movies. It can be filtered by platform, on which the movies are available for viewing. Furthermore a personal watchlist can be created.
- **Pacifica[12]**: It is a mental health application to prevent depressions and stress. The daily mood can be monitored, voice records can be saved and a conversion with the community can be started.
- **MarketWatch[13]**: It is the mobile news and data reader of the MarketWatch website.
- **Sworkit[14]**: It is a fitness application, where different workouts and exercises are provided.
- **Untappd[15]**: It is a social application, where the user can find other users, with whom he can go for a beer. Furthermore, it shows the user the nearest bars depending on his location and gives him information about beers which are trending.
- **Honeyfi[16]**: This application can track a couple's income and expenses and help them manage their monthly budget

2.4. REST APIs

This section describes the concept of Representational State Transfer (REST) APIs. Afterwards, I present different technologies and frameworks, which are necessary to create a REST API.

Application programming interface, short API, describes a software, which allows the communication between two applications. Web APIs provide this

[10] https://showcase.ionicframework.com
[11] https://justwatch.com
[12] https://thinkpacifica.com
[13] https://www.marketwatch.com
[14] https://sworkit.com
[15] https://untappd.com
[16] https://honeyfi.com

interface via the internet. There exist thousands of Web APIs for different purposes like updating the social media status or for checking the weather. The most established Web API type nowadays is the REST API. It is protocol independent but if it is used for Web APIs, it typically takes the advantage of the HTTP protocol. There are six key constraints that define a REST API [10, 32]:

- **Client-Server**: Client and server should be separated, which means that the development of client and server can be done independently without knowing about each other.
- **Stateless**: Requests can be sent and processed independently. The server, as well as the client, does not need to know the state of the other part. Each request contains all necessary data to create a successfully completion. This constraint helps REST to achieve reliability, scalability and a good performance.
- **Cache**: To handle the overhead of requests, which a stateless API caused, the REST API should be designed to encourage the storage of cacheable data.
- **Uniform Interface**: This is a key constraint to decouple client and server. The uniform interface should allow the client to communicate with the server using only one language, independently of the architectural backend of the systems.
- **Layered System**: Different functionalities should be divided into multiple layers. With these layers, a hierarchy can be created to improve the performance and to create a more scalable and modular application.
- **Code on Demand**: It is the only optional constraint. It allows, that code or applets can be transmitted over the API, which are then usable for the application.

REST APIs can be build in various languages with different frameworks. The next section will focus on a Python[17] framework called Flask.

2.4.1. REST API Frameworks

In this section I introduce the Flask framework, which is a microframework for Python. Furthermore I describe an extension of Flask for quickly building REST APIs called Flask-RESTPlus.

[17] https://python.org/

Flask

Flask is a microframework for Python, which provides tools, libraries and technologies to build web applications. Because of its architecture, Flask has only a few fixed dependencies, like Werkzeug[18] and Jinja2[19]. Other dependencies must be added manually. This makes Flask very lightweight, but a lot of work is needed to increase the functionality. Flask has a rich set on powerful extensions, which can handle the most common web development tasks like HTTP request parsing and response handling or session management and template rendering with Jinja2 [34].

Flask RESTPlus

Flask RESTPlus gives the user the possibility to build REST APIs quick and easy. It defines and documents endpoints, validates input and formats the output as JSON[20]. Furthermore it turns Python exceptions into HTTP responses, minimises boilderplate code and it is able to generate interactive documentation using Swagger UI [19].

2.4.2. Message Broker and Task Queues

Message broker and task queues are used, if tasks have to run asynchronously like expensive background calculations, notifications or thumbnails generation. The following example takes RabbitMQ[21] as the message broker and Celery[22] for the task queues. As shown in Figure 2.11 the producer, which is a Celery client, creates a task and send it to the broker. The broker then distributes the tasks according to their routing rule to the different queues. Afterwords the broker delivers the tasks from the queues to the workers of the consumer. The consumer can have one or multiple Celery workers, which executes the tasks. The result of these tasks is stored in the result backend, which can be a database. Only via this backend, the result data is accessible [36].

[18] http://werkzeug.pocoo.org/

[19] http://jinja.pocoo.org

[20] JavaScript Object Notation

[21] https://rabbitmq.com

[22] http://celeryproject.org

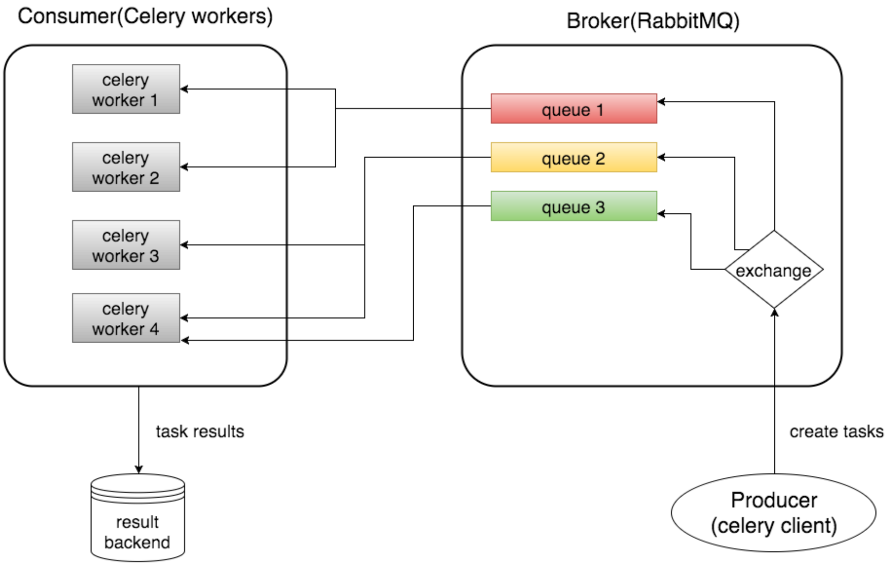


Figure 2.11.: Architecture of Celery and RabbitMQ which schematically shows how a task is handled from [36]

2.4.3. Docker

Docker is a software to isolate applications into small and lightweight execution images called containers. A container packages the entire application with all the libraries, configuration files and dependencies. Also the container shares the operation system kernel, which makes it possible to run the container in different operating systems. The big advantage of Docker is that due to this technology it is simple to deploy a system on multiple machines without having to fear drawbacks.[11]
To run and link multiple containers, Docker-compose can be used. This tool can start several containers with only one command. The container start order can be specified in a file [12].

3. Requirements and Use Cases

This chapter provides a detailed description of the application. Furthermore, I define use cases and requirements for the application and link them.

The requirements are split into functional and non-functional requirements. A functional requirement describes a behaviour that a system shows under certain conditions. A non-functional requirement describes a property or characteristic that a system must provide or a restriction that it must comply with [35].

3.1. General description

The main goal of the project is to provide recommendation lists of different domains to the user via a mobile application. Multiple platforms have to be supported and no special device permissions should be needed to use the application. Available domains include movies, board games and video games. These domains should be easily extensible in the future, which means, it should be able to easily add a further domain. Furthermore, the application should store personal data such as favourites, ratings and a search history. To get a better overview about the quality of the recommendations, an evaluation system should be provided for individual items in the list. These evaluations should be stored in a database. To create the best possible user experience, the user should receive a notification after a recommendation computation is finished. This avoids long static waiting times during a computation, since other tasks can be performed in the meantime. It should not be possible to send multiple requests to one engine simultaneously. To create a request for one of the several recommendation engines, a search functionality for the different parameters is provided. The search functionality uses a provided dataset, e.g. a filtered and sorted IMDb dataset is used for the movie domain. It is accessible through the supplied API. An optional user login is available to backup and restore the personal data. To create a user for the application, a user name, an email address and a password is needed. For managing the stored data, a functionality to delete the different personal data is implemented.

Only movie specific details will be discussed, because only the movie recommender engine is finished at the moment. In the future this project will be adapted with other domains.

3.2. Use Cases

A use case describes the externally visible behaviour of a system from the user's point of view. This user interacts with the system to achieve a certain goal or behaviour. A use case has a short description and a basic path, how the goal can be achieved. Additionally, it can optionally has an alternative path, which describes a behaviour of the system in case of an error or it describes an alternative path to reach the goal. Use cases are written in a simple language so that both parties, users and developers, get the same understanding of the system.

In the following section I define certain use cases, describe them and draw an use case diagram, to illustrate the connections between them.

UC1 Start application

The user starts the application for the first time.

Basic Path:

1. The user clicks on the rbz.io icon in the application list of the operating system to open the application.
2. The application starts and an introduction tour is displayed.
3. The user goes through the tour and finishes it.
4. Afterwards, the home screen is displayed.

Alternate Path:

- Instead of **step 3**, the user presses the button "skip introduction tour".

UC2 Create movie recommendation

A movie recommendation list is displayed after the user starts the computation with some parameters. This use case begins with the start of the application by the user.

Basic Path:

1. The user selects the movies domain on the home page.
2. Afterwards, he searches for different parameters like movies, actors or genres. If he finds something he likes or dislikes, he adds this parameter to the recommendation request. Also the result length and the preferred year can be selected.
3. The user starts the computation of the recommendation list by pressing a button.
4. The user has to wait until the recommendation engine provides the result.
5. After waiting a few seconds, the result page with the recommendation list is displayed (**UC3**).

Alternate Path:

- **Step 2** can be skipped and the user can start the computation without searching and inserting some parameters.
- At any time during **step 1-4**, the user can cancel the request and start from the beginning again.

UC3 Show result

This use case is preceded by **UC2**. After the computation of the recommendation list, the result page with the recommendation list gets shown to the user.

Basic Path:

1. The user is waiting till the calculation has finished.
2. Afterwards, the recommendation list is displayed on the result page.

Alternate Path:

- **Step 1** can be extended. The user is able to browse through the application while waiting. He receives a notification when the result is available.
- After **step 2**, it is possible to refine the request and start the computation of a new recommendation list again (**UC2**)

UC4 Show history with details

A list with history entries is displayed. It includes older recommendation requests with their results. This use case starts on the home page of the application.

Basic Path:

1. The user opens the side menu and goes to History.
2. A list with history entries, sorted by date, is shown.
3. The user has to click on one entry to expand it.
4. All request parameters, as well as an overview about the recommendation list is shown.

Alternate Path:

- Between **step 2 and 3**, an additional step can be added. The user is able to filter the history entries. A predefined set of time periods is provided, but the user can self-select the time period as well.
- After **step 4**, the user can go to the result- and request page of the history entry.
- Instead of **step 4**, the user has the possibility to delete the entry.
- **Step 2-4** can be omitted if no entries exist.

UC5 Show favourites and delete entry

The user sees the favourites list after navigating to it. He deletes one entry of the list. This use case starts on the home page of the application.

Basic Path:

1. The user opens the side menu and goes to My Favourites.
2. The favourite entries, containing the name, year, added-on date and additional data according to the domain, are displayed in a list.
3. The user swipes one entry to the left and presses the delete button.

Alternate Path:

- Instead of step 3, the user can delete a favourite entry via the ratings- or result page by clicking on the favourites icon.
- **Step 2-3** can be omitted if no entries exist. Instead a description, how to add an entry to the favourite list, is displayed.

UC6 Show rating and change entry

The user navigates to the ratings section and sees the rating list, where the entries are grouped according to their rating scores. Afterwards, he changes one rating score of an entry. This use case starts on the home page of the application.

Basic Path:

1. The user opens the side menu and goes to My Ratings.
2. The rating entries, grouped by their rating scores, are shown.
3. The user swipes one entry to the left and gets two possibilities:
   a) delete entry,
   b) change the rating score.
4. The user changes the rating score and the entry is moved to the correct category.

Alternate Path:

- Instead of **step 4**, the user can change a rating of an entry via the favourites- or result page by clicking on the ratings icon.
- **Step 2-4** can be omitted if no entries exist. Instead, a description, how to rate items, is displayed.

UC7 Register and login

The user can register himself with a user name, an email address and a password. After the registration, he is able to login successfully. This use case starts on the home page.

Basic Path:

1. The user opens the side menu and goes to settings.
2. In the settings page, the user navigates to the account section.
3. The user presses the registration button and fills out the required fields. Afterwards, he submits the data.
4. The user is now able to login with the created account.

Alternate Path:

- **Step 4** can be omitted, if the user enters invalid data into the registration form.

UC8 Backup and restore personal data

The user can backup and restore his favourite, rating and history entries. This use case starts on the settings page, where the user is already logged in (**UC5**).

Basic Path:

1. The user navigates to the account section, where the different possibilities to backup or restore the personal data is shown.
2. The user presses the button to backup or restore the data.
3. A message is shown to the user, that the backup or the restore process was successful.

Alternate Path:

- Instead of **step 3**, an error message gets displayed if the process fails.

UC9 Delete all personal data

The user can delete all entries of the favourite, rating or history list at once. This use case starts on the settings page.

Basic Path:

1. The user navigates to the storage section, where the different possibilities to delete the entries is shown.
2. The user presses the delete button.
3. Before the data is deleted, the user has to confirm his decision.
4. After confirming, the data gets deleted.

Alternate Path:

- Instead of **step 4**, the user cancels the procedure and the data is not deleted.

UC10 Show more recommendations

This use case is preceded by **UC3**. After pressing the show-more button, five more entries are listed.

Basic Path:

1. User pressed the "show-more" button on bottom of the page.
2. Five more entries are displayed after some loading time.

Alternate Path:

- Instead of **step 2**, an error message gets displayed, if the calculation of the new entries fails.
- **Step 2** can be extended. The user is able to browse through the application while the "show-more" process is working. He receives a notification when the result is available.
- Instead of **step 2**, the user cancels the process and no more entries are displayed.

UC11 Share recommendation list with social accounts

This use case is preceded by **UC3**. The user shares his recommendation list with his preferred social account by clicking on the share button.

Basic Path:

1. The user navigates to the share button, which is located in the header, and presses it.
2. A list of different options for sharing is shown.
3. The user chooses his preferred social account and shares the recommendation list.

UC12 Feedback for recommendation list entry

This use case is preceded by **UC3**. The user provides feedback for an entry of the recommendation list. He wants to tell the recommendation engine, if he is satisfied with the result.

Basic Path:

1. The user navigates to the bottom of an entry in the recommendation list.
2. A button to provide feedback is shown.
3. The user presses the button and five icons with a different meaning appear.
4. After that, the user selects one of the provided options and evaluates the entry.

Alternate Path:

- After **step 4**, the user is able to change his opinion and he can evaluates the entry again.

UC13 Rate specific item of recommendation list

This use case is preceded by **UC3**. The user rates one entry of the recommendation list.

Basic Path:

1. The user navigates to the upper left corner of an entry in the recommendation list.
2. A rating icon is shown. By clicking on it, it is expanded to a five-star rating.
3. The user selects the desired rating.
4. Afterwards, the entry is rated and also stored in the rating list.

Alternate Path:

- After **step 4**, the user is able to change his opinion and he can rate the entry again.

UC14 Go to external link

This use case is preceded by **UC3**. The user enters a provided external link to get more details about the recommendation list entry.

Basic Path:

1. The user navigates to an entry in the recommendation list. Different links are provided for each entry.
2. The User clicks on one link and he is redirected to an external page.

UC15 Add recommendation item to favourites

This use case is preceded by **UC3**. The user adds an entry of the recommendation list to the favourite list.

Basic Path:

1. The user navigates to the upper right corner of an entry in the recommendation list.
2. An favourites icon is shown.
3. The user clicks on the icon.
4. Afterwards, the entry is stored in the favourite list.

Alternate Path:

- After **step 4**, the user is able to delete the entry of the list by clicking on the icon again.

Use Case Diagram

Figure 3.1 shows the relations between the different use cases. The two actors are on the one hand the user, who runs the application, on the other hand the recommender engine, which computes the recommendation list for each domain.

