ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Faculty of Engineering, Computer Science and Psychology
Institute of Databases and Information Systems

Concept and Implementation of a Mobile Interactive Floor Plan

Bachelor thesis at Ulm University

Submitted by:
Raphael Schneider
raphael.schneider@uni-ulm.de

Reviewers:
Prof. Dr. Manfred Reichert

Supervisor:
Marc Schickler

2015/16

Version May 14, 2016

Satz: PDF-LaTeX 2$_\varepsilon$

Abstract

The relocation of a company facility always comes with great strategic and organizational challenges. The new site needs to be found, employees need to relocate to the new location, the workplaces need to be assigned to the employees, conference rooms need to be added to the conference room reservation system and the staff needs to be introduced to the new building. This process can cause high financial costs as well as a reduced employee performance during the time of acclimatization.

To minimize the additional stress created through these changes, a lot of effort needs to be put in informing the employees about news concerning the relocation. Guided tours through the building need to be organized, regular appearing newsletters need to be created and events to integrate new hires into the company culture need to be held. New technologies and the wide coverage of mobile devices provide a great opportunity to support the employees and can offer a modern and intuitive assistance both during the time of adjustment and every day work.

This thesis covers the development of an interactive floor plan, designed to help the employees find a certain person or conference room in the building. It discusses the challenges during the design and development and gives an overview of the future development goals of the project.

Acknowledgment

Before starting I would like to thank my supervisors, Marc Schickler of the Faculty for Datbases and Information Systems at the University of Ulm and Kaushik Datta of Mercedes Benz USA IT for the advise and support during my thesis. Both offered me their feedback whenever needed, while still allowing me to make this thesis my own.

Furthermore I would also like to thank Hans Moertl, without whom my 6 month internship at Mercedes Benz USA, just as the project which made up this thesis would not have been possible.

Finally, I must express my gratefulness to my parents and my girlfriend for the encouragement throughout my time abroad and the continuous support throughout my studies. Thank you.

Raphael Schneider

Contents

1

Introduction

Since the release of the iPhone in 2007 [1] as the first commercially successful smartphone, the number of personally owned touch devices has increased significantly. According to the *PewResearchCenter* nearly two-thirds of people in America own a smartphone [2]. In Germany more than 50% of the population owns a smartphone [3]. By today the average adult in the US spends more time working with mobile devices, than using a desktop computer, as shown in a study of the KPCB (see Figure 1.1). This circumstance has made the smartphone become a daily used tool and a platform for many new applications [4]. The additional improvment of celular data availablility over time, reaching up to 300 MB/s [5] with the latest standard of LTE, has led to a number of location based applications and services, such as *Google Maps* and *Uber*, which use the devices GPS module in combination with the data connection to display information optimized to the user's location [6][7].

However in the office environment similar applications are only rarely found. Apparently the idea of location based applications, mapping the surroundings of the user to the user interface (UI), has not fully reached this application area. Even though companies such as UPS, Wells Fargo, and Cerner Corporation [8] have taken up the concept to build applications supporting the business process, facility management and navigation tools can only rarely and with high resulting expenses be found [9].

Since in today's world of technology and the corporate world, the adaptiveness to frequent changes is an essential quality companies need to be prepared for. Employees are required to deal with organizational changes or even relocation of the facility. This

request for flexibility causes stress on the employees, lowering their productivity. To ensure the competitiveness of the organisation, this flexibility needs to be supported by convenient tools, which help the employees deal with the dynamic environment. The separate challenges opposed to the employees need to be analysed in order to find working concepts of support. Where can mobile applications help minimize the mental effort to adapt to the changes?