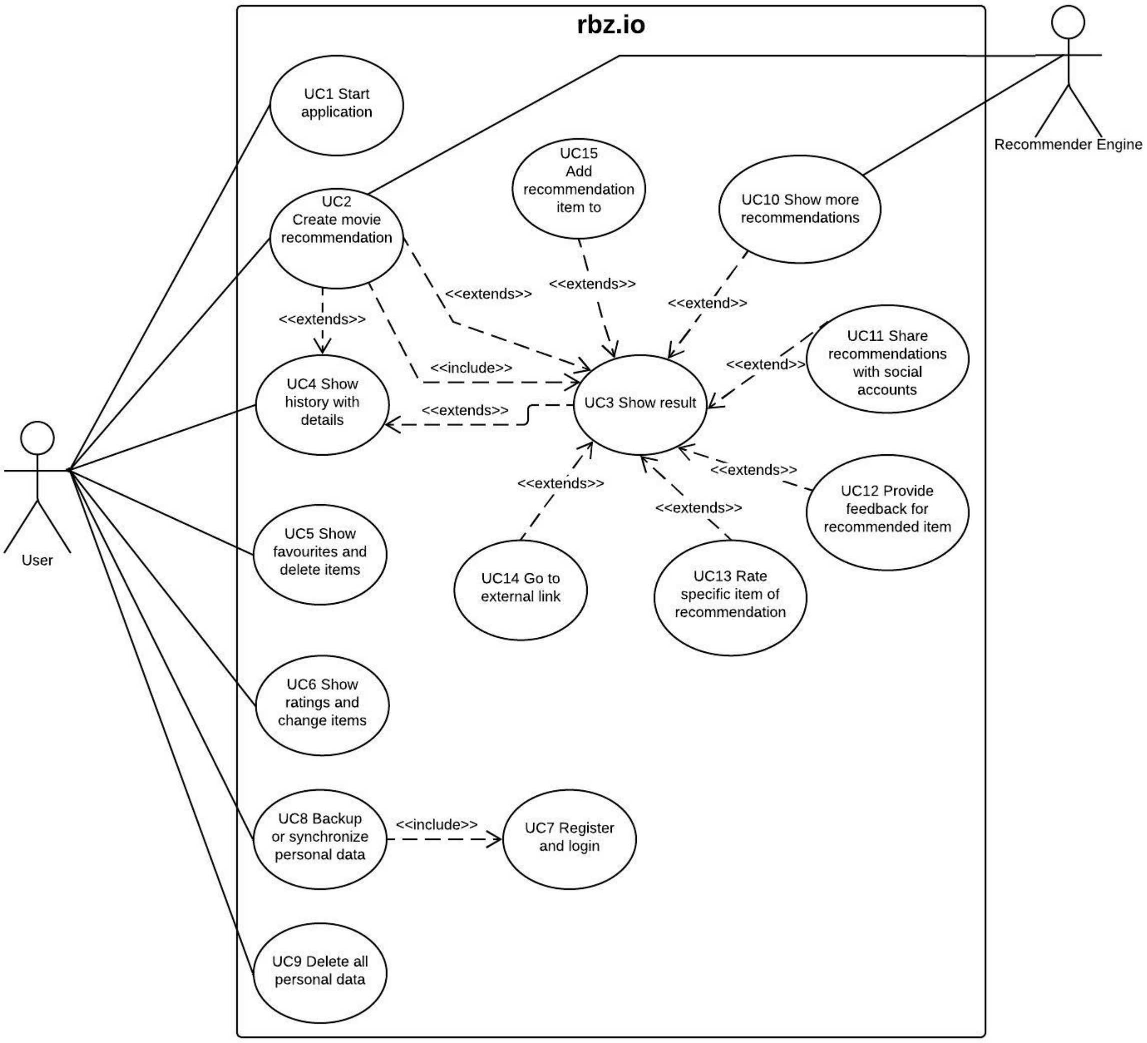

rbz.io
User
Recommender Engine
UC1 Start application
UC2 Create movie recommendation
UC15 Add recommendation item to
UC10 Show more recommendations
UC11 Share recommendations with social accounts
UC3 Show result
UC4 Show history with details
UC12 Provide feedback for recommended item
UC5 Show favourites and delete items
UC14 Go to external link
UC13 Rate specific item of recommendation
UC6 Show ratings and change items
UC8 Backup or synchronize personal data
UC7 Register and login
UC9 Delete all personal data
<<extends>>
<<extends>>
<<extends>>
<<extend>>
<<extend>>
<<include>>
<<extends>>
<<extends>>
<<extends>>
<<extends>>
<<include>>

Figure 3.1.: Use case diagram of the predefined use cases. It shows the relations between each use case.

3.3. Functional requirements

The following functional requirements are extracted from the predefined use cases and the description. They should help to get an technical understanding of the application and they should help to estimate the approximate workload.

F1 Start Page - On the start page it should be possible to choose between the different domains. If a recommendation is currently being calculated or a result is already available, a notification should be displayed on the screen.

F2 Movie Requests - The user should be able to create a request with given parameters. Table 3.1 represents the required parameters. Each parameter, except of the length, should be assignable to a positive or a negative alignment. A positive alignment represents items the user likes, a negative alignments points to an aversion.

Parameter	Type	Optional	Description
Movie	Array[string]	yes	List of selected movies
Actor	Array[string]	yes	List of selected actors
Genre	Array[string]	yes	List of selected genres
Keyword	Array[string]	yes	List of selected keywords
Year	Array[number]	yes	Time period
Length	Number	no	Number of movies, which are displayed on the result page

Table 3.1.: Parameter specification for a movie request

F2.1 Modify selected items - It should be possible to delete each item. Furthermore, the alignment of each element should be changeable.

F2.2 Minimize items - To save space on the screen, it should be possible to minimize the items in each section.

F3 Movie Detailed Search - The user should be able to search for movies, genres and actors within one list. There should be automatic suggestions for each entity after each key press. The suggested movies should include the title, the year and the movie poster. There should also be a section where you can enter keywords.

F4 Movie Recommendation Result - The recommendation result should be displayed in a list. Each item should contain a title and the year, as well as the movie poster. Furthermore, links to Youtube, IMDb and Amazon should be provided.

F4.1 Show more - A "show more" functionality should be implemented. After a call, five more movies should be displayed.

F4.2 Add to My Favourites - It should be possible to add each movie to My Favourites.

F4.3 Add to My Ratings - It should be possible to add each movie to My Ratings. The rating scale should be between 1 and 5.

F4.4 Social share - It should be possible to share the entire recommendation list with your social accounts.

F4.5 Refine request - The user should be able to refine the request after the recommendation was computed.

F4.6 Evaluate recommendations - The user should be able to evaluate every single recommendation in the list. This evaluation should indicate if the calculation for the recommended movie was good or bad.

F4.7 External link - The user should be able to get information about the entry of the recommendation list via an external link.

F5 Push Notifications - The user should receive a push notification after a computation of a recommendation is finished. The notification should contain the domain from which the result comes. The different domains should contain different notification texts. In addition, it should be differentiated whether the result comes from a new request or whether the user has used the show-more functionality.

F6 History - A history of the requests should be provided. It should include the timestamp, the request and the corresponding recommendation list.

F6.1 Filter - The history should be filterable by date. The user should be able to filter by a self-selected timeperiod, also a predefined set of date (last week, last month and last year) should be provided.

F7 My Favourite page - A page with a list of favourite entries should be provided. Each entry should have an "added on" date. Every item should be deletable.

F7.1 Movies - Each item in the My Favourite list, which belongs to the movies domain, should include the title, the year, the genre, the poster and the rating if it is available. Also the functionality to rate the movie should be provided.

F8 My Ratings page - A page with the rated entries, should be provided. Each entry should have an "added on" date and a function to add the item to the favourites. The entries should be grouped by the rated scores. Every entry should be deletable and a function, for adding it to the favourite list, should be provided.

F8.1 Movies - Each item in My Ratings, which belongs to the movies domain, should include the title, the year, the genre, and the poster.

F9 Help tour - A tour, which leads the user through the app using a simple example, should be displayed at startup. There should be the possibility to disable the tour at startup. Furthermore, the tour should be accessable within the app, that the user can walk through it whenever he wants.

F10 Optional user login to backup data - The user should have the possibility to backup the history, favourite and rating entries. For that purpose an optional user login has to be provided.

F10.1 User registration - For the registration, the user has to provide a user name, an email address and a password.

F11 Delete data - The user should be able to delete all entries of a history, favourite and rating list at once.

F12 Device detection - To see which user or device is sending a request and how many users are using the app, the device UUID should be stored in the database when the application starts.

3.4. Non-functional requirements

The following non-functional requirements are extracted out of the description. They should give an overview of the needed behaviour of the application. Furthermore, they are import for the architectural design and the technology selection.

Extendibility - The application should be easily extendible for other domains.

Usability - The application should allow the user to use the application without any prior knowledge or explanation.

Usage of different engines - It should be possible to make simultaneous requests to different engines.

Usage of one engine - It should not be possible for a user to make another request to one engine, if a computation is still running.

Connectivity - The application only has to work online.

Platforms - Android and iOS must be supported by the application.

Permissions - No special operating system permissions should be required.

3.5. Relations between use cases and requirements

Table 3.2 and Table 3.3 shows the relations between use cases and requirements. Only functional requirements are covered, because these are directly connected to the use cases. Every use case has to cover one or more requirements.

	Use Cases							
Requirements	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
F1	■	■						
F2		■						
F2.1		■						
F2.2		■						
F3		■						
F4			■					
F4.1								
F4.2								
F4.3								
F4.4								
F4.5			■					
F4.6								
F4.7								
F5			■					
F6				■				
F6.1				■				
F7					■			
F7.1					■			
F8						■		
F8.1						■		
F9	■							
F10								■
F10.1							■	
F11								
F12	■							

Table 3.2.: Part 1 of the use case coverage of functional requirements

	Use Cases						
Requirements	UC9	UC10	UC11	UC12	UC13	UC14	UC15
F1							
F2							
F2.1							
F2.2							
F3							
F4							
F4.1		■					
F4.2							■
F4.3					■		
F4.4			■				
F4.5							
F4.6				■			
F4.7						■	
F5							
F6							
F6.1							
F7							
F7.1							
F8							
F8.1							
F9							
F10							
F10.1							
F11	■						
F12							

Table 3.3.: Part 2 of the use case coverage of functional requirements

4. Implementation

This chapter describes the architecture and the design of the system. First, I present the overall architecture, then I continue with the explanation of the individual components and which technologies are used. At the end of this chapter I discuss the page design and explain the structure and the functionality of each page.

4.1. Software Architecture

Software architecture is a fundamental discipline in the software development process. At the beginning of creating a software, it defines how the structure looks like and how the individual components will be working together. These decisions are very business critical, because it is expensive to change the architecture during or after the implementation. Bass et al. [7] defines software architecture as follows:

> *"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."*

The set of structures of my system consists of five different parts as shown in Figure 4.1. The Web API is the interface between the various components. It handles requests from the application. The application provides different possibilities to create requests. Also it supports the user to find parameters which make the result more accurate. After the API receives a request, it checks in the database, if such a request has been sent before. If this is the case and the database entry is not older than seven days, the API can directly return a result to the application. Otherwise the request will be stored in the database and is forwarded to the recommender engine. The engine then computes a list of recommendations with the given data and returns this list to the API. The API stores the result in the database, and sends a message that the result is available, to the notification manager. This message contains the database entry ID and the notification device ID. The notification device ID is created at the start of the application and is transferred with the request. The notification manager notifies the application that a result is available. After the application receives the notification, it sends a request with the provided ID to the API, which returns the result immediately. The application is then able to display the computed recommendation list.

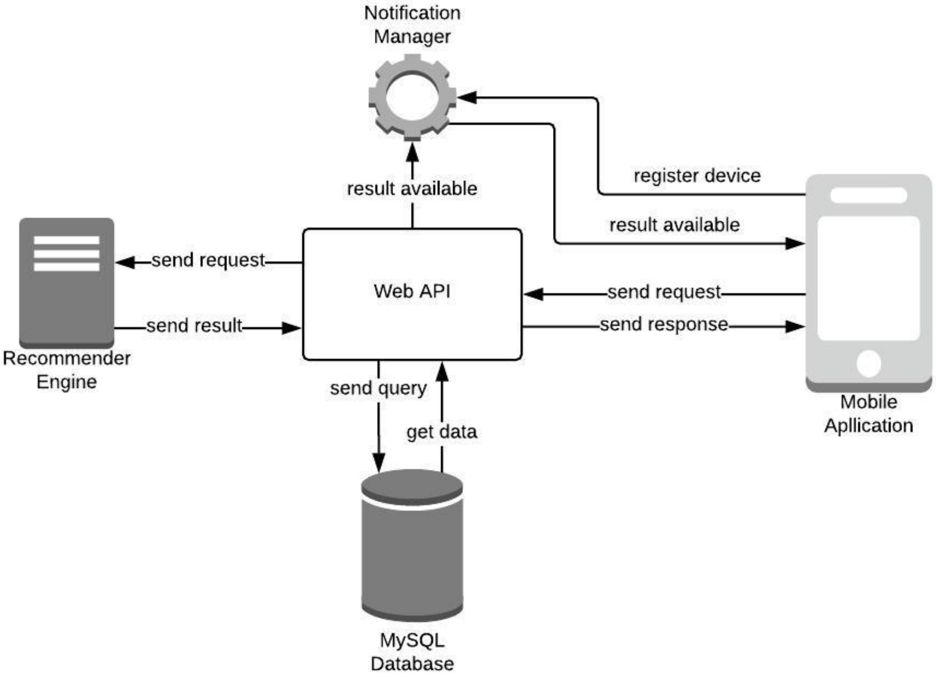


Figure 4.1.: Schematic representation of the overall system architecture

In the following sections, I am going to discuss each component in detail and how they are implemented.

4.1.1. Web API

The Web API connects each component of the system. It provides data for the search suggestions in the mobile application, as well as responses with the recommender engine results or with the corresponding database ID. The API has its own internal logic to optimize the performance of recommender engine responses. There are three different cases how the API returns a recommendation result list to the application:

1. **Immediate response via database**: The API checks in the database if there exists an identical request with a valid result, i.e., it is not older than seven days. If this is the case, the API returns this result immediately to the application. The request is stored as a new database entry with a link to the identical request as its parent.
2. **Immediate response via recommender engine**: The API checks the database and finds no entry with a request, which is identical to the new request. Then the API creates a new database entry and forwards the request to the correct domain queue. If there are other requests in the queue, it has to wait until the others have been processed. If there are no other requesst in the queue, it is forwarded to the Celery worker, which sends the request to the recommender engine. The recommender engine computes a result list. If the Celery worker receives a response with the result from the recommender engine within three seconds, the database entry is updated with the result list and the API returns this result immediately to the application.
3. **Response after defined timeout**: This case works similar to case 2, but the recommender engine does not provide the result within the defined timeout. In that case, the API returns only the ID of the database entry to the application. When the engine finishes the computation, the Celery worker notifies the notification manager that a result is available. Also the worker updates the database entry with the provided result list.

The database requests for the search suggestions are handled in a very simple way. The application sends a request to the correct API endpoint and the API returns the result of the database query to the application. I will discuss the different endpoints and their meaning at the end of this section.

Structure

The API is implemented with five Docker containers as shown in Figure 4.2. Docker is used to increase the portability on different systems. To link the docker containers, the tool docker-combine is used. The NGINX[1] web server provides the communication between the mobile application and the API via the internet. To make the communication secure, the web server works with SSL. SSL stands for Secure Sockets Layer and it is a protocol to encrypt data for a secure transport via the internet. The API is implemented with the Python Flask framework. The connections between NGINX and Flask is made by

[1] https://nginx.com

Gunicorn[2]. Gunicorn is a Web server Gateway Interface (WSGI) for Python. It is an interface between web servers and web frameworks like Flask, to increase the portability of web applications on different web servers. The API supports multiple endpoints. To provide a good usability, the API generates a documentation with Swagger UI for each endpoint. To store data, the API works with an MySQL database (which is described in Section 4.1.2). Also the provided data for the search suggestions is stored in this database. To handle multiple requests in order to avoid overloading of the recommendation engine, queues from RabbitMQ are used. For each domain there is a separate queue. These queues process the request individually one after the other. They get the data from the API's Celery client and forward it to the Celery worker, which manages the communication with the recommender engine, the notification manager and also the database. Both, Celery and RabbitMQ, are implemented separately in into different Docker containers.

[2] https://gunicorn.org

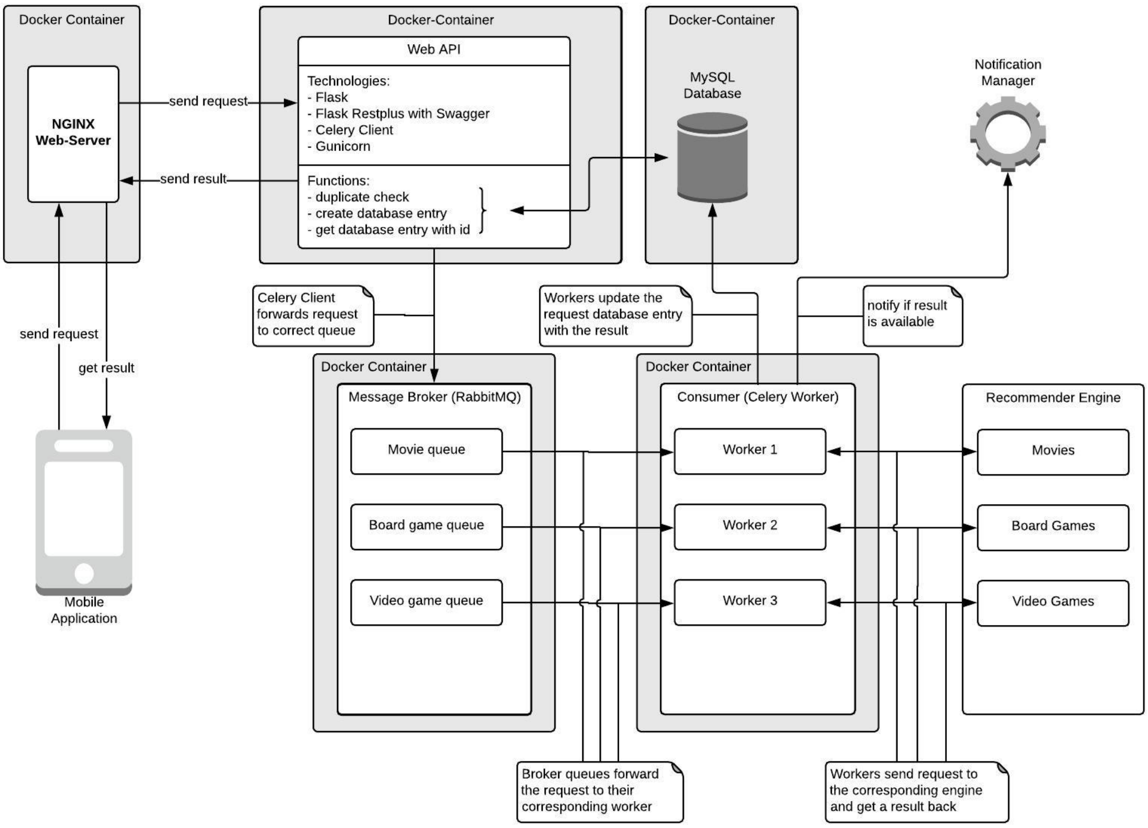


Figure 4.2.: Detailed representation of the API structure

API Endpoints

An API endpoint defines the connection point to the API. If an endpoint is called, the API knows which functions or resources are needed and what operations have to be performed. To structure the endpoints, Flask Restplus namespaces are used. Table 4.1 lists the different endpoints. The API supports the HTTP methods GET and POST. The GET method is for retrieving data from the API and the POST method is for sending data to the API. The data is sent through the body of the request. To identify a valid request to the API, an API key is used. This key must be included in the header of every request, otherwise the request will be rejected. The structure of a request is as follows:

https://<host>:<port>/api/rbz/<namespace>/<endpoint>

where host defines the ip address of the server and port defines, on which port the API is available.

HTTP Method	Namespace	Endpoint
POST	movies	<string>/<int>
		movie/vote
	general	uuid/<string>
		user
		user/deviceId
		backup
GET	movies	<int>
		genre/<string>
		movie/<string>
		person/<string>
		movie/details/<string>
	general	user/<string>
		password/<string>/<string>
		backup/history/<int>
		backup/favourite/<int>
		backup/rating/<int>
		backup/dates/<int>

Table 4.1.: List of all API endpoints

4.1.2. MySQL Database

A MySQL database is used to store API specific data and to provide datasets for the search suggestions. It is an own Docker container in the API Docker association. To cover all functionalities, nine tables are required as shown in Figure 4.3. The connection between the API and the database is established via the Python library SQLAlchemy. The following list gives a brief overview of each table:

- **user**: This table is used for user specific data. The registration process of the application uses this table to add a user to the system. The login process verifies the user via this table.
- **device**: It stores the unique device id to enable user specific evaluations.
- **rbz_api**: This table includes API specific data, like the request from the mobile application and the corresponding result list of the recommender engine.

- **user_request**: To link the request to a user or a device, this table is used. The purpose of this table is to make a user evaluation possible.
- **recommendation_votes**: This table stores the voting score of a recommendation list entry. It is used to check if the recommender engine delivers satisfactory results to the user.
- **movie**: This table provides a list of movies extracted from IMDb.
- **person**: This table provides a list of persons, e.g. actors, extracted from IMDb.
- **genre**: This table provides a list of genres extracted from IMDb.
- **backup_data**: This table is used to store personal data of a user, like the history, the favourite list or the rating list. Also an added-on date is added to the respective entry, to be able to provide information to the user.

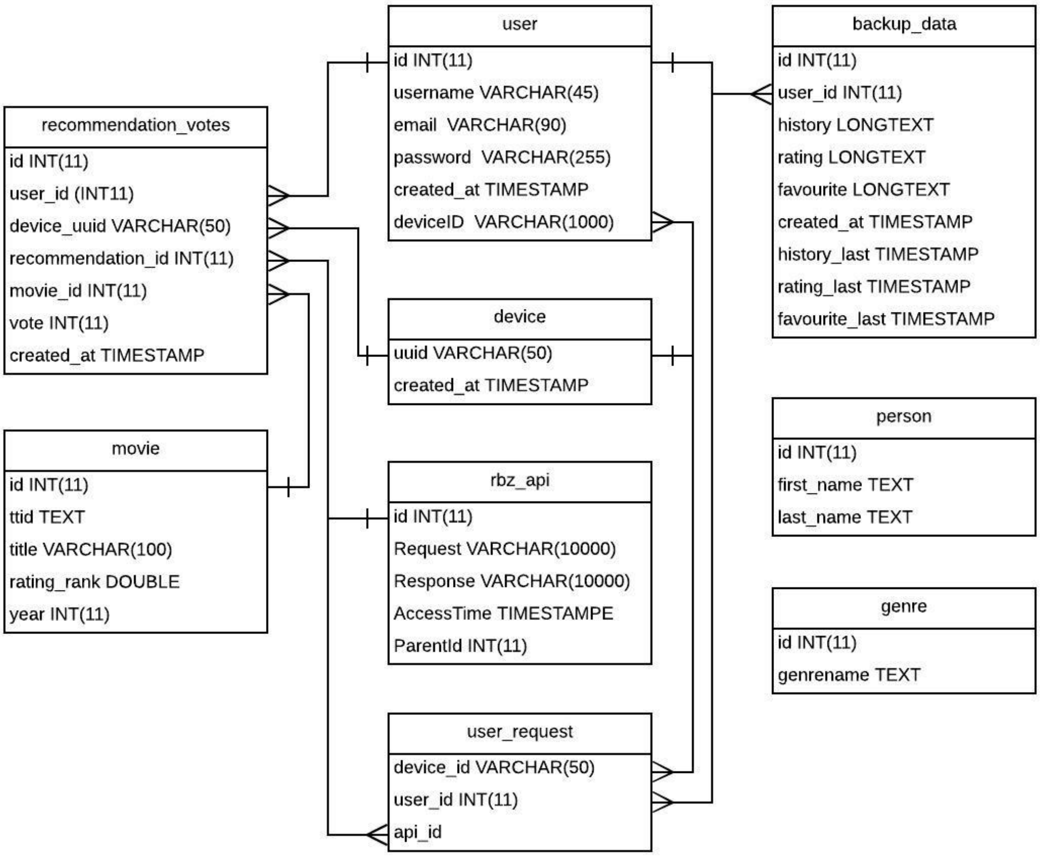


Figure 4.3.: Database schema with relations

4.1.3. Recommender Engine

The recommender engine is provided by the Institute of Interactive Systems and Data Science of the Technical University of Graz. It creates narrative-driven recommendations for different domains like movies, board games and video games [13]. The mobile application can interact with the engine via the API. The API can access the engine directly, because it is located on the same server.

4.1.4. Notification Manager

The notification manager is implemented with OneSignal[3]. This is an open source software for push notification services. Push notifications are used to inform the user about different activities of the mobile application. In our case, it informs the user that a recommendation result list for the previous request is available. It is easy to implement, because it provides a RESTful API and Ionic offers a corresponding plugin. In our system, the mobile application registers the device at OneSignal and OneSignal returns a unique push ID. This ID is sent with every request to the API. The API stores this ID and informs the notification manager when a result is available from the recommender engine. This information message includes the push ID and additionally a notification text. With this information, the manager knows to whom to send a notification. If the manager sends the notification, two different cases exist on the mobile device:

- **Application is opened**: In this case, the application shows only a prompt with the given text.
- **Applications is closed**: In this case, a notification is displayed on the screen. In addition, it can ring or vibrate depending on the device settings. If the notification gets pressed, the application opens and the recommendation result list is displayed.

4.1.5. Mobile Application

This section provides a detailed description of the mobile application. First, I explain why I have chosen Ionic for the implementation.. Afterwards, I give a brief description of the used Ionic plugins and I show the structure of the application. A page flow diagram at the end of the section will introduce the application page design of the following section.

[3] https://onesignal.com/

Technology Selection

For the technology selection I compared the different application types according to their characteristics that the application requires. Table 4.2 shows the rating of each characteristic. If the type matches the characteristic completely. it receives 2 points. If the type matches the characteristic partly, it receives 1 point and otherwise 0 points. In addition, there is a weighting which indicates the importance of the property. The total score is calculated from the sum of the points multiplied by the their weight.

Aspects	**Native**	**Hybrid**	**Web**	**Weight**
Multiple Platforms	0	2	2	2
Easy Maintainability	0	1	2	2
Expertise Programming Language	1	2	2	1
Local Storage	2	2	0	1
Development Costs	0	1	2	1
Performance	2	1	0	1
Notifications	2	2	0	0.5
App Store Distribution	2	2	0	1
Device Information	2	1	0	0.5
Total Score	**9**	**15.5**	**12**	

Table 4.2.: Evaluation of the different technologies

It can be seen that the hybrid solution achieves the highest score. For this reason, I have chosen the Ionic Framework, which is a framework for hybrid applications. Another reason for this framework was, that many user interface elements are needed and Ionic provides a rich set of it. Furthermore, I have already worked with it, which leads to a reduction of the development time. At start of the master thesis, Ionic Framework provided a new beta version called v4. I chose the beta version, because it seemed to be more efficient and I wanted to use the latest technology. Another advantage is that it is an open source framework with a big developer community. Furthermore, it provides a good and clear documentation and for testing the application on a device, an application for Android and iOS called Ionic DevApp is provided.

Used Plugins

The following list shows the used Native Ionic plugins [17] and which function they take over in the application.

- **NativeStorage**: Provides access to the native storage for Android and iOS devices. This plugin is used for managing the complete data storage of the application.
- **HTTP**: Provides the communication with HTTP servers for Android and iOS devices. It is used for the API communication.
- **Device**: Provides information about the underlaying device. This plugin is used to get the unique device ID, which are stored in the database and used for user recognition.
- **ScreenOrientation**: Provides the functionality to set the orientation or lock the screen. This is used to restrict the application to the portrait orientation.
- **OneSignal**: Provides the connection to the OneSignal Service. It is used to register the device at OneSignal and to receive notifications.
- **Network**: Provides access to the network information. It is used to check the connectivity of the application.
- **Keyboard**: Provides access to the platform keyboard. It is used to close the keyboard after typing in a search text.

Structure

The Ionic application is mainly split into page classes, service classes and interfaces. Page classes provide the view with page specific logic, service classes provide logic, which can be used in multiple classes. Interfaces define the expected structure of objects and can also be used in multiple classes. The application starts with the index.html on the platform's web view. This file initiates the application with calling the app-component, which provides the structure and navigation of it. The application uses a sidemenu as navigation to the different pages. Furthermore, this component prepares the application for startup by setting the screen orientation, initializing the native storage and loading data from the storage to service variables.

Figure 4.4 shows the connections between different page and service classes. The service classes HelperService, ConstantsService and StorageService have not been considered because the are associated with all page classes.

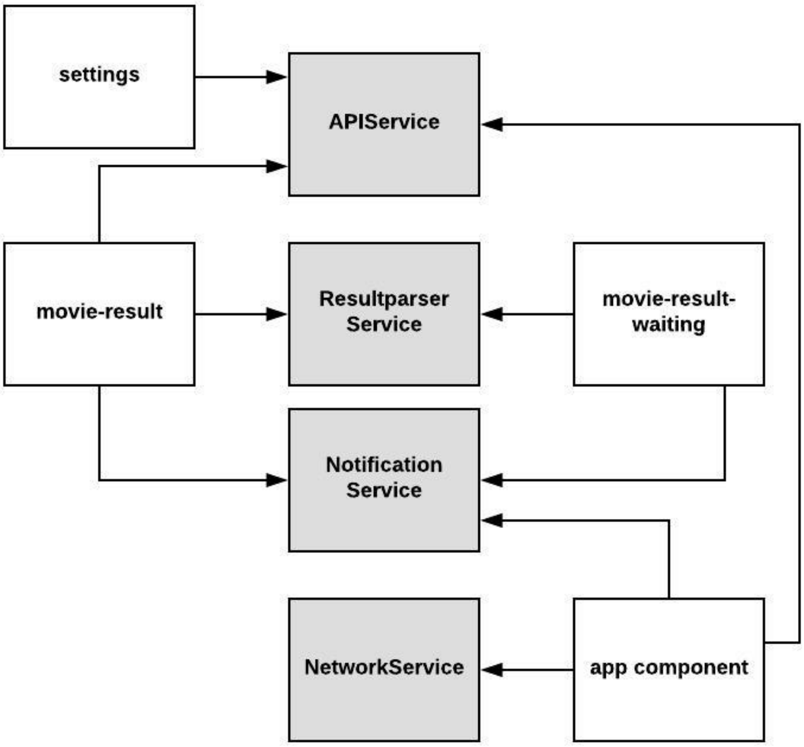
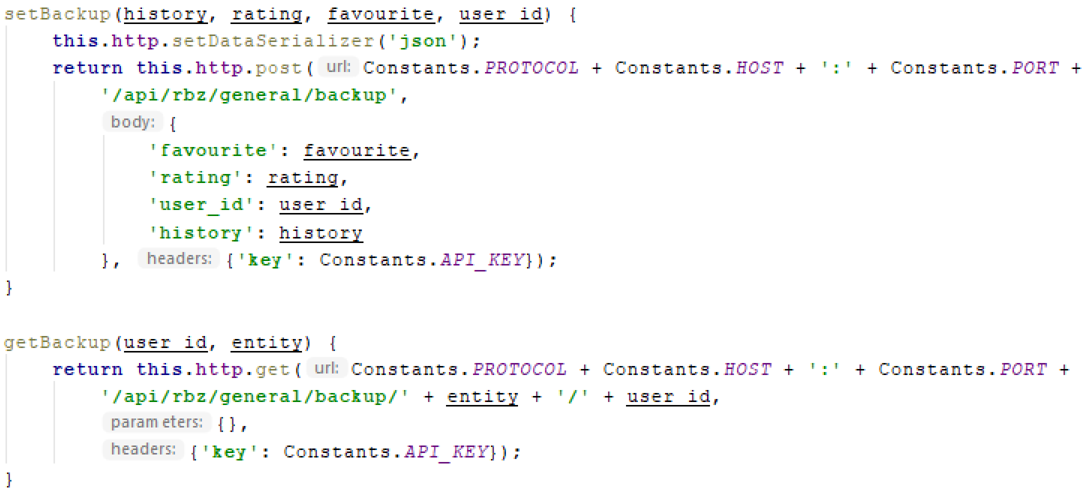


Figure 4.4.: Part of the mobile application class structure

Services

A service class provides functionalities and data, which can be used in multiple classes. Each service-class should have a special purpose for example one service class provides the API communication, another manages the network functionalities. Service classes help to keep the page classes light and clean. In the following list I describe all used service classes in detail.

- **ApiService**: This service class provides the connection to the API. It has multiple functions, which send HTTP GET and HTTP POST calls to the respective endpoint. Figure 4.5 shows a GET and a POST call to the API. Both sends the API key via the header, the POST call also sends provided data through the body.



```
setBackup(history, rating, favourite, user_id) {
    this.http.setDataSerializer('json');
    return this.http.post( url: Constants.PROTOCOL + Constants.HOST + ':' + Constants.PORT +
        '/api/rbz/general/backup',
        body: {
            'favourite': favourite,
            'rating': rating,
            'user_id': user_id,
            'history': history
        },  headers: {'key': Constants.API_KEY});
}

getBackup(user_id, entity) {
    return this.http.get( url: Constants.PROTOCOL + Constants.HOST + ':' + Constants.PORT +
        '/api/rbz/general/backup/' + entity + '/' + user_id,
        param eters: {},
        headers: {'key': Constants.API_KEY});
}
```

Figure 4.5.: HTTP GET and POST call to the API. The setBackup function stores personal data in the database. The getBackup function receives the personal data from the database with a given user id.

- **ConstantsService**: It provides constants for the classes, to centralize the error messages, button- or header texts. Also it provides setting specific data, such as the API key, the OneSignal app id or the host address and port of the recommender engine.
- **HelperService**: The main goal of this service is to provide data to all classes. It provides flags for the view logic, as well as user specific and personal data. Following flags are provided:
  - movie_request_refine: indicates if a request was refined.
  - is_user_logged_in: indicates if a user is logged in.
  - movie_from_history: indicates if the result page is called from the history.

  - waiting_for_movie_result: indicates that a movie recommendation computation is currently in progress.
  - result_computation_finished: indicates that a result is available.
  - result_computation_failed: indicates that a computation failed.
  - result_show_more: indicates that the "show more" function was called and that the recommendation list computation is currently in progress.

  This service stores also the current request and the corresponding recommendation result list. If a user is logged in, this is also recognized by the HelperService. In addition, it provides the rating and favourite list.
- **NetworkService**: This service checks the connectivity of the application. It recognizes if the application is online or offline. If it is offline, an alert pops up, which tells the user that the application is only online usable.
- **NotificationService**: It connects the application with OneSignal. It is responsible for the initialization of the device. Furthermore, it defines what happens, if a notification receives. If a notification is received while the application is running, an alert is displayed, that the recommendation computations has finished. After closing the alert, the user stays on the same page and does not get redirected to the result. If a notification is received while the application is closed and the user clicks on it, the application starts and the user gets redirected to the result list automatically.
- **ResultparserService**: This service parses the recommendation engine result, which is in JSON format, to the corresponding objects. Furthermore, it builds the request, such that the recommender engine can interpret it and then it sends this request to the recommedation engine through the ApiService.
- **StorageService**: This service administers the native storage, which means, it initializes the storage, sets data and gets data from it. The storage is structured with key-value pairs. The value can be an object, but also just a boolean value. The application stores the favourite, rating and history list, as well as the current request with the corresponding result, the logged-in user, and some flags, which manage functionalities on the different pages.

Pages

Pages are primarily responsible for the user interface. Each page has multiple files which provide the logic and the design of them. These are combined in one folder per page. A folder includes an HTML template for the layout, a CSS file for the style and a page component file for the specific logic. The component file uses the Angular @Component decorator, which defines how it should be handled and processed during start and runtime. The template uses standard HTML tags, which can be modified with Angular's template syntax to make it more powerful. An example for this syntax is ngFor, which loops over a tag and displays it as often as defined. Also it is possible to use variables from the component or call component functions. In following list I show the page classes and I describe the functionality of their components.

- **about**: The about page states the author of the application and the licences of third-party libraries.
- **favourites**: The component is responsible for the initialization of the page. It calls the HelperService to get the data and then it calls the StorageService to get the images for each entry. In addition, a delete function is provided, as well as a function to change the rating of an entry.
- **history**: The history component gets the data from the StorageService. It initializes the view with it and provides the filter functionality. The component is responsible to change the view by expanding an entry. Furthermore, it navigates the user to the request- and result page. Since only 10 entries are displayed at the beginning, a "show more" function, which calls the StorageService again, is provided by the component.
- **home**: It provides the navigation to the implemented pages with an implemented side menu. Also, it forwards the user to search pages of the different domains.
- **movie-query**: It collects the data for the search query and provides the function to go the the search page and to process the data received from it. Additionally, it is responsible for changing the alignment or deletion of an entry. It also navigates to the "waiting for result" page with the query data.
- **movie-result**: The component is responsible for displaying the result list from the recommender engine. Furthermore, it provides functions to rate a movie and to add a movie to the favourites list. The different views of the evaluation of an result list entry is also managed by the component.
- **movie-result-waiting**: It only sends the request to the ResultParserService and navigates the user to the result page, if a recommendation result list is available
- **movie-search**: This component uses the ApiService to create search suggestions with inserted search terms. It provides entries to the movie-query page, that this page can build a request.
- **ratings**: The component is responsible to get data, which is displayed. This happens in the initialization phase of the page. Also it provides a delete function, a function to add the entry to the favourite list and a function to

rate the entry again.

- **settings**: It manages the user registration, the login and logout process. Furthermore it provides the functionality to backup or restore the personal data and to delete the complete rating, favourite and history list.
- **tour**: This component is responsible for the functionality, that the introduction tour does not get displayed at start of the application.