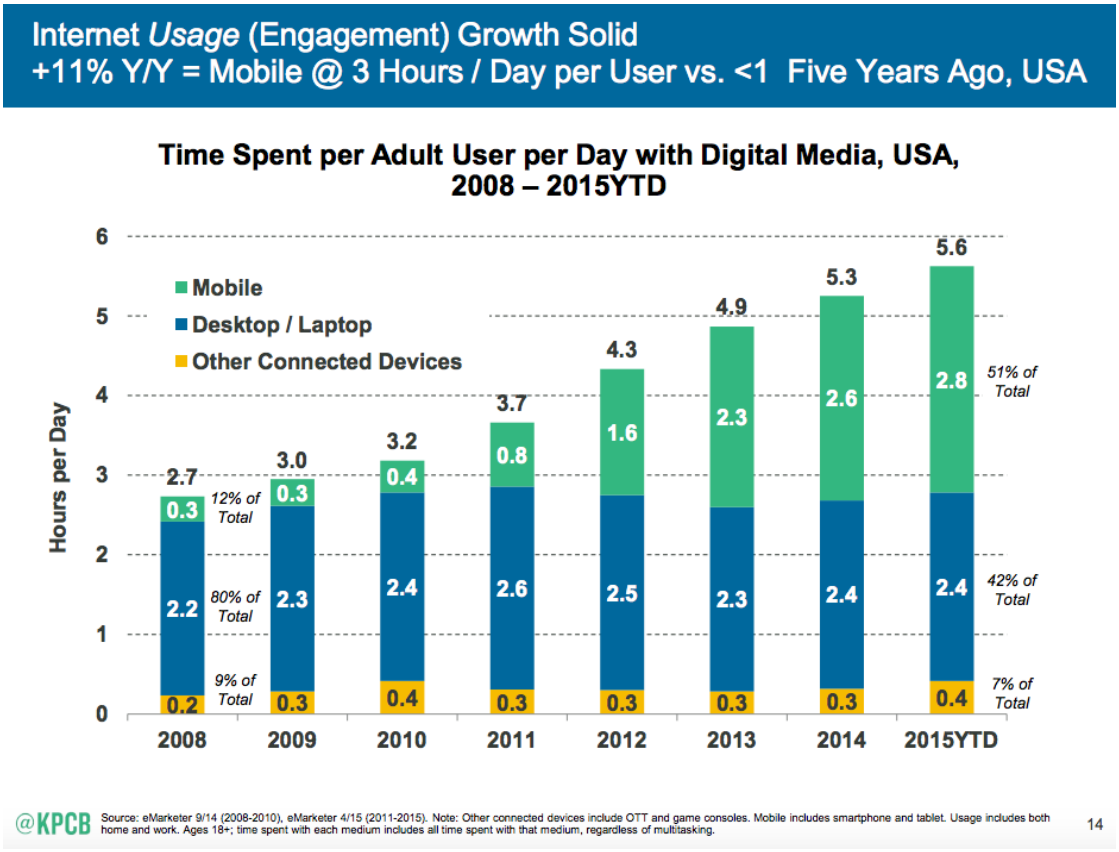

Figure 1.1: Internet Usage (Engagement) Growth Solid [10]

1.1 Purpose of the Thesis

Given the problem statement a concept for supporting users with changes in the floor layout or even the introduction of a new building in the course of a relocation should be discussed. Inspired by location based applications a mobile application will be implemented to prove the value of the developed concept. As a desired outcome the application should help navigating through the facility building and discover the allocation of conference rooms in the building.

The thesis should show the advantages and challenges given by the application area and result in a web based solution available to the targeted user group. Because of the high acceptance and availability of iPhones in the business environment, the application support will focus on Apple devices and the Internet Explorer 11. However the finished application should offer basic support for other platforms as well.

1.2 Scope

Since the project will be maintained as a long-term project, this thesis will cover the concepts and implementation of the project's first six months of development. To enable the continuous development on the project the agile process model SCRUM has been chosen. The project management tool Pivotal Tracker by Pivotal Software is used throughout the project. This section is supposed to give an overview on what the project's scope was planned to be during this first six month iteration.

The project will result in a responsive web application offering both a mobile and a desktop experience to the user. There will be separate views for the administrator to update seating and room naming and for the user to navigate through the floor plan. It will enable the user to search for an employee or conference room by free text search or by scrolling through an alphabetical list of employees. To ensure the value of this project in future, the focus will be set on the quality and flexibility of the software. The application

will offer adaptability to other facilities to incorporate the software in the company network in a future iteration.

The application will not support a locationing and navigation service through the building, because of the additional hardware and installation costs of indoor location services. Once the costs for Beacons needed to implement indoor locationing reach a reasonable cost, this feature can be added to future interations.