Interfaces

Interfaces define the expected structure of objects. With interfaces it is easy to correctly handle objects, because they only accept those parameters that have been defined before. Interface parameters can correspond to any data type, it can also be another interface. Additionally, they can be defined as optional. Listing 4.1.5 shows the interface for genres, which consists of three parameters, Listing 4.1.5 shows the PartialMovieSearchRequest interface, which consists of five parameter. This parameters are arrays with other interfaces and each parameter is optional.

```
export interface Genre {
  id: number;
  name: string;
  alignment: string;
}
```

Listing 4.1: Structure of the Genre interface

```
export interface PartialMovieSearchRequest {
  movies?: Array<Movie>;
  actors?: Array<Actor>;
  timeperiod?: Array<Year>;
  keywords?: Array<Keyword>;
  genres?: Array<Genre>;
}
```

Listing 4.2: Structure of the PartialMovieSearchRequest interface

4.1.6. Software Design

Figure 4.6 shows the page flow of the application. It illustrates how the application navigates through the different pages and which actions are required to get there. A page is structured in a way that it allows intuitive access to other pages. I will describe the design of each page in detail in the next section.

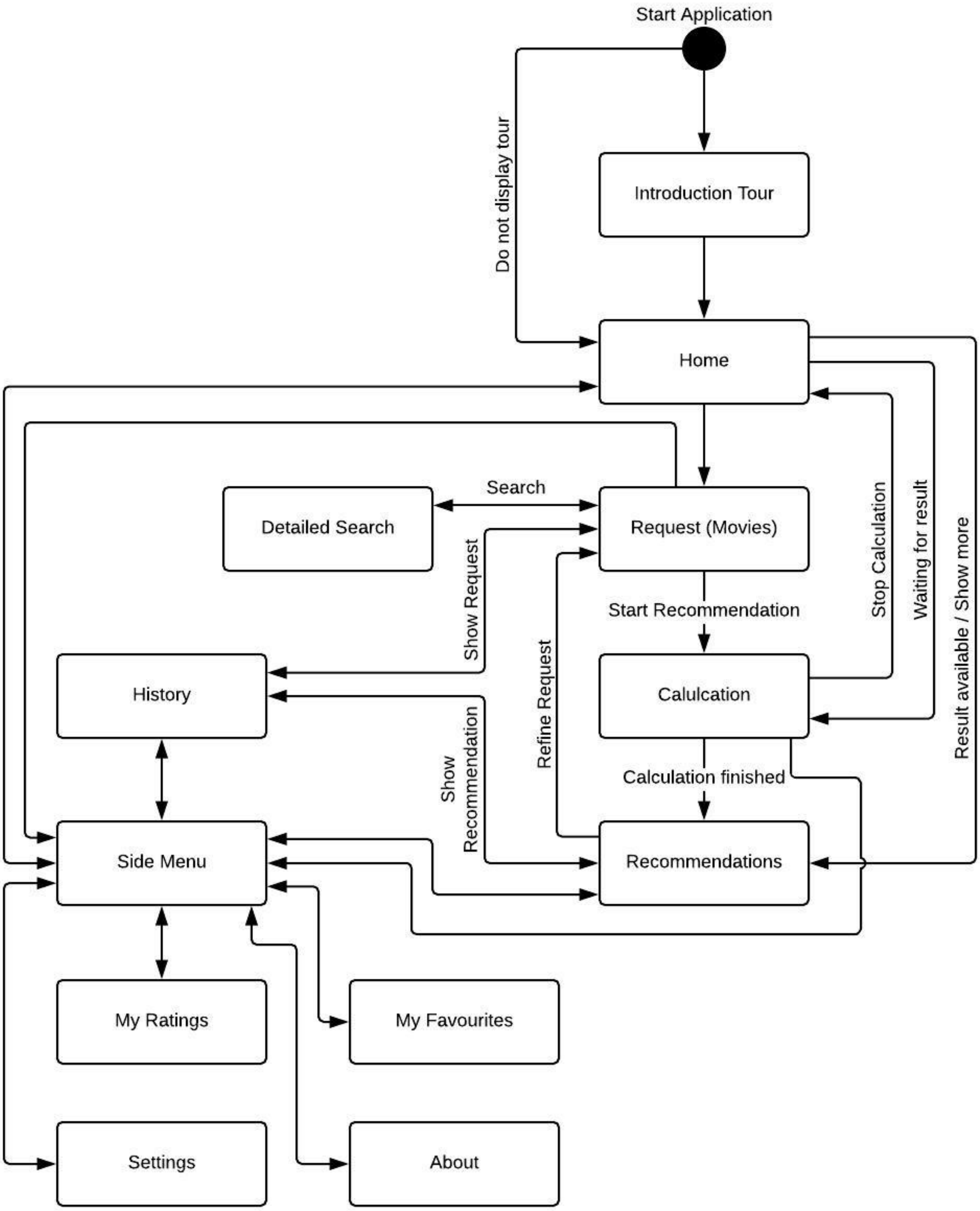


Figure 4.6.: Page flow diagram to show the connections between each page.

4.2. Page Design

The following section describes the design of each page. I will discuss the important UI elements and explain the connections between them.

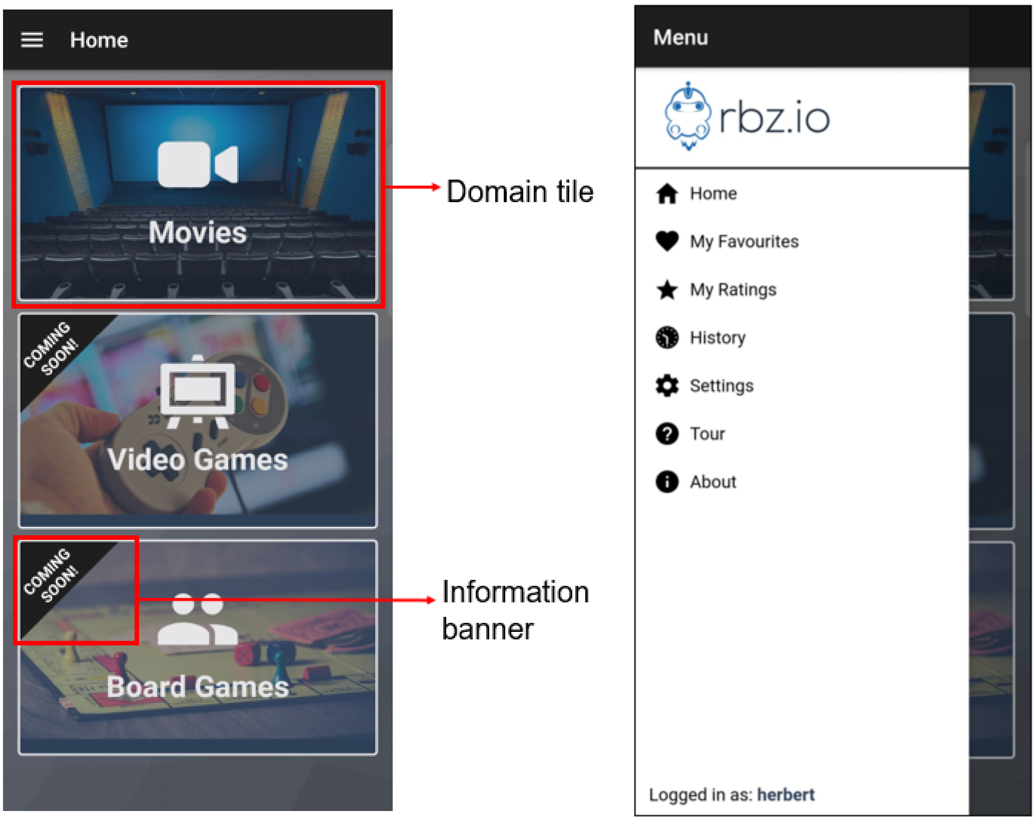
4.2.1. Home

The home page is used to select the domain. It is divided into tiles as shown in Figure 4.7(a). This provides a good overview of the different domains. Also it is easy to expand with another domain. A tile is able to show the user information about the status of a recommendation computation. This information is displayed as a banner in the upper left corner of a tile. Following banners are available:

- Waiting for result!
- Waiting show more!
- Result available!
- Coming soon!

4.2.2. Menu

The menu is implemented as a side menu, which can be opened on mostly every page with the burger button on the left side of the header. On top of the side menu, the logo is provided. Underneath, the individual pages can be accessed. To get a nice design, an icon on the left of the page text is displayed. Furthermore, if an user is logged in, the name of the user is displayed on the bottom of the menu. Figure 4.7(b) illustrated the side menu with a user, which is logged in.



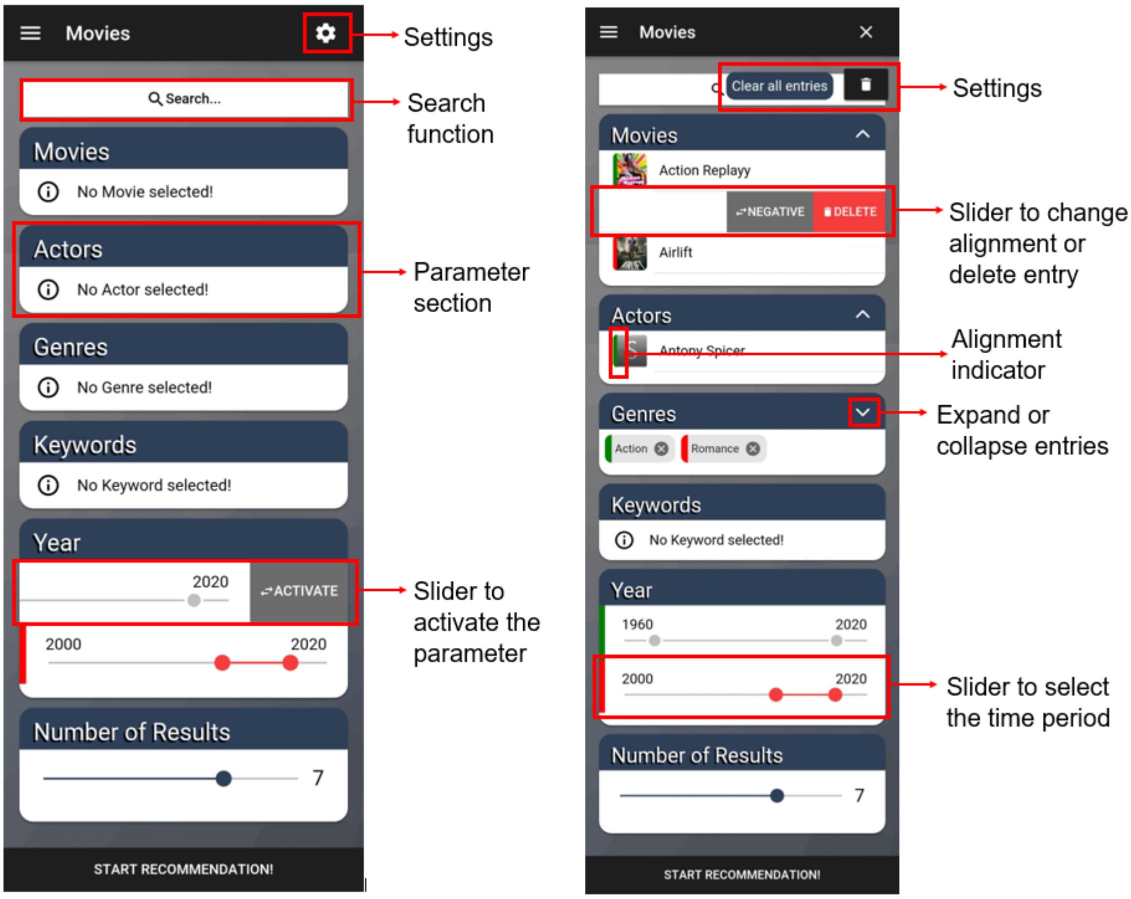
(a) Home page with different domains and tiles with an information banner

(b) Side menu with logged in user

Figure 4.7.: Home page with different domains and tiles with an information banner. Also the side menu is opened.

4.2.3. Movies

This page is used to create a search query with different parameters for the movie domain. This query is forwarded to the recommender engine. To give the user an overview of the different parameters, each of these are displayed as shown in Figure 4.8. If there is no entry for an parameter, an information is provided. To insert entries, a search function on top of the page is implemented, which leads to the detailed search page. Also a function to clear all entries is provided in the settings section. Each parameter section can contain one or multiple entries, which are displayed in a list. An entry is structured as follows: on the left side, a picture with an alignment indicator is shown. If the alignment indicator is green, the user likes the entry, otherwise it is red and the user dislikes it. Next to the picture, the name of the entry is shown. To change the alignment or to delete the entry, the entry can be slid to the left, where these two options are displayed. Each parameter section is sorted after the alignment, which means, that the entries which the user likes are displayed before the others. If the alignment changes, the list is re-sorted. In addition, there is the possibility to collapse the list to a simple chips view to a get a better view of the entries. In this view, the entries can only be deleted. A special case is the year parameter. This parameter can be activated and deactivated by sliding it to the left. After activation, the time period can be selected by using another slider. On the bottom of the page there is the "start recommendation" button, which leads to the computation page.



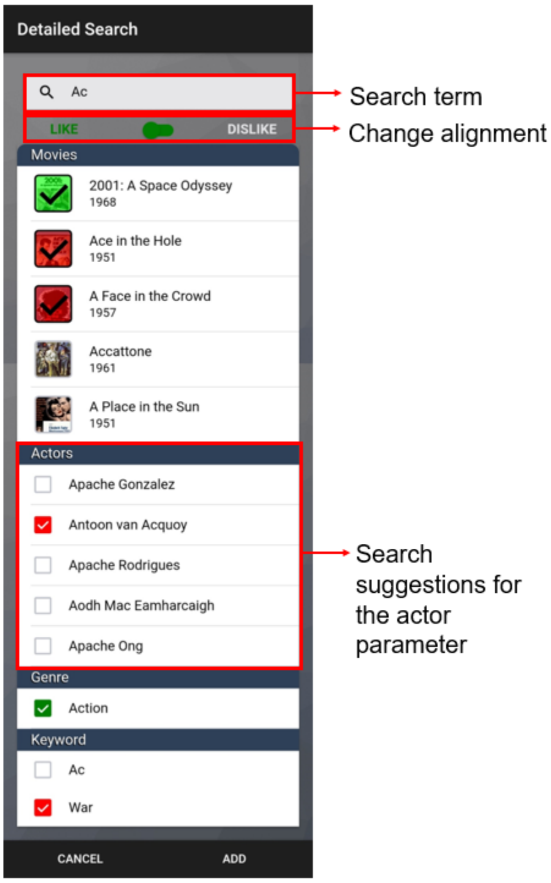
(a) Empty movie query

(b) Movie query with some entries

Figure 4.8.: Movie search query with and without entries.

4.2.4. Detailed Search

This page is for searching and adding parameters to the search query. The search bar is on top of the page. Just below there is the toggle to choose the alignment. Further down, suggestions for each individual parameter follow. A maximum of five entries is displayed per parameter. If the desired entry is not displayed, the search term has to be more specific. To select an entry, the user has to tick the checkbox. The movies parameter shows the poster for each entry, which should help to easily find the correct movie. The keyword parameter reflects the search term. If the entry is selected, it is stored in the keyword list and it is displayed permanently. At the bottom of the page, an add and a cancel button is provided to get back to the search query view. The add button stores the selected entries, the cancel button discards them.



Ac

Movies

2001: A Space Odyssey
1968

Ace in the Hole
1951

A Face in the Crowd
1957

Accattone
1961

A Place in the Sun
1951

Actors

□ Apache Gonzalez

☑ Antoon van Acquoy

□ Apache Rodrigues

□ Aodh Mac Eamharcaigh

□ Apache Ong

Genre

☑ Action

Keyword

□ Ac

☑ War

Search term

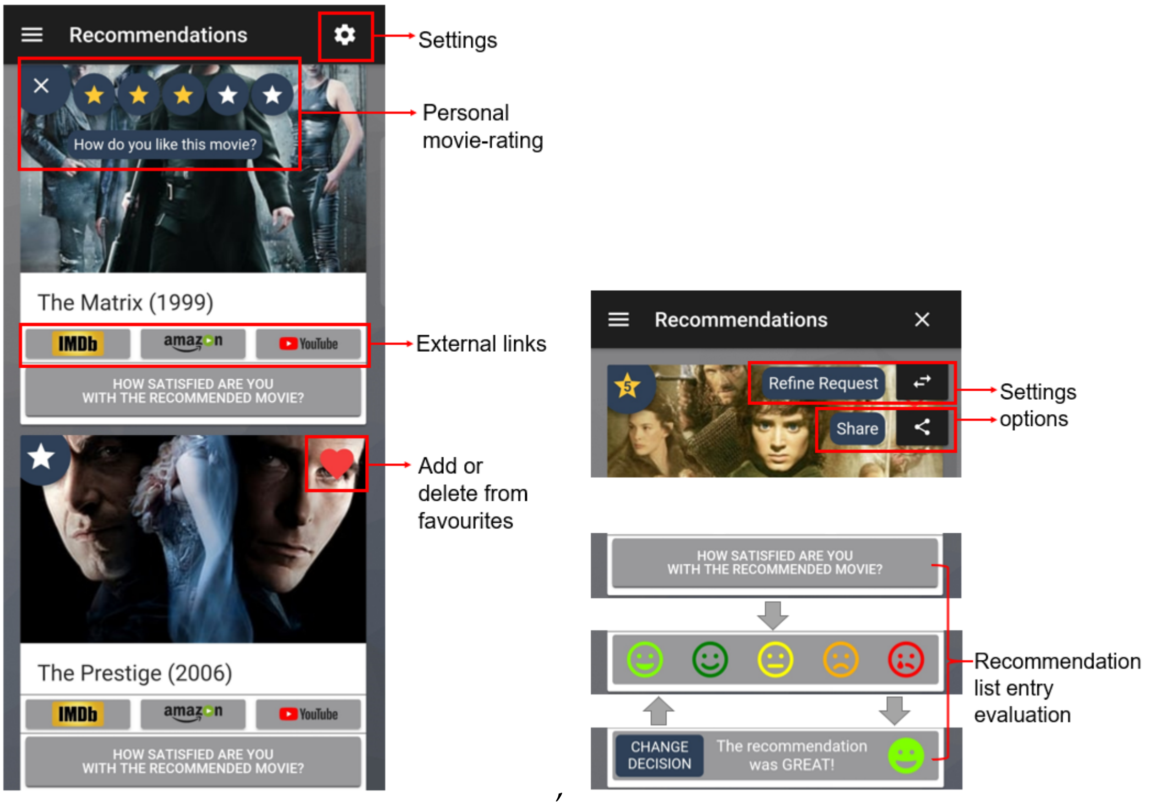
Change alignment

Search suggestions for the actor parameter

Figure 4.9.: Detailed search screen with search term "ac" and descriptions

4.2.5. Recommendations

On this page, the computed recommendation list is displayed. The header includes a setting function, where different options, like to share the result on social accounts and to refine the request, are available. This is shown in Figure 4.10(b). Each entry in the list consists of different parts. As shown in Figure 4.10(a), in the upper left corner of an entry there is the ratings function, in the upper right corner there is the add to favourites function. Below the poster, the title and year of the entry are displayed. Three different external links are provided in the middle of the entry. The button for the entry evaluation is at the bottom of an entry. Figure 4.10(b) shows the three possible designs of this button, regarding of the state.


Recommendations
Settings
How do you like this movie?
Personal movie-rating
The Matrix (1999)
IMDb
amazon
YouTube
External links
HOW SATISFIED ARE YOU WITH THE RECOMMENDED MOVIE?
Add or delete from favourites
The Prestige (2006)
IMDb
amazon
YouTube
HOW SATISFIED ARE YOU WITH THE RECOMMENDED MOVIE?
Recommendations
Refine Request
Share
Settings options
HOW SATISFIED ARE YOU WITH THE RECOMMENDED MOVIE?
Recommendation list entry evaluation
CHANGE DECISION
The recommendation was GREAT!
,

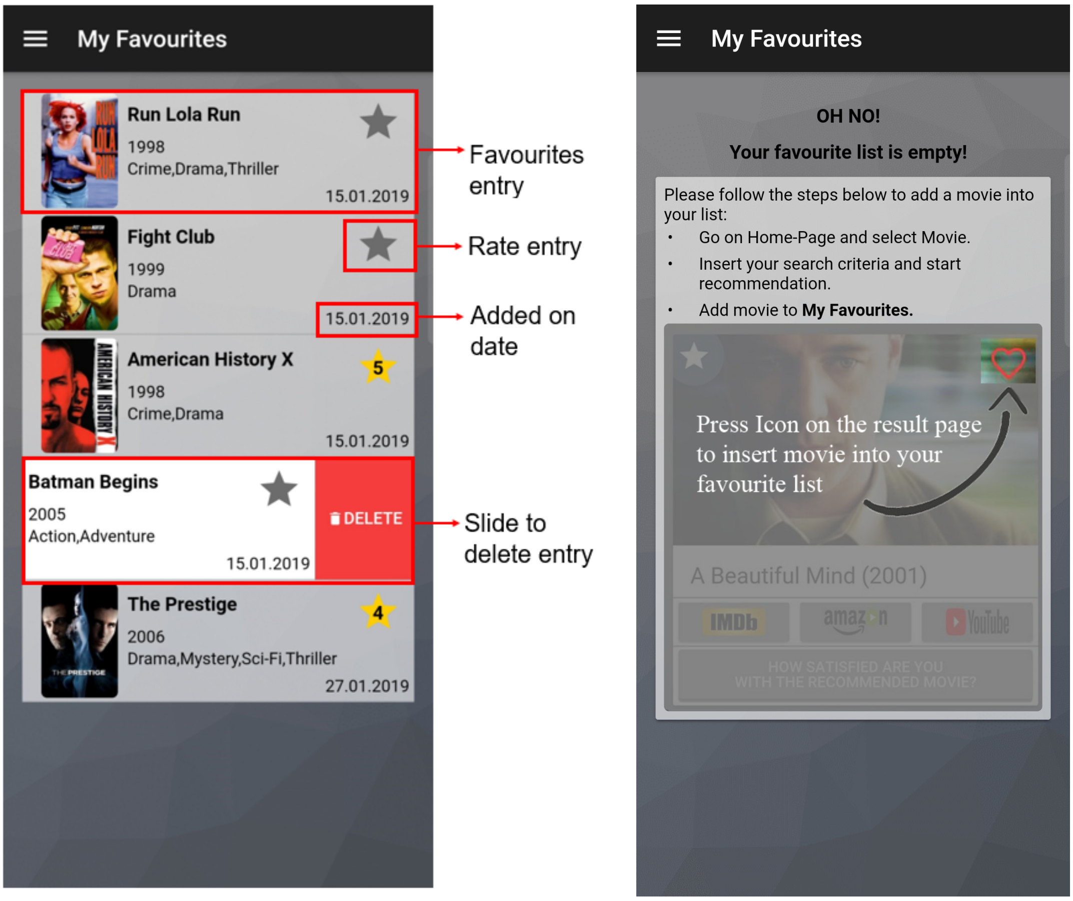
(a) Result entries with description

(b) Possible settings options and flow of the entry evaluation

Figure 4.10.: Recommendations screen displays list with results

4.2.6. My Favourites

This page displays all entries, which are added to the favourites list. If this list is empty, a help information is displayed as shown in Figure 4.11(b). Otherwise the entries are displayed as illustrated in Figure 4.11(a). Each entry consists of the name, the year, an added-on date and additional data depending on the domain. In the upper right corner of an entry, the ratings functionality is provided. To delete an entry, it has to be slid to the left and the delete function is displayed.



My Favourites

Run Lola Run
1998
Crime,Drama,Thriller
15.01.2019

Fight Club
1999
Drama
15.01.2019

American History X
1998
Crime,Drama
5
15.01.2019

Batman Begins
2005
Action,Adventure
15.01.2019
DELETE

The Prestige
2006
Drama,Mystery,Sci-Fi,Thriller
4
27.01.2019

Favourites entry

Rate entry

Added on date

Slide to delete entry

My Favourites

OH NO!

Your favourite list is empty!

Please follow the steps below to add a movie into your list:

- Go on Home-Page and select Movie.
- Insert your search criteria and start recommendation.
- Add movie to **My Favourites.**

Press Icon on the result page to insert movie into your favourite list

A Beautiful Mind (2001)

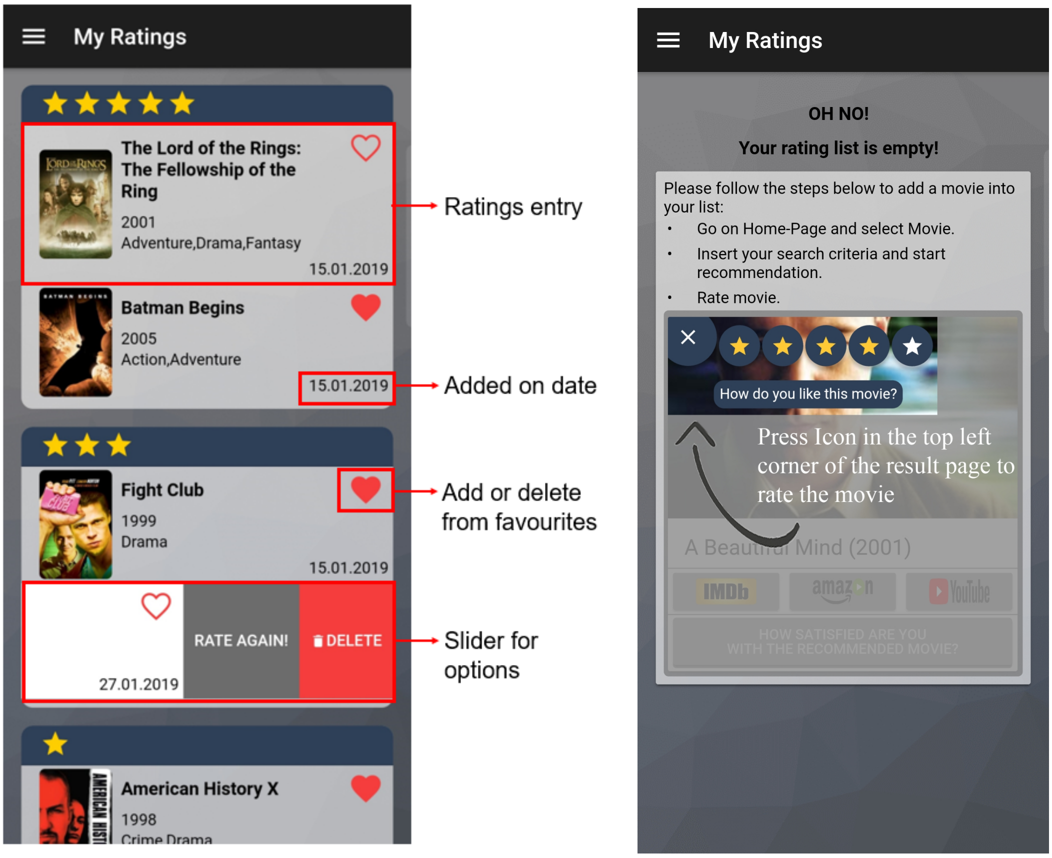
(a) Explanation of My Favourite list functions

(b) Empty favourite list

Figure 4.11.: My favourite screen with and without entries

4.2.7. My Ratings

The My Ratings page displays all rating list entries grouped by their rated score. If this list is empty, a help information is displayed as shown in Figure 4.12(b). Otherwise the entries are displayed as shown in Figure 4.12(a). Each entry consists of the name, the year, an added-on date and additional data depending on the domain. In the upper right corner, the icon for adding the entry to the favourites list is provided. To delete the entry or make the rating again, the entry has to be sled to the left and these options are shown.



OH NO!

Your rating list is empty!

Please follow the steps below to add a movie into your list:

- Go on Home-Page and select Movie.
- Insert your search criteria and start recommendation.
- Rate movie.

How do you like this movie?
A Beautiful Mind (2001)
IMDb
amazon
YouTube
HOW SATISFIED ARE YOU WITH THE RECOMMENDED MOVIE?

Press Icon in the top left corner of the result page to rate the movie

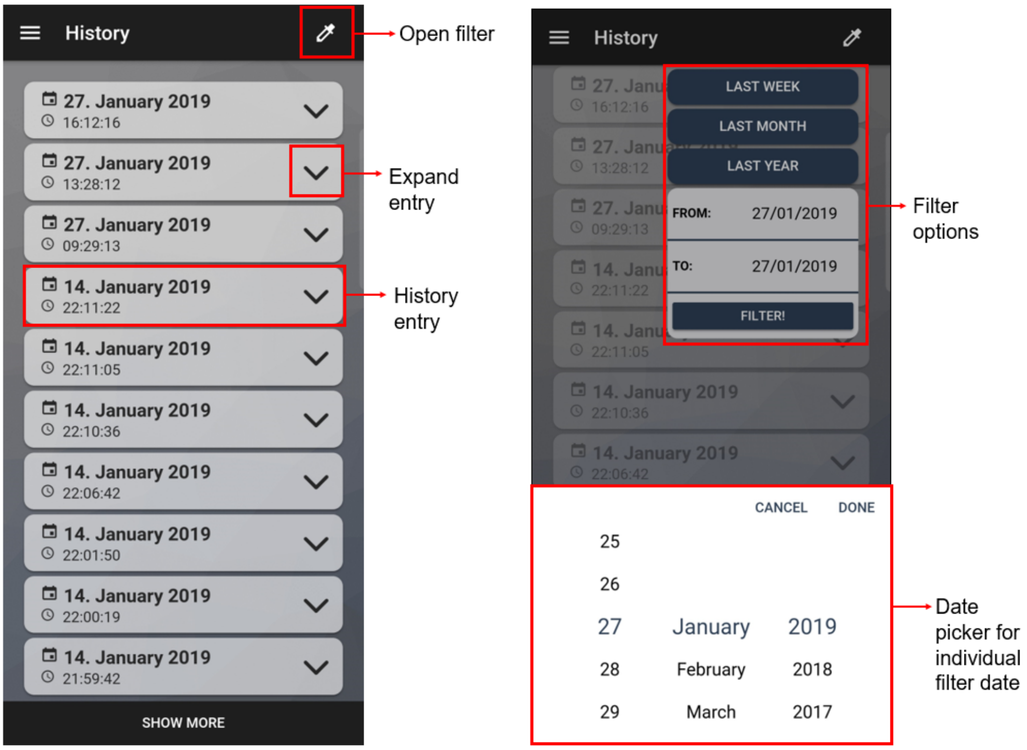
(a) Rating entries with options

(b) Empty ratings list

Figure 4.12.: My Ratings shows entries grouped by the rating score

4.2.8. History

The history page displays old search requests with their result list. It is sorted by date as shown in Figure 4.13(a). Each entry can be expanded to show the details as shown in Figure 4.14. The details view provides all information about the request. Additionally, a horizontal slider shows a preview of the recommendation result list. This preview shows if an entry was added to the favourite list and if an entry was already rated. Furthermore, it displays if an entry was already evaluated. At the end of the details view two buttons are provided. One forwards the user to the request, where he can refine it, whereas the other one forwards the user to the recommendation result list. As additional feature, the history page provides a filter functionality, where the entries can be filtered by date. There, three predefined time periods are provided, also a user defined time period can be chosen as illustrated in Figure 4.13(b).



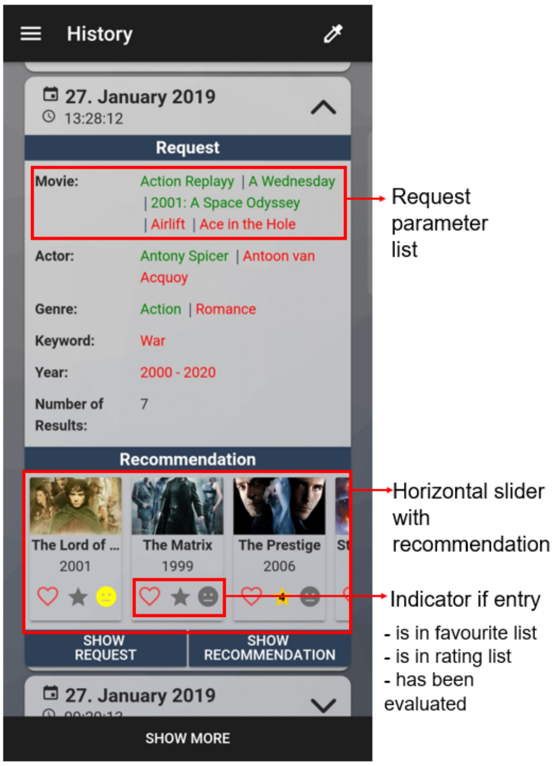
FROM: 27/01/2019

TO: 27/01/2019

(a) Collapsed history entries

(b) Filter functionality

Figure 4.13.: History entries with filter functionality



27. January 2019
13:28:12

Request

Movie: Action Replayy | A Wednesday | 2001: A Space Odyssey | Airlift | Ace in the Hole

Actor: Antony Spicer | Antoon van Acquoy

Genre: Action | Romance

Keyword: War

Year: 2000 - 2020

Number of Results: 7

Recommendation

The Lord of ... 2001

The Matrix 1999

The Prestige 2006

Request parameter list

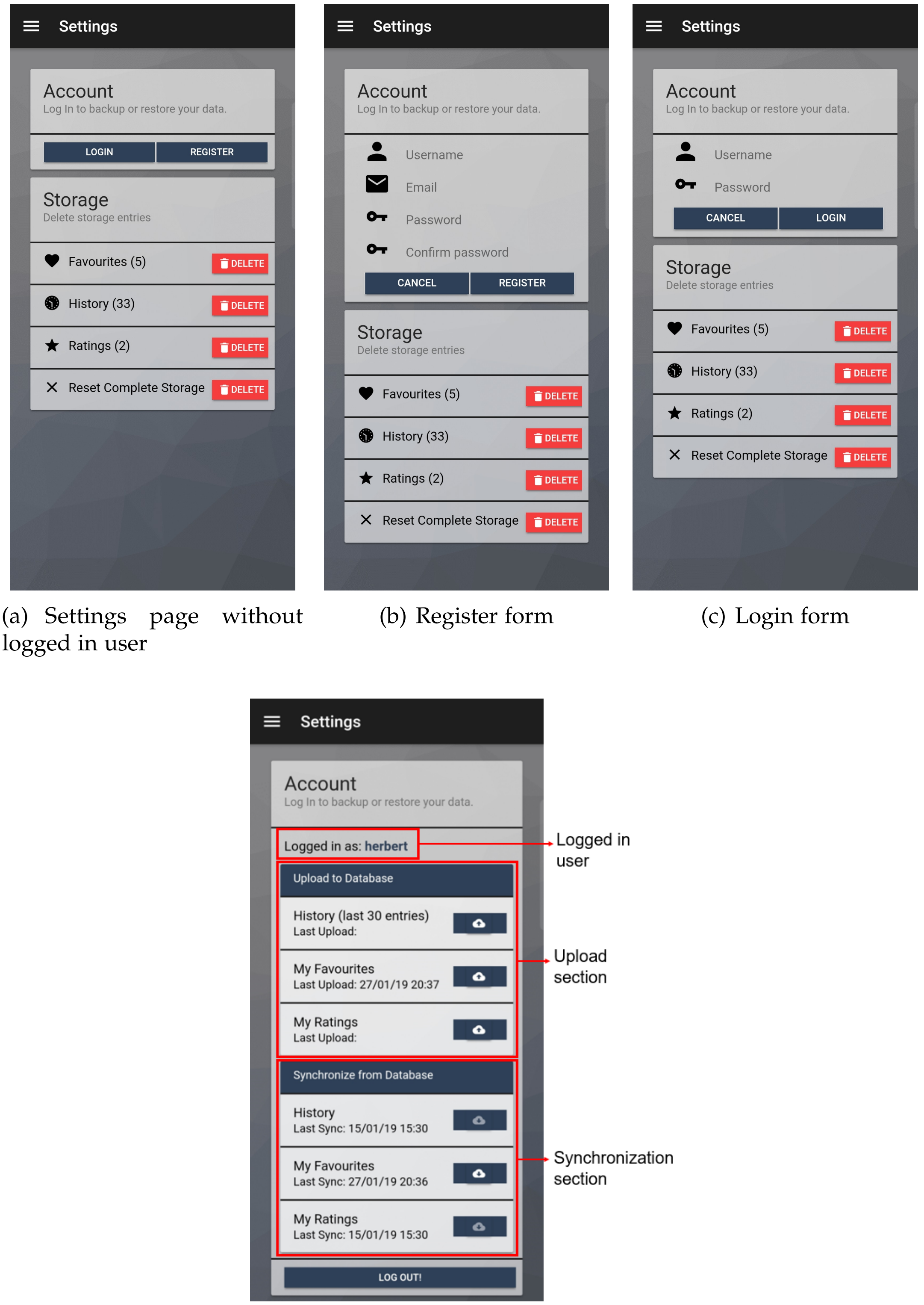
Horizontal slider with recommendation

Indicator if entry
- is in favourite list
- is in rating list
- has been evaluated

Figure 4.14.: Expanded history entry

4.2.9. Settings

The settings page is divided into two sections as illustrated in Figure 4.15(a). The first section provides the account management. There, the user is able to register himself (Figure 4.15(b)). Also he is able to login with existing user data (Figure 4.15(c)). If incorrect data has been entered, error messages are displayed. After the user is logged in, he is able to backup and synchronize his personal data. History, My Favourites and My Ratings can be separately treated (Figure 4.15(d)). At the end of this section, a log out button is provided, but only if the user is logged in.
The second section provides the storage management. There, the user is able to delete the entries in the storage separately or he can delete the complete storage. To avoid accidental deletion, a pop-up warns the user of the consequences.