1.3 Structure of the Document

In chapter 2 of this thesis essential topics needed for later proceedings of this thesis are introduced. Beginning in chapter 3 the process of analysing the problem statement is covered. Starting with a thorough analysis of the given problem statement and an evaluation of concepts the document discusses the requirements for the application. In chapter 3.10 the functional and non-functional requirements are defined. Chapter 4 presents the interaction scenarios and presents the designed solutions for these, before the technical perspective of the implementation is covered in chapter 5. Furthermore in chapter 6 the requirements evaluation, just as a collection of challenges during the project, is stated. Lastly a summary can be found in chapter 7. This chapter recaps the challenges, gives an outlook over the future development of the application and states a personal experience summary.

1.4 Context of this Thesis

Being developed during a six month internship in the information technology department of *Mercedes Benz USA* (MBUSA), this thesis was developed in a corporate context, making it possible to get immediate feedback by the end users.
Due to the ongoing move of the MBUSA headquarters from Montvale, New Jersey to Atlanta, Georgia the employees coming from Montvale, as well as new employees need to find their assigned cubical or office inside the new building. To assist this process an external company was assigned to implement a solution. The resulting solution was a

desktop web interface based on PHP which offered the employees to look up the location of employees and conference rooms. The performance and maintainability however did not fit the needs of the situation offering a desktop UI experience only and therefore need a redesign. The new concept implemented for this thesis offers a fast, Java-based solution optimized for a mobile, tablet and desktop user experience. It enables users to navigate through the building without being on their desktop computer. The targeted platforms for the application covers the current versions of the commonly used mobile platforms (iOS, Android and Windows Phone 8) and the Internet Explorer 11 for desktop users. The software offers enough flexibility to adapt to other MBUSA sites.

2

Fundamentals

This section aims to give a brief overview of the technologies used in the project and why these were beneficial to the project. These topics are essential for the further understanding of this thesis. It is not intended to be an exhaustive documentation of the subjects.

2.1 MVVM - Design Patter

The *Model-View-ViewModel* (MVVM) pattern is a variation of the classical *Model-View-Controller* (MVC) pattern. It was introduced by John Gossman in October 2005 on the Microsoft developer blog [11]. Targeted at modern UI applications it offers advantages in reducing the effort of indepentent business logic and user interface development.

The Model-View-ViewModel pattern defines 3 components:

- The Model contains the business logic
- The View constructs the user interface
- The ViewModel initializes the Model and offers an interface to the View to bind to the model. It acts as a transtator between Model and View.

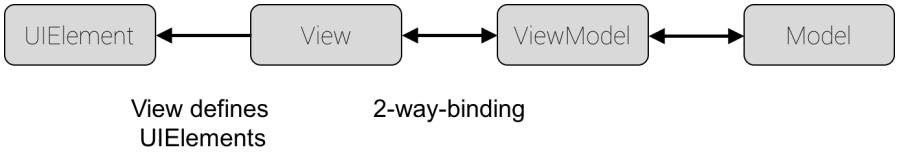

Figure 2.1: Model-View-ViewModel pattern structure [11]

2.2 REST Services

The REST paradigm (*Representational State Transfer*) defines a set of constraints to implement the communication between components in distributes systems. Is has first been introduced as an alternative solution to SOAP and WSDL services by Roy Fielding in the year 2000 [12]. In his disertation Roy Fielding defines a set of five constraints, which need to be met by REST services. The level of implementation however can be adapted to the project's scope. The constraints include the *client-server architectural design*, the *statelessness of apis*, *client-side caching*, *uniform interfaces* and the *layered implementation* of the architecture. Due to the uniform interfaces REST services are ideal for large data transfers between the client and the server.