Settings

Account
Log In to backup or restore your data.

LOGIN | REGISTER

Storage
Delete storage entries

Favourites (5) DELETE

History (33) DELETE

Ratings (2) DELETE

Reset Complete Storage DELETE

(a) Settings page without logged in user

Settings

Account
Log In to backup or restore your data.

Username

Email

Password

Confirm password

CANCEL | REGISTER

Storage
Delete storage entries

Favourites (5) DELETE

History (33) DELETE

Ratings (2) DELETE

Reset Complete Storage DELETE

(b) Register form

Settings

Account
Log In to backup or restore your data.

Username

Password

CANCEL | LOGIN

Storage
Delete storage entries

Favourites (5) DELETE

History (33) DELETE

Ratings (2) DELETE

Reset Complete Storage DELETE

(c) Login form

Settings

Account
Log In to backup or restore your data.

Logged in as: herbert

Upload to Database

History (last 30 entries)
Last Upload:

My Favourites
Last Upload: 27/01/19 20:37

My Ratings
Last Upload:

Synchronize from Database

History
Last Sync: 15/01/19 15:30

My Favourites
Last Sync: 27/01/19 20:36

My Ratings
Last Sync: 15/01/19 15:30

LOG OUT!

Logged in user

Upload section

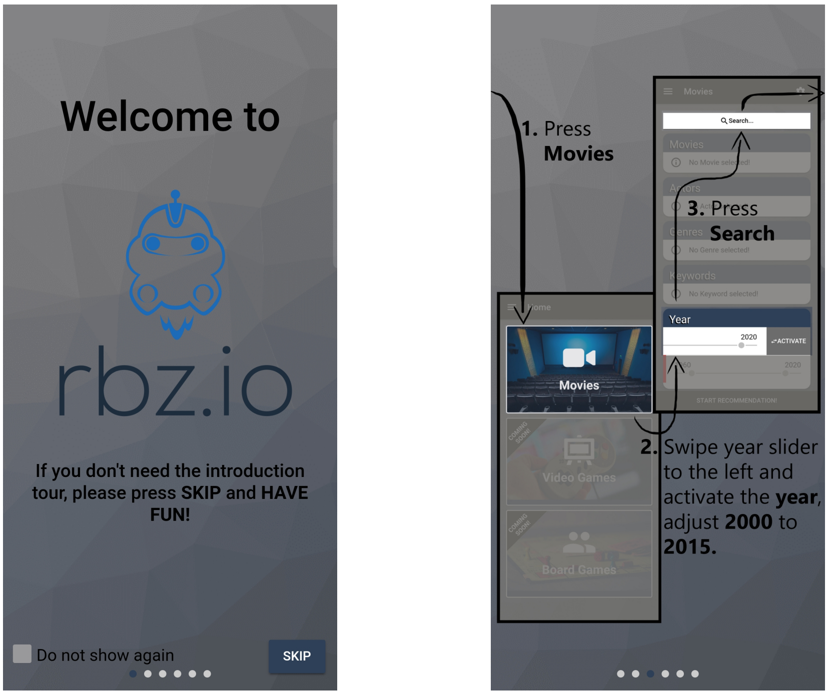
Synchronization section

(d) Restore and synchronize section when user is logged in.

Figure 4.15.: Different views of the settings page.

4.2.10. Introduction Tour

The introduction tour is shown at the launch of the application. As illustrated in Figure 4.16(a), it can be deactivated with the "Do not show again" checkbox or it can be skipped to get to the home page immediately. The introduction tour should help the user to easily understand how the applications works and which features are available. It is implemented with a step by step path as shown in Figure 4.16(b). This should guide the user to the simplest way to get a result.



Welcome to

rbz.io

If you don't need the introduction tour, please press SKIP and HAVE FUN!

□ Do not show again

1. Press **Movies**

2. Swipe year slider to the left and activate the **year**, adjust **2000** to **2015.**

3. Press **Search**

(a) Start page with "Do not show again" function

(b) First page of the step by step guide

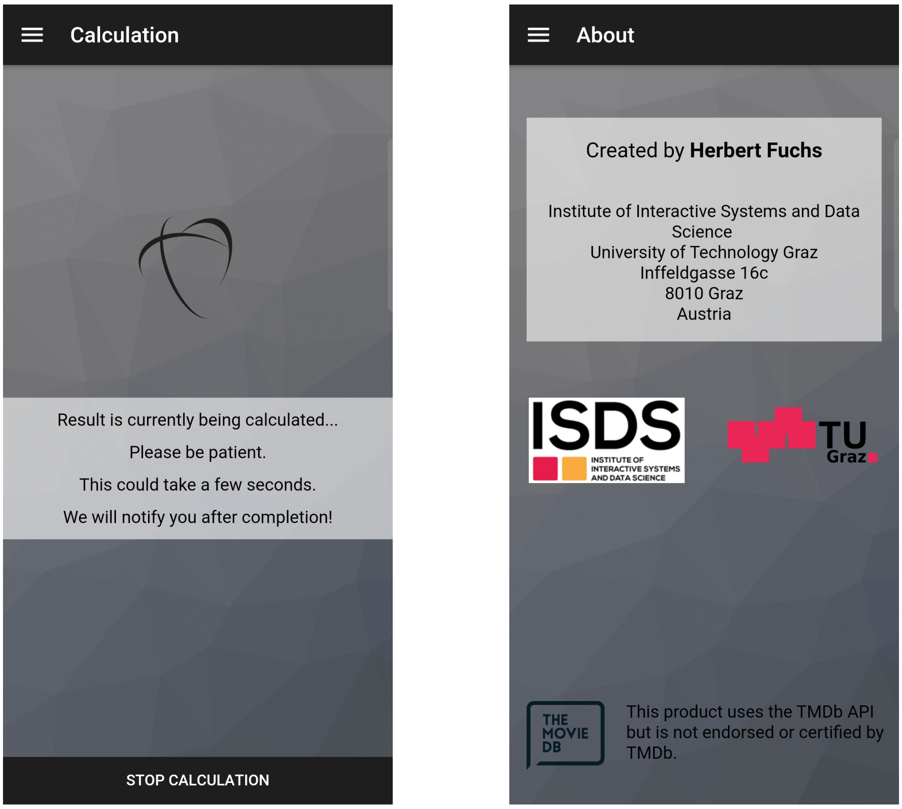
Figure 4.16.: Part of the introduction tour to show the basic principle

4.2.11. Computation

This page is shown during the computation of a recommendation list. It should inform the user with a spinner that the computation is in progress as illustrated in Figure 4.17(a). Also it is possible to stop the computation of the recommendation.

4.2.12. About

This page displays the author of the application as shown in Figure 4.17(b). Furthermore, licenses of different third party libraries and plugins are displayed.



Result is currently being calculated...
Please be patient.
This could take a few seconds.
We will notify you after completion!

Created by **Herbert Fuchs**

Institute of Interactive Systems and Data Science
University of Technology Graz
Inffeldgasse 16c
8010 Graz
Austria

This product uses the TMDb API but is not endorsed or certified by TMDb.

(a) The computation page shows a spinner, to indicate that the recommendation is currently being computed.

(b) The about page displays information about the creator.

Figure 4.17.: Computation page and about page.

5. Evaluation

This chapter explains the evaluation procedure of the application. A case study with 10 participants was performed to evaluate the mobile application. They performed several tasks and afterwards they had to answer a questionnaire. This feedback has been elaborated and the result identified the scale of usability of the application and potential weaknesses.

The structure and the experimental setup of the case study is described in Section 5.1. The description of the tasks and the questionnaire is shown in Section 5.1.1 and Section 5.1.2. At the end in Section 5.2, I provide the result of the case study and discuss the outcomes of it.

5.1. Case Study

A case study is a procedure where participants have to perform pre-defined tasks. It will be observed how the user completes a task. It is of interest, whether the user completes the task according to the planned path or whether he navigates with another route to the goal. Afterwards, he has to fill out a questionnaire.

Our user study was performed on a Samsung S8 smartphone with Android 8.0.0 as its platform. The participants were not briefed before. They had to perform nine tasks. The instruction sheet with the task can be found in Figure A.1. Afterwards, they had to answer some personal questions, two open post-test questions and 10 questions, which were evaluated with the System Usability Scale (SUS). This questionnaire can be found in Figure A.2 and Figure A.3.

The aim of this study is to check if all developed features will be used as intended and if the user thinks if this application is practical and useful. Additionally, weaknesses in the system should be detected and it should provide information whether the application is self-explanatory or not.

5.1.1. Tasks

The tasks should cover a large part of the provided features and it should be possible to complete all tasks without help. For this reason, in the first task, the participant has to start the application and he has to finish the introduction tour. In the following tasks, the participant has to create and refine a movie request with predefined parameters. With this tasks, the usability of the search and query page gets evaluated.
Furthermore, the participant has to add movies to the favourites list and rate different movies. Afterwards, he has to go to the home page. Then he has to change the rating of a movie, which he has rated before. This task focuses on the usability of navigation between the different features.
In the last three tasks, the participant has to backup the history, delete it afterwards, and restore it. He has to perform all necessary tasks. These tasks focus on the comprehensibility of the help texts in the application. In addition, the time how long the user needs for the tasks is recorded. To ensure that the participants solve the tasks carefully and do not feel any time pressure, the participants are only informed after the study that the time has been recorded.

5.1.2. Questionnaire

The questionnaire should provide information about different areas. On the one hand, it should provide a statement about the usability, on the other hand it should find out the explicit strengths and weaknesses in the system. For that purpose, different sets of questions are used. In order to be able to make the statements about the participants, it includes personal questions about the them.
The following sections provide an overview of the different questions. Also a description what the System Usability Scale (SUS) is and how it can be applied, is given.

Personal Questions

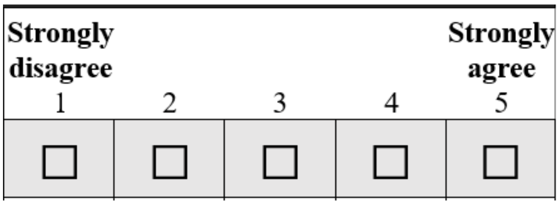
The personal questions should give an overview of the set of participants. The standard questions such as the name, the age and the gender are included. Furthermore, the questionnaire includes a question about the participant's computer knowledge level, which can range from "never used a computer" to "expert". Additionally, the highest level of completed education can be indicated.

SUS Questions

The System Usability Scale was developed in 1986 by Brooke [8] to create an end-of-test subjective assessments of usability. It was created as a quick and dirty solution for usability testing. It is based on the Likert Scale, which is described in the next section. The SUS should be performed after completion of the tasks and before the debriefing or other discussions [29]. The advantage of SUS is that it is cheap and it can be performed quickly. Additionally, it is possible to evaluate the user satisfaction. The disadvantage is, that it is not possible to give precise indications of weaknesses. In addition, no differentiation of the individual participants is possible.

Likert Scale

The Likert Scale was invented by Rensis Likert in 1932 [23]. It is a series of statements, which should indicate the feeling about a system or a product. The statements range from highly positive to highly negative, e.g from "strongly aggree" to "strongly disagree" which is shown in Figure 5.1. It is quiet common to use an 5- or 7- point scale. With the odd scale, the user has the possibility to give a neutral answer [28].



Strongly disagree 1	2	3	4	**Strongly agree** 5
□	□	□	□	□

Figure 5.1.: Five point Likert Scale

Implementation

The SUS uses ten statements, where five of these statements are positive (odd numbers) and five are negative (even numbers). The alternating order was invented, that the participant answers the statements correctly by reading them carefully. Brooke [8] introduced following statements for the usability test:

1. I think that I would like to use the system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.

6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

For answering the questions, a five point Likert Scale is used. Each statement can be rated from "strongly disagree" to "strongly agree". The participants should give immediate answers to the questions and should not think too long about each answer. Furthermore, if an question can not be answered, the centre point of the scale should be selected.

Calculation

The get the System Usability Score following calculations has to be done:

- For each of the odd statements, subtract 1 from the score.
- For each of the even statements, subtract their value from 5.
- Sum up previous calculated values to a total score.
- Multiply total score by 2.5.

This leads to an overall score in the range of 0 to 100, where 0 is the worst possible and 100 the best possible result.

Interpretation

The SUS score is not the percentage of the reached usability, it is the percentile. Sauro et al. [29] figured out with using 446 studies and 5000 individual responses, that the overall mean score of the SUS is 68. With this knowledge, he created a table, where he converted the raw score to a percentile rank as shown in Figure 5.2(a). The percentile rank indicates how usable the system is relative to the other products. For example, a raw SUS Score of 60 has a percentile rank of 29%, which means, that 71% of other products are more usable then the tested one. Everything with a SUS more than 68 has a percentile rank of 50%, which indicates that the usability is above average.

Furthermore, Sauro et al. [29] invented a grading scale, which differs from the known scale of Bangor et al. [6]. The scale of Bangor et al. [6] is a traditional school grade scale, where e.g. a SUS between 90 and 100 is an A and a score below 60 is an F. Sauro et al. [29] states, that with this scale it is virtually impossible to get an A, because in his study, less than 1% had a mean SUS above 90. To provide a fairer grading assignment, he developed the curved grading scale shown in Figure 5.2(b).

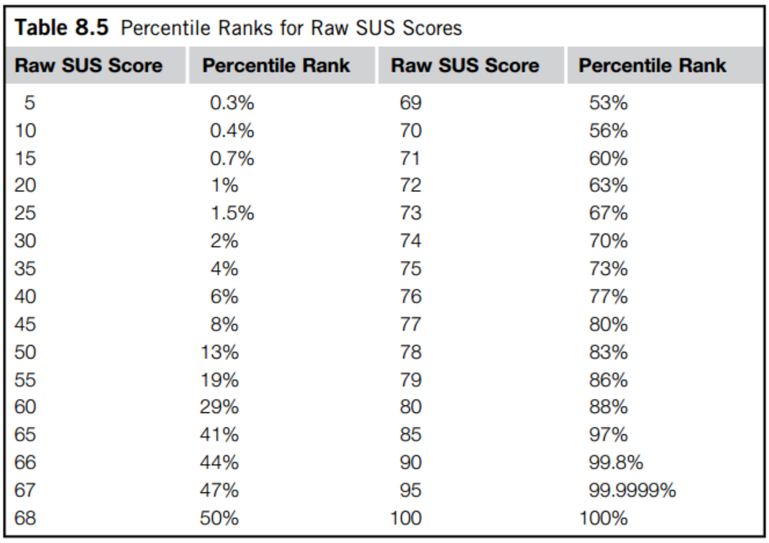


Table 8.5 Percentile Ranks for Raw SUS Scores

Raw SUS Score	Percentile Rank	Raw SUS Score	Percentile Rank
5	0.3%	69	53%
10	0.4%	70	56%
15	0.7%	71	60%
20	1%	72	63%
25	1.5%	73	67%
30	2%	74	70%
35	4%	75	73%
40	6%	76	77%
45	8%	77	80%
50	13%	78	83%
55	19%	79	86%
60	29%	80	88%
65	41%	85	97%
66	44%	90	99.8%
67	47%	95	99.9999%
68	50%	100	100%

(a) Percentile Ranks for Raw SUS Scores

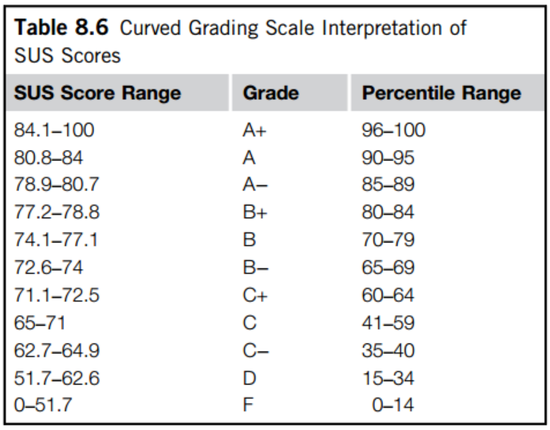


Table 8.6 Curved Grading Scale Interpretation of SUS Scores

SUS Score Range	Grade	Percentile Range
84.1–100	A+	96–100
80.8–84	A	90–95
78.9–80.7	A–	85–89
77.2–78.8	B+	80–84
74.1–77.1	B	70–79
72.6–74	B–	65–69
71.1–72.5	C+	60–64
65–71	C	41–59
62.7–64.9	C–	35–40
51.7–62.6	D	15–34
0–51.7	F	0–14

(b) Curved Grading Scale Interpretation of SUS Scores

Figure 5.2.: System Usabilty Score evaluation tables from [29]

Post-test Questions

The open post-test questions are to find out explicit strengths and weaknesses in the system. They should be asked at the end, as the user should think about the whole application again. The two questions are:

- What are two things that you really liked?
- What are two things that you did not like?