The paradigm has received a high acceptance especially in web development. Most implementations use the *Uniform Resource Identifiers* (URI) to address resources offered by the service. In the context of web development requests are created using the *Hypertext Transfer Protocol's* (HTTP). The HTTP methods *GET, DELETE, POST and PUT* define the action performed on the service, while the URI defines the resource the methods are operated on. In listing 2.1 a sample request can be seen.

```
1 HTTP/1.1 GET /v1.0/users/
2 Host: api.floorplan.com
```

Listing 2.1: Sample GET-Request

2.3 The Java Script Object Notation Format

The *Java Script Object Notation* is a compact data format, which has been defined by Douglas Crockford in *RFC 7159* [13]. It offers a data representation, readable for both humans and machines.

```
<document> ::= <object> | <list> | <value>
<object>   ::= {} | {<atrlist>}
<attrlist> ::= <attr> | <attr>, <attrlist>
<attr>     ::= "<string>" : <document>
<value>    ::= <number> | "<string>" | <boolean> | null
<boolean>  ::= true | false
<string>   ::=  <char> | <char><string>
<char>     ::= A | B | C | D | ... | Z | a | b | ... | z
<number>   ::= <digit> | <digit><number>
<digit>    ::= 0|1|2|3|4|5|6|7|8|9
```

Listing 2.2: BNF of the JSON Syntax

2.4 AngularJS

The AngularJS framework [14] is a structural JavaScript framework based on the MVVM design pattern 2.1. It enables the development of dynamic single-page web applications with the usage of data binding. The developer can use HTML as a templating language to create visual components, which can be bound to a controller.

Angular implements 3 basic concepts for the implementation of web applictaions:

- Controllers: used to configure the scope, which the view can bind to and connects the UI to the application logic
- Services: define the application logic. Each service is a singleton instance and therefore enables resource sharing between controllers
- Directives: define view specific behaviour and components

Aditionally to these main components Angular defines *filters*, which define logic to translate bound data between the view and the model.

3

Requirements Analysis

This chapter defines possible features and evaluates their priority. As a result the functional and non-functional requirements can be found in chapter 3.10. Chapter 6 covers evaluation of the implemented features.

3.1 Different User Groups

Throughout the usage of the application, users will go through different stages of knowledge about the building and the application resulting in different user groups. These need to be offered different features in order to ensure the long term usage of the floor plan mapper. Initially the users will be newcomers. Once these people get to know the building they are acclimated users. Additionally, guest users can be possible users.

Newcomers

During the relocation of employees the application's main use will be to assist people navigating through the building. These users, whose objective in using the application is to learn about the buildings structure, will need a fast and easy way to look up the position of a conference room or employee.

Acclimated Users

After having used the floor plan mapper for a period of time, users will know about the regularly used locations in the building and reduce their need for the navigation feature

of the floor plan mapper. These users need to be considered to ensure the application's long term usage. Since the application offers an intuitive view on the floor plan many features of the daily workflow, such as conference room reservation, can be integrated into the application to target these users. Additionally, the search of employees can be expanded by offering different search criteria, such as department, abilities and area of responsibility to quickly find the right person of contact. To improve the search for conference rooms a detailed search, enabling to enter further details, such as capacity and equipment, is beneficial.

Guest Users

In future it can be made possible to send an invitation to a customer or business partner visiting the facility, to enable the guest only to see the relevant locations in the building filtering out all unnecessary distractions from the digital floor plan.

3.2 Digital Floor Plan

As of today new employees get a printed floor plan in the lobby of the facility showing them the organization of the building and the location of their cubical or office. This map helps the newcomers to explore the building. However, due to this media format it often is not possible to have these plans at hand the whole time. Usually these documents are kept at the working desk and lost within a short period of time. A mobile application offering digital versions could simplify this process for the employees, reduce the use of paper floor plans and keep the current floor plan offered to the employees easily [15][16].

Concept - Digital Floor Plan

Initially the floor plan can be deployed on a simple web server and maintained manually by an administrator. To offer an intuitive interface to the mobile application the data can be made available in SVG and JSON formats. In a later iteration the server can

access an external database to integrate the application in the company environment and minimize the administrational effort. On the front end a mobile application should be made available to the employees, offering a simple and intuitive user experience. To make the application easily available to all employees it can be deployed as a web application on an internal HTTP server. Since there is currently no need to access hardware features on the mobile device, a native application will not be needed. The key feature of the mobile application is the displaying of these plans and incorporating the location of employees and conference rooms. Therefore the application should display the digital version of the distributed current floor plan with further possibilities of interaction. The digital distribution of the plans ensures the latest information for the users.

Evaluation - Digital Floor Plan

From a financial point of view a digital distribution of the floor plan enables fast changes to the floor layout and chair assignement without the need to reprint the documents. The floor plan can be edited in the facility management software and distributed directly from the source eliminating unnecessary intermediate steps. On the user side a current version of the floor plan is available to access at all times. By accessing the floor plan mapper through a private or corporate device, the floor plan is always quick at hand.
The downside of the digital implementation of a floor plan mapper is the effort which needs to be taken to incorporate the application in the existing environment to access external data instead of creating a new individual database for the application. Another issue occurs, when choosing the server to deploy the application. Since most employees do not have a corporate smartphone, the deployment on the intranet would only enable employees with corporate phones to access the page. However, deploying the application on the internet would lead to the publication of potentially confidential data on the internet.

3.3 Support for Device Specific Gestures

Implementing the application as a web based application enables users to access the utility from multiple devices and platforms. To support users from all devices a responsive design is required. Users from desktop devices will rely on the usage of mouse and keyboard, while mobile users will expect to interact using touch gestures.

Concept - Support for Device Specific Gestures

Since the point and click paradigm translates easily to the touch paradigm, simple buttons to open the menu and select the search bar have no need for device specific optimization. The navigation through the map needs separate navigation mechanisms though. Desktop users should be able to move the map using click and drag. Additionally, the ability to zoom in and out can be made available through the mouse wheel. For two-key-mouse users a separate on screen control or the possibility to use the + and - keys to zoom in and out should be implemented. In case the users prefer not to use the mouse for browsing the map, the arrow keys can be used to move the map on screen. Mobile device users will expect a pinch and drag support just as a support of swipe gestures to open and close the offered menus.

Evaluation - Support for Device Specific Gestures

The usage of device specific interaction patterns is a crucial requirement for maximum user acceptance. Therefore the advantage lies in minimizing the cognitive load for the user. However the integration of these features can lead to technical challenges. Since the application is to be implemented as a single web application, this demands for a responsive UI design.

3.4 Optimized Employee and Conference Room Search

The key objective of the floor plan mapper is to find the cubical or office of a specified employee or the location of a specific conference room. Employees and especially new hires often need to meet people and conference rooms whose exact location they do not know. This causes a lot of time being invested in searching the correct way to the destination, therefore reducing the time of productivity.

Concept - Optimized Employee and Conference Room Search

To minimize the time to find the location data of the cubical, office or conference room and the route to the target location, the mobile application should offer a free-text search to the user. This search enables the user to enter the name of an employee or conference room and select the wanted entity from the result list. On Selection the target should be shown on the floor plan giving the user basic information about the location and person or conference room.

Evaluation - Optimized Employee and Conference Room Search

It is clear that the majority of new users will preferably use this feature to find the wanted locations in the building. However a simple implementation of the search may lead to difficulties on the user side. It is conceivable that a user knows the name of the person or conference room he is looking for, but does not know the correct spelling.

3.5 List of Employees and Conference Rooms

In addition to the free-text search, a list of all employees and conference rooms used to be implemented in the former implementation of the floor plan mapper. This list offered an alphabetically ordered list enabling the user to browse through. Showing a total of close to 700 employee names, this feature's usage stayed at a minimum and therefore needs optimization.

Concept - List of Employees and Conference Rooms

To make browsing the employee and conference room list as quick and easy as possible, the list can be separated in multiple sublists containing only entries with specified initial letter. As a proposal the sublists A-G, H-P, Q-Z should be implemented. The intervals of the sublists can be dynamically adjusted according to the set of employees in the database.

Evaluation - List of Employees and Conference Rooms

The separation of the employees in three sublists has a definite advantage over the single list. However, for smaller data sets, such as the list of conference rooms, an alphabetical grouping will cause unnecessary user interactions. Optimizing the range of these sublists to even out the amount of employees in each sublist will probably cause immense performance issues, since the entire set needs to be evaluated in order to find the ideal points of separation.

3.6 InDoor Tracking

When using a simple map only showing the target location, many users could have issues finding their own location to find the path to the wanted location. This is especially applicable to first time visitors of the building.