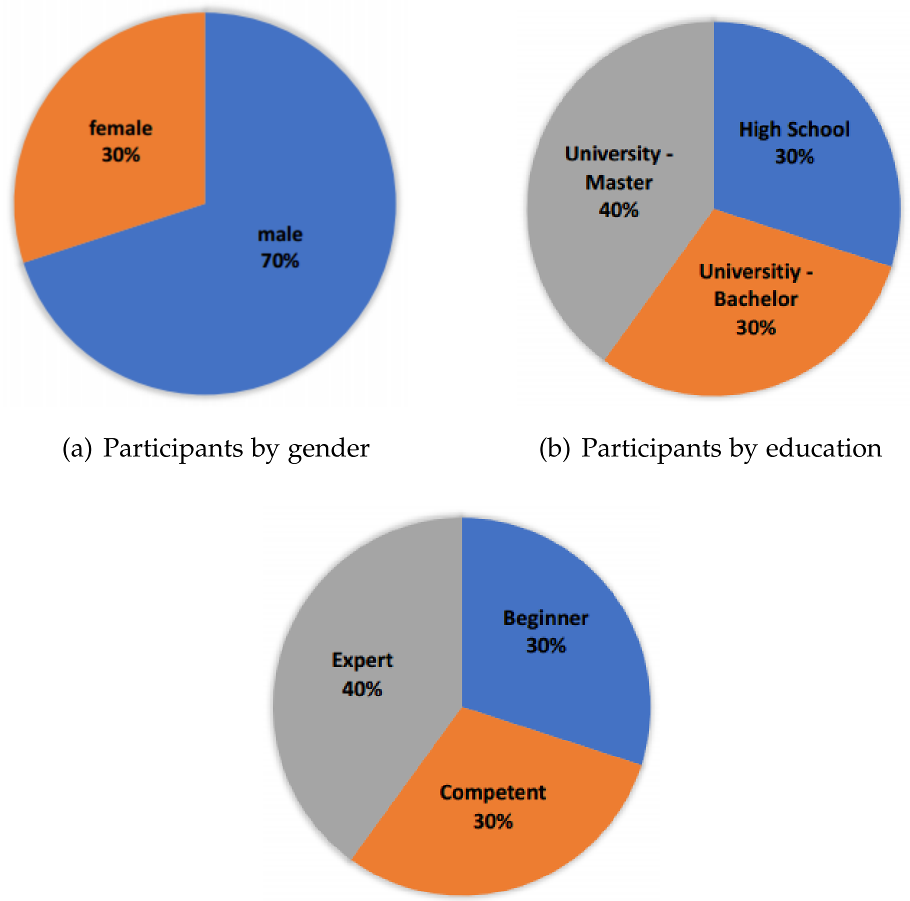
The first question should bring the participant in a positive mood, by thinking about things in the application, which worked well. The second question should indicate serious errors. By asking only two things, the participant get forced to prioritize his decision. This should lead to a better quality of the error report [31].

5.2. Results

In this section I discuss the outcoming result of the case study. At the beginning, I analyze and categorize the set of participants according to their characteristic. Afterwards, I evaluate the SUS questionnaire and present the result. At the end of the section, I discuss the open post-test questions and highlight the strengths and weaknesses of the application.

5.2.1. Participants

A total of ten candidates took part in the study. The proportion of women was 30%, while that of men was 70% as shown in Figure 5.3(a). The participants were aged between 24 and 29. All participants had at least the high school diploma and even 70% had an university degree (Figure 5.3(b)). The participants had to assess their computer knowledge themselves. 40% considered themselves to be experts, 30% as competent users and 30% as beginners, which is illustrated in Figure 5.3(c).



(a) Participants by gender

(b) Participants by education

(c) Participants by computer knowledge

Figure 5.3.: Graphical evaluation of the participants by gender, education and computer knowledge.

5.2.2. SUS Score

The average SUS score is 82,75 out of 100, which is in the grading scale of Sauro et al. [29] an **A**. The minimal score given by a user is 75, which is a **B** in the grading scale. The best score is 95, which is an **A+**. These different scores are shown in Figure 5.5 and the overall evaluation of each question is shown in Table 5.1. As illustrated in Figure 5.4, the average SUS score of 82,75 corresponds to a percentile rank of 94%. This means, that the application has a higher usability than 94% of all tested products. Therefore, I conclude that the participants considered the application as very usable.

Question	**1**	**2**	**3**	**4**	**5**	**Average SUS Score**
1. I think that I would like to use the system frequently.	1	0	3	4	5	6,5
2. I found the system unnecessarily complex.	4	4	1	1	0	7,75
3. I thought the system was easy to use..	0	1	3	4	2	6,75
4. I think that I would need the support of a technical person to be able to use this system.	7	2	1	0	0	9
5. I found the various functions in this system were well integrated.	0	0	0	5	5	8,75
6. I thought there was too much inconsistency in this system.	6	2	2	0	0	8,5
7. I would imagine that most people would learn to use this system very quickly.	0	0	0	5	5	8,75
8. I found the system very cumbersome to use.	8	1	1	0	0	9,25
9. I felt very confident using the system.	0	0	1	7	2	7,75
10. I needed to learn a lot of things before I could get going with this system.	9	1	0	0	0	9,75
Total SUS Score						**82,75**

Table 5.1.: Evaluation of the SUS questionnaire. The table indicates which answers how many participants gave to the questions. The average score is calculated with the rules from Section 5.1.2.

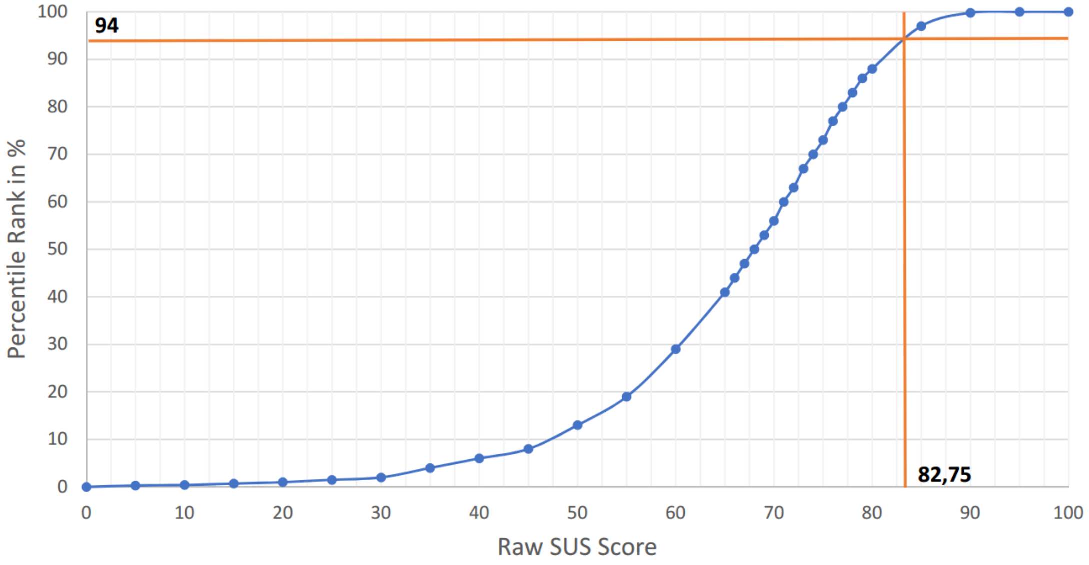


Figure 5.4.: Graphical representation of the percentile ranks for the raw SUS Score. The blue line shows the percentile ranks associate with the raw SUS score, the orange line shows the intersection point with the average SUS score.

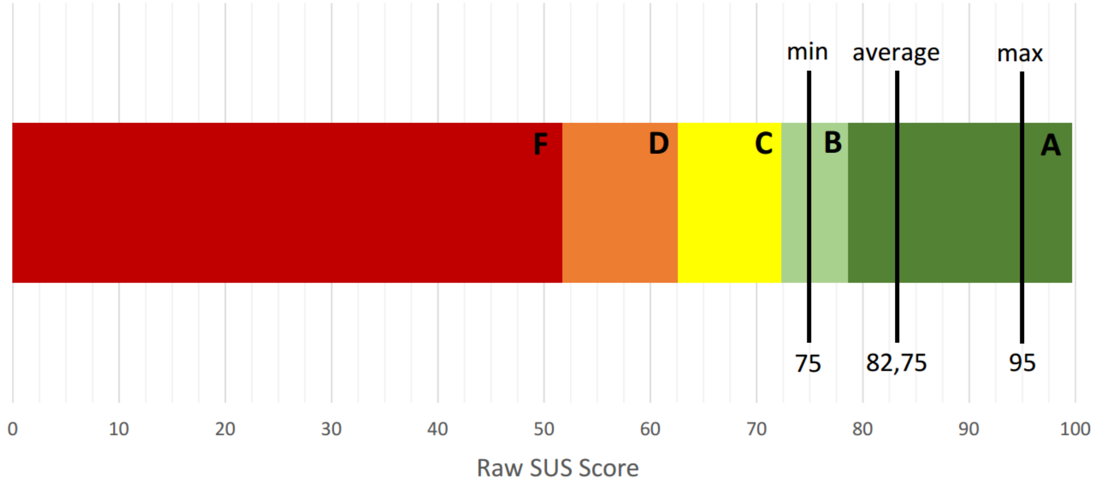


Figure 5.5.: Grading Scale of SUS score with minimal, average and maximal value.

5.2.3. Post-test Questions

All participants have answered the positive question completely. The negative question was fully answered by four participants, five participants only raised one critical point and only one user had no critical point.
50% of the participants stated that the design of the application was particularly appealing. Especially the structure with the side-menu and the tiles on the home screen were positively mentioned. Furthermore, the usability was emphasized. The participants particularly liked the fact that the functions are additionally described with icons and that an introduction tour is provided at the launch of the application. The backup system was also reported as very helpful.
60% of the participants did not like the sliding function, which enables the user to get access to entry based functions. Additionally, the "like-dislike" function on the search page was criticized by two participants, because they were not able to change the alignment after selecting an entry.
A non-unique result returns the general search page, where the user can search for all parameters. Two participants would prefer a separate search page for each parameter, whereas two other participants liked the functionality of a general search page.

5.2.4. Discussion

Participants

For more representative results, the participants have to be more diversified, which means that all different ages have to be included into the study and they should be evenly distributed. In my study, only one age group was included. Additionally, too few women were involved in the study. There should be an even distribution between women and men to reflect the Austrian population[1] [4, 3]. The participants were all highly educated, so people with lower education should also be included, because only 32,5% of the Austrian population [5] has such a high eduction as the participants (at least a high school diploma). All in all, the case study already shows the trend of the result and it provided insights about strengths and weaknesses of the application.

[1] The study took place in this country.

Tasks

The tasks were all successfully completed, but several smaller problems occurred during the execution. All tasks were completed in 15 minutes on average. The fastest participant needed 9 minutes, the slowest 19 minutes. The following list shows a summary of the problems and the different solution paths of the participants.

- **Task 1**: Sliding function of the introduction tour was not recognized.
- **Task 2**: To start the search function for a parameter, many participants tried to click on the corresponding item card. Furthermore, the given parameters were added individually, the multiple selection was not used. Another problem within the search function was changing the alignment from like to dislike. Here, the user first marked the parameter and after that, he changed the alignment, which does not work. The sliding function to activate the year parameter, was not found by any participant.
- **Task 3**: Occasionally, the rating icon was clicked first.
- **Task 4**: The functionality to refine the request was sometimes not found without a help. Many participants tried to click on the label of the "refine request" button, which closes the settings section and does not redirect them to the correct page. On the request page, only one participant used the slider to change the alignment, the others deleted the entry and added them again with the changed alignment.
- **Task 5**: All participants found the rating and feedback section on the result page.
- **Task 6**: Many participants tried to go to the movie request page again to get the rated movies. But after recognizing that this is the wrong path, they found the "My Ratings" page in the side-menu.
- **Task 7**: Most of the participants first went to the history page and clicked on the filter function. After a little hint, they found the back up function. The registration and login process worked well for everyone.
- **Task 8 and 9**: No problems occurred at these tasks.

Questions

All participants clearly understood the questions and answered these to the best of their conscience. The open post-test questions reflect very well the problems and the positive impressions that have occurred during the different tasks. These questions fulfilled their purpose and helped finding strengths and weaknesses of the application, which is not possible with the SUS questionnaire alone.

Result

The result of the case study shows that the mobile application is well structured and user-friendly. The feedback of the participants was quite positive and they would use the application frequently. The simple design, which is self-evident, was highlighted. Another good feedback given was that many different functions exist in the application to generate personal lists. In this case, the users meant the "My Ratings" and "My Favourite" list. Furthermore, the backup functions have convinced the participants and three participants explicitly stated that they think that the function is very useful.

The sliding function was mostly criticized because such a functionality is not available in other applications and therefore the participants did not find it. Many participants tried to get the functions, which are only available in the slider (delete, change entry), by a long- and a double press on the entry. But some said after the study that these functions become more understandable after repeated use. A further problem was how the participants tried to open the search function. They tried to click on the respective card of the search parameter to open it. An improvement of the application would be necessary by making these cards clickable. One participant noted that the order of the feedback smileys is twisted. According to him, the sequence should go from bad to good and not as in the application vice versa. Additionally, it was criticized that some functions are hard to find. However, this can often be traced back to the fact that the user did not read the introduction tour exactly, as it describes exactly those functions that he did not find.

6. Conclusion and Future Work

In this thesis a mobile application and a corresponding API for rbz.io was created and afterwards, a case study was performed to evaluate the user interface of the mobile application.
After implementing the mobile application, it can be stated that the appropriate technology for the application has been selected, because all defined requirements have been fulfilled. Through the use of the Ionic framework, creating a user interface which is structured and clean was very simple. In addition, it was a good choice to implement push notifications with OneSignal, because it was uncomplicated and this tool provides a powerful set of functions, which can later on be used to analyze users. A minor disadvantage, however, was that the Ionic framework was still in the beta phase. Therefore, some functions were not fully developed and a workaround had to be used. For example, when updating data in a pop-over window, the screen refresh had to be triggered by the system itself.

In the case study, it turned out that the slider function has not yet been accepted by the users. One reason for this could be, that the widely used Android and iOS applications currently rely more on the long press than on a sliding function. Furthermore, the study shows that when searching for functionalities, users click through the application before reading all the help texts on the page. Therefore, it is important during development to place functionalities clearly and easily on the screen, otherwise users will not find them.

In conclusion, the used technologies, as well as the performed case study worked well and only a few problems have occurred. Some usability improvements will need to be made in the future to make the application ready for the market, but for a first prototype, the outcoming result was successful.

Future work

The future work could include the improvement of the usability, in particular the problems encountered in the case study could be addressed. The slider of each entry could be replaced by a long press, the button labels and the

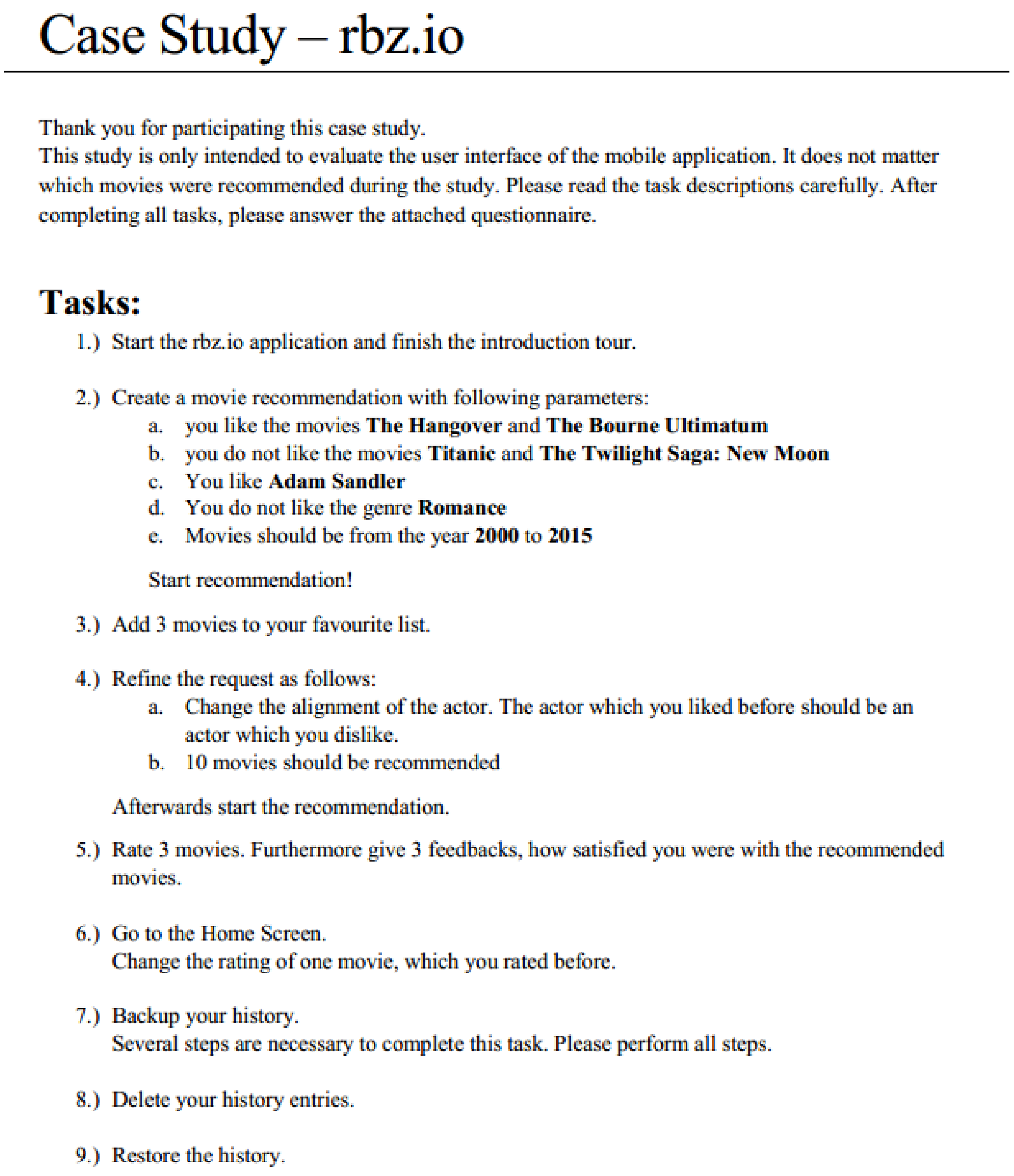
parameter cards on the query page could be made clickable.
In order not to make too many changes, one idea would be to design the introduction tour as an interactive tour. This forces the user to deal with the functionalities when starting the application and it can contribute to the understanding of the different functions. Furthermore, the functionalities of the domains "board games" and "video games" have to be designed and implemented.
For future research, it would be interesting to do a case study with a larger number of people, where each age group, gender and education is equally distributed. To get a representative result, each group should contain at least eight to ten persons in order to be able to make a uniform statement per group.