Concept - InDoor Tracking

Since the publication of the originally for military usage designed Global Positioning System (GPS) many outdoor navigation applications have been developed not only enabling to show maps of the environment, but also integrating the user's position in the user interface. This feature had not been possible for in door usage until the introduction of Bluetooth Low Energy Beacons in 2014 [17]. These Beacons enable the exact locationing of a device in the room.

With this new technical development the application can offer a precise indicator of the user's current location in the building and even suggest an ideal route to the target location.

Evaluation - InDoor Tracking

The obvious advantage of the implementation of an indoor navigation is the decrease of extraneous and intrinsic load (see Definition 1). The additional information helps finding the correct floor and leads to an enhanced assistance for new users. However, the implementation of inDoor Tracking asks for additional hardware which, at the time of this thesis, is still a costly investment. Each room needs to be equipped with a least four beacons. The battery life of a beacon is specified to be between two to three years [18]. This runtime proves the low power consumption of the devices. Non the less the number of beacons needed for a building leads to a high effort in maintenance.

Definition 1. *"The load is called 'intrinsic' if it is imposed by the number of information elements and their interactivity. If it is imposed by the manner in which the information is presented to learners [...], it is called 'extraneous' [...]." [19]*

3.7 Rotation of Map according to Device Orientation

Apart from the position in the building, users often have difficulties finding the right orientation of the map to fit the building. This leads the users to make wrong assumptions of the direction to their target location. Especially business buildings with close to symmetric floor layout and few points of differentiation cause these problems.

Concept - Rotation of Map according to Device Orientation

To help users find the correct orientation of the map, HTML5 offers the Device Orientation API for supporting mobile devices. This enables the web application to access the device's compass which can be used to rotate the floor plan accordingly. To hide the inaccuracy of the compass sensor, the rotation can be limited to 45 degree steps.

Evaluation - Rotation of Map according to Device Orientation

The aim of this feature is to maximize the user experience and minimize time and effort to navigate to a specific target location in the building. However, the real-time rotation of the map can cause the user to lose track of disposition in the event of a rotation.

3.8 View and Book Conference Rooms

In the current workflow of booking a conference room, employees need to use the room list in outlook and select the wanted timeframe. This UI offers basic information about the room such as capacity and the support for video conferences. However, the majority of employees are not familiar with the workflow and therefore not able to book a conference room. As a consequence, the booking of conference rooms is often avoided, leading to delays in finding a location for business meetings.

Concept - View and Book Conference Rooms

As the floor plan mapper is to be designed to offer a quick way of finding a specific cubical, office or conference room, it has the potential to offer a more intuitive way of booking conference rooms. After having found a conference room, the user will be offered detailed information about the capacity and equipment of the room. From there the user is able to open the current schedule of the conference room and select the wanted time slot.

Evaluation - View and Book Conference Rooms

Since low room availability is a common complaint in day to day business, an intuitive application offering an alternative way of mapping and booking conference rooms would be highly appreciated by employees. However the interface to the Room Reservation System is not easily accessible and would need a longer development time.

3.9 Maintaining User Data and Seating

Since the data displayed by the floor plan mapper will be a dynamic set of data, a way to efficiently manage and maintain this data is essential to the project. To ensure the up-to-dateness of the data displayed in the floor plan mapper, a suitable administration pattern needs to be found.

Concept - Maintaining user data and seating

Generally there are two way of dealing with the challenge of data management, which are presented and evaluated in the following.

- Usage of an Administrational Tool: The classic approach of data management is the implementation of an administrational tool. This way, only a limited set of users, the administrators, are able to modify the data in the database offering the advantage of preventing unwanted changes.
- Usage of a Community-based Approach: Inspired by the modern social networks another possible implementation of data management is to integrate the users in the maintenance of the data. This enables all users to edit their own data just as the data of other employees. Depending on the privacy situation this approach requires an aditional login mechanism to identify data owners.

Evaluation - Maintaining user data and seating

On the one hand the limited set of privileged users in administrational data management, simplyfies the maintenance of data correctness. On the other hand this leads to an enormous amount of effort needed on the administrators side to keep the data up to date. With a smaller group of administrator users a user friendly UI is less important. However, an introduction of the users to the workflow is essential.

In comparison to this approach, the community based approach reduces the work load for the administrator by distributing the task of data management to the users, making the users responsible for updating their data. Of course in an open environment this approach would need extra authentication functionalities to ensure only the owner to edit his own data, but in the context of a company internal application, the simplicity of usage and flexibility overweights this aspect.

3.10 Requirements Definition

After the possible application features have been covered and the decisions on which features should be implemented have been stated, this section summarizes the functional and non-functional requirements which were selected for the scope of this thesis. In this thesis, the requirements are separated in two categories: functional and non-functional requirements. Other features mentioned in the application scenarios will be subject of further development. The degree of coverage of these requirements will be analyzed in the chapter 6.

3.10.1 Functional Requirements

The Functional Requirements (FR) describe, the features which are to be offered to the user by the mobile application.

FR00 View Floors

The floor plan should be displayed in the application as a scalable SVG allowing dragging and zooming without a loss of quality. The user should be able to switch between the available floors.

FR01 (Un-)occupied Seats

All chairs should be displayed on the floor plan with a colorcoding indicating whether the chair is occupied or vacant at the moment.

FR02 View Conference Rooms

Conference rooms should also be displayed in the floor plan.

FR03 View Entity Data

For each employee and conference room relevant information should be specifiable. This information contains attributes like full name, job title, phone and cell phone number, mail, department, location number, floor, capacity and support for video conferencing for rooms. These attributes can be extended to fit the available data source.

FR04 Free-text Search

In the main view the user should be able to locate a search box to search the wanted employee or conference room. The search box should offer immediate response by offering a list of possible results.

FR05 Employee & Room List

In addition to the free-text search, the application is supposed to offer a separate, sorted list for employees and for conference rooms. The list of employees should, due to the bigger number of entries, be split in alphabetical sublists.

FR06 Offline Mode

To enable the usage of the floor plan mapper without wifi connectivity and to minify loading times after the first usage of the application, the mobile web browser should be able to run the application in offline mode without any major disadvantages.

FR07 Indicator for Device Rotation

The application should incorporate a visual indicator of the device orientation in comparison to the building. Since this feature can lead to distractions the possibility to deactivate is required.

FR08 Administrational Tool

To enable an easy modification of the chair allocation and populating the floors with chairs, an administrational tool should be built. This tool should offer the possibility to add, move and remove a chair on a floor.

FR09 Community-based Seat Management

In the app users should have the possibility to claim a chair and thereby keep the seating information up to date at all times.

FR10 Community-based Data Management

Users should have a possibility to keep their contact information up to date. This feature should be implemented within the floor plan mapper without introducing a separate view to the user.

3.10.2 Non-Functional Requirements

Non-functional requirements (NR) describe the requirements from a meta perspective and therefore have no direct impact on the applictaion's functionality. They define the wanted characteristics of the application by setting quality specifications. Meeting these specifications is crucial for the overall user experience, application security and maintainability of the application.

NR01 High Compatibility

As the project focuses on the usage of the floor plan mapper on mobile devices the highest priority concerning support needs to be put on the most commonly used mobile platforms. Therefore the application should support current versions of the following mobile operating systems:

- Windows Phone 8
- iOS 9.3
- Android 6.0 Marshmallow

Concerning desktop browsers the focus should be set on the Internet Explorer 11.

NR02 Accessibility

The floor plan mapper should be accessible both on the intranet and the local guest internet connection to enable temporary guests to use the application on site, too.

NR03 Fluid UI and Performance

The reaction time to user interactions needs to be under one second on current mobile devices. The scaling and dragging of the floor plan should render smoothly.

NR04 Data Safety

Precautions against data loss on the server need to be taken. A regularly executed backup of the database is required.

NR05 Data Security

The data saved on the server should be protected against unauthorized access and modification.

NR06 Usage of the MVVM Pattern

To ensure the maintainability of the application and reduce the amount of code written for data synchronization, the MVVM design pattern (see section 2.1) should be used for the client application.