Appendix

Appendix A.

Case Study

In this appendix, I present the tasks and the questionnaire of the case study, which were used in the Chapter 5 to evaluate the mobile application.



Case Study – rbz.io

Thank you for participating this case study.
This study is only intended to evaluate the user interface of the mobile application. It does not matter which movies were recommended during the study. Please read the task descriptions carefully. After completing all tasks, please answer the attached questionnaire.

Tasks:

1.) Start the rbz.io application and finish the introduction tour.

2.) Create a movie recommendation with following parameters:
   a. you like the movies **The Hangover** and **The Bourne Ultimatum**
   b. you do not like the movies **Titanic** and **The Twilight Saga: New Moon**
   c. You like **Adam Sandler**
   d. You do not like the genre **Romance**
   e. Movies should be from the year **2000** to **2015**

   Start recommendation!

3.) Add 3 movies to your favourite list.

4.) Refine the request as follows:
   a. Change the alignment of the actor. The actor which you liked before should be an actor which you dislike.
   b. 10 movies should be recommended

   Afterwards start the recommendation.

5.) Rate 3 movies. Furthermore give 3 feedbacks, how satisfied you were with the recommended movies.

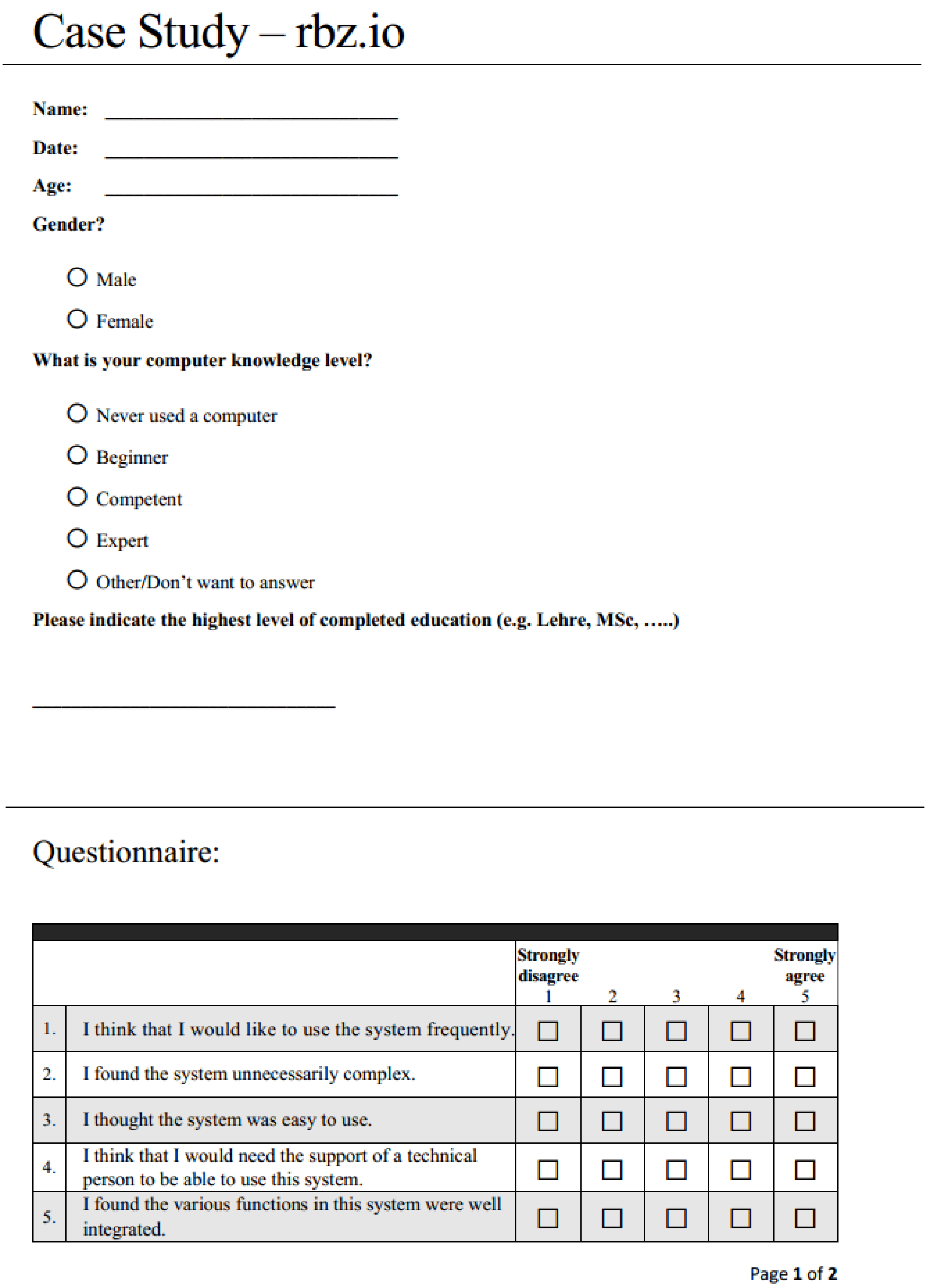
6.) Go to the Home Screen.
   Change the rating of one movie, which you rated before.

7.) Backup your history.
   Several steps are necessary to complete this task. Please perform all steps.

8.) Delete your history entries.

9.) Restore the history.

Figure A.1.: Tasks for case study



Case Study – rbz.io

Name: ______________________________

Date: ______________________________

Age: ______________________________

Gender?

○ Male

○ Female

What is your computer knowledge level?

○ Never used a computer

○ Beginner

○ Competent

○ Expert

○ Other/Don't want to answer

Please indicate the highest level of completed education (e.g. Lehre, MSc, …..)

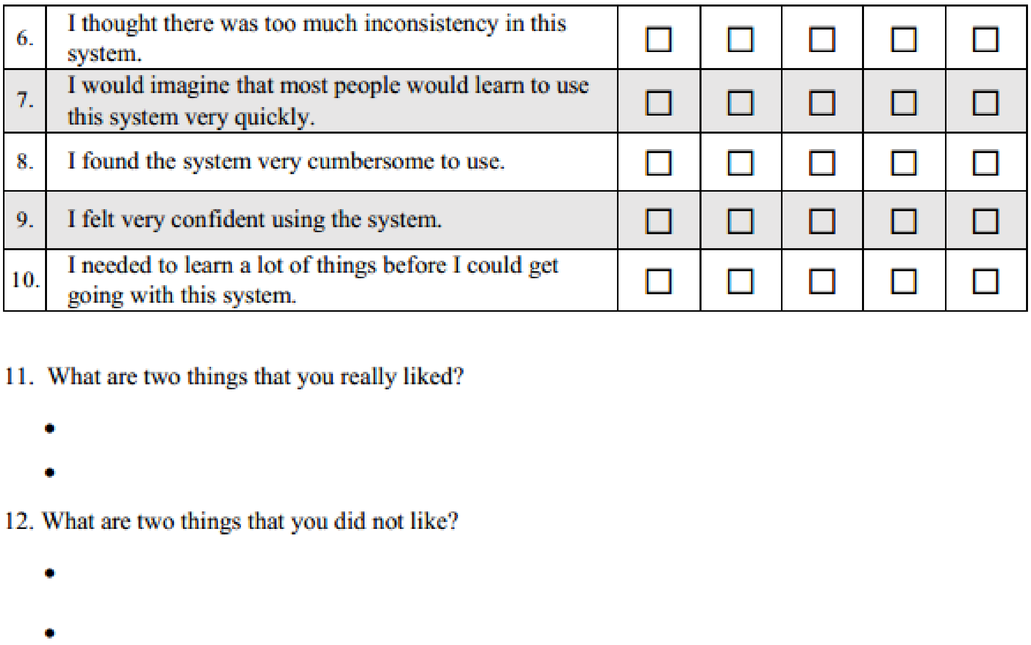
____

Questionnaire:

		Strongly disagree 1	2	3	4	Strongly agree 5
1.	I think that I would like to use the system frequently.	□	□	□	□	□
2.	I found the system unnecessarily complex.	□	□	□	□	□
3.	I thought the system was easy to use.	□	□	□	□	□
4.	I think that I would need the support of a technical person to be able to use this system.	□	□	□	□	□
5.	I found the various functions in this system were well integrated.	□	□	□	□	□

Page **1** of **2**

Figure A.2.: Questionnaire for case study part 1



6.	I thought there was too much inconsistency in this system.	□	□	□	□	□
7.	I would imagine that most people would learn to use this system very quickly.	□	□	□	□	□
8.	I found the system very cumbersome to use.	□	□	□	□	□
9.	I felt very confident using the system.	□	□	□	□	□
10.	I needed to learn a lot of things before I could get going with this system.	□	□	□	□	□

11. What are two things that you really liked?

-
-

12. What are two things that you did not like?

-
-

Figure A.3.: Questionnaire for case study part 2

Bibliography

[1] Angular. *Architecture overview*. Jan. 2019. URL: https://angular.io/guide/architecture (visited on 01/21/2019) (cit. on p. 21).

[2] B.Sindhya Anmol Khandeparkar Rashmi Gupta. "Introduction to Mobile Application Development Ecosystems." In: *International Journal of Computer Applications* (2015), pp. 31–33 (cit. on p. 6).

[3] Statistik Austria. *Bevölkerung am 1.1.2018 nach Alter und Bundesland - Frauen*. Jan. 2019. URL: https://www.statistik.at/web_de/statistiken/menschen_und_gesellschaft/bevoelkerung/bevoelkerungsstruktur/bevoelkerung_nach_alter_geschlecht/023472.html (visited on 02/10/2019) (cit. on p. 80).

[4] Statistik Austria. *Bevölkerung am 1.1.2018 nach Alter und Bundesland - Männer*. Jan. 2019. URL: https://www.statistik.at/web_de/statistiken/menschen_und_gesellschaft/bevoelkerung/bevoelkerungsstruktur/bevoelkerung_nach_alter_geschlecht/023471.html (visited on 02/10/2019) (cit. on p. 80).

[5] Statistik Austria. *Ergebnisse im Überblick: Bildungsstand*. Jan. 2019. URL: https://www.statistik.at/web_de/statistiken/menschen_und_gesellschaft/bildung_und_kultur/bildungsstand_der_bevoelkerung/020912.html (visited on 02/10/2019) (cit. on p. 80).

[6] Aaron Bangor, Philip Kortum, and James Miller. "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale." In: *J. Usability Studies* (May 2009), pp. 114–123 (cit. on p. 74).

[7] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice: Software Architect Practice_c3*. SEI Series in Software Engineering. Pearson Education, 2012. ISBN: 9780132942782 (cit. on p. 43).

[8] John Brooke. *SUS: A quick and dirty usability scale*. 1996 (cit. on p. 73).

[9] Rachel Carmena. *So what is CSS, really?* Jan. 2019. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics (visited on 01/20/2019) (cit. on pp. 16, 17).

[10] Codecadamy. *What is REST?* Jan. 2019. URL: https://www.codecademy.com/articles/what-is-rest (visited on 01/22/2019) (cit. on p. 23).

[11] Docker. *Docker overview*. Jan. 2019. URL: https://docs.docker.com/engine/docker-overview/ (visited on 01/22/2019) (cit. on p. 25).

[12] Docker. *Overview of Docker Compose*. Jan. 2019. URL: https://docs.docker.com/compose/overview/ (visited on 01/22/2019) (cit. on p. 25).

[13] Lukas Eberhard et al. "Evaluating Narrative-Driven Movie Recommendations on Reddit." In: Marina del Ray, CA, USA: ACM, 2019. DOI: 10.1145/3301275.3302287. URL: https://doi.org/10.1145/3301275.3302287 (cit. on p. 50).

[14] eMarketer. *Number of smartphone users worldwide from 2014 to 2020 (in billions).* URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (visited on 02/02/2019) (cit. on p. 1).

[15] Apache Software Foundation. *An Introduction to JavaScript*. Jan. 2019. URL: https://cordova.apache.org/docs/en/latest/guide/overview/ (visited on 01/21/2019) (cit. on pp. 6, 19).

[16] Ionic Framework. *Framework Docs*. Jan. 2019. URL: https://beta.ionicframework.com/docs/ (visited on 01/21/2019) (cit. on p. 20).

[17] Ionic Framework. *Ionic Native*. Jan. 2019. URL: https://beta.ionicframework.com/docs/native (visited on 01/28/2019) (cit. on p. 52).

[18] Gartner. *Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018.* URL: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems (visited on 01/19/2019) (cit. on pp. 1, 5).

[19] Axel Haustant. *Welcome to Flask-RESTPlus's documentation!* Jan. 2019. URL: https://flask-restplus.readthedocs.io/en/stable/ (visited on 01/22/2019) (cit. on p. 24).

[20] B.P. Hogan. *HTML5 & CSS3 (Prags)*. Pragmatic programmers. O'Reilly Verlag, 2011. ISBN: 9783897213173 (cit. on p. 15).

[21] Ilya Kantor. *An Introduction to JavaScript*. Jan. 2019. URL: https://javascript.info/intro (visited on 01/21/2019) (cit. on p. 18).

[22] Anastasiia Lastovetska. *Native App Development vs. Hybrid and Web App Building*. Nov. 2018. URL: https://mlsdev.com/blog/167-native-app-development (visited on 01/20/2019) (cit. on pp. 5, 8).

[23] R. Likert. *A Technique for the Measurement of Attitudes*. A Technique for the Measurement of Attitudes Nr. 136-165. 1932 (cit. on p. 73).

[24] IQ Magazine. *Value of the global entertainment and media market from 2011 to 2021 (in trillion U.S. dollars)*. URL: https://www.statista.com/statistics/237749/value-of-the-global-entertainment-and-media-market/ (visited on 02/03/2019) (cit. on p. 1).

[25] Ivano Malavolta. "Beyond Native Apps: Web Technologies to the Rescue! (Keynote)." In: *Proceedings of the 1st International Workshop on Mobile Development*. Mobile! 2016. Amsterdam, Netherlands: ACM, 2016, pp. 1–2. ISBN: 978-1-4503-4643-6. DOI: 10.1145/3001854.3001863. URL: http://doi.acm.org/10.1145/3001854.3001863 (cit. on p. 6).

[26] Charo Malik. *Angular Architecture: Overview and Concept*. Nov. 2018. URL: https://dev.to/charumalikcs/angular-architecture-overview-and-concept-n8f (visited on 01/21/2019) (cit. on p. 20).

[27] Chris David Mills. *So what is HTML?* Jan. 2019. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (visited on 01/20/2019) (cit. on p. 15).

[28] D.R. Monette, T.J. Sullivan, and C.R. DeJong. *Applied Social Research: A Tool for the Human Services*. Cengage Learning, 2010. ISBN: 9780840032058 (cit. on p. 73).

[29] J. Sauro and J.R. Lewis. *Quantifying the User Experience: Practical Statistics for User Research*. Elsevier Science, 2012. ISBN: 9780123849687 (cit. on pp. 73–75, 77).

[30] University of Southern California. *Web Technology and Trend: HTML5*. Jan. 2019. URL: http://www-scf.usc.edu/~chenemil/itp104/webtech.html (visited on 01/20/2019) (cit. on p. 15).

[31] Jared Spool. *Two Simple Post-Test Questions*. Mar. 2006. URL: https://archive.uie.com/brainsparks/2006/03/23/two-simple-post-test-questions/ (visited on 01/31/2019) (cit. on p. 76).

[32] Michael Stowe. *Undisturbed REST*. MuleSoft, 2015. ISBN: 9781329116566 (cit. on p. 23).

[33] Max Summers. *Everything You Need to Know About Mobile App Development Architecture*. Sept. 2018. URL: https://magora-systems.com/mobile-app-development-architecture/ (visited on 01/20/2019) (cit. on p. 7).

[34] Pallets team. *Welcome to Flask*. Jan. 2019. URL: http://flask.pocoo.org/docs/1.0/ (visited on 01/22/2019) (cit. on p. 24).

[35] Karl E Wiegers and Joy Beatty. *Software Requirements 3*. Redmond, WA, USA: Microsoft Press, 2013. ISBN: 0735679665, 9780735679665 (cit. on p. 27).

[36] Jia Zhang. *Python Celery & RabbitMQ Tutorial*. Apr. 2016. URL: https://tests4geeks.com/python-celery-rabbitmq-tutorial/ (visited on 01/22/2019) (cit. on pp. 24, 25).