NR07 Usage of Dependency Injection

To enable a quick modification of the server code, the development of the Java Spring server application should be implemented using dependency injection. The loose coupling of Java Spring components also helps the reuse of code for future projects.

NR08 Intuitive User Interface

The design of an easy to learn user interface should eliminate the need of a user documentation and enable users to access features immediately.

NR09 Device Specific Gestures

The application should offer adequate possibilities of navigation for each device. On mobile devices the user should be able to browse through the map using commonly used gestures. Desktop users need be able to zoom in and out with the scroll wheel of the mouse and move the map by dragging the mouse. Additionally, a navigation using the arrow keys on the keyboard or a separate on screen navigation should be offered for users of two-key-mice.

NR10 Corporate Design

The application should match the design guide lines of the current Mercedes Benz corporate identity.

NR11 Minimize Data Traffic

Since the application should mainly be used on mobile devices with limited data volumen and network speed, the network impact should be kept to a minimum by caching data where possible.

4

Application Scenarios

To gain a detailed overview of the features offered by both the user and administration components as well as the relating workflows, this chapter contains the application scenarios of each components. The Application Scenarios are backed with the original wireframes created for the application to offer a more detailed insight into the set of possible features and the challenges of the UI design.

4.1 The Administration Application

The administration of the data can be managed through a separate web interface (see Figure 4.1), which is optimized for desktop usage only. The application offers a view of the floor plan with the available chairs and conference rooms. In the floor plan mapper each chair and conference room can be selected with a click. After having selected a chair or conference room the administrator can reposition the object using the arrow keys on the keyboard. Once the object has been moved to its final position, the administrator can save the changes by selecting the save button in the speed dial control on the bottom right of the user interface. Alternatively the object can be deleted from the floor plan. The speed dial control offers options to add a chair or conference room to the floor plan, if no object is selected. The different states of the speed dial control can be seen in Figure 4.2.

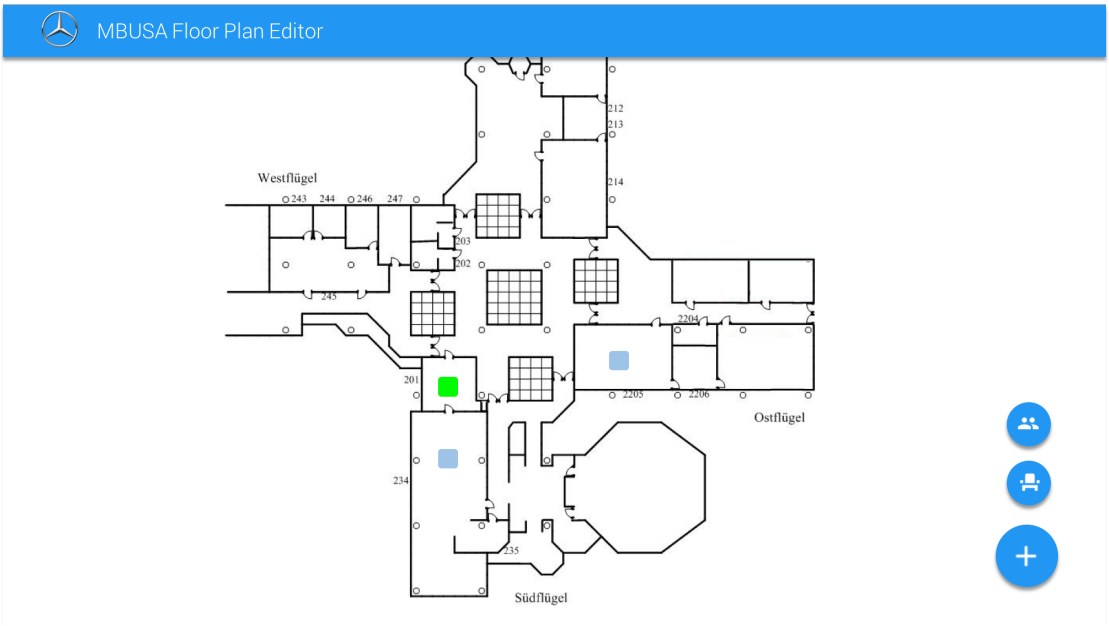

Figure 4.1: General Administrarion UI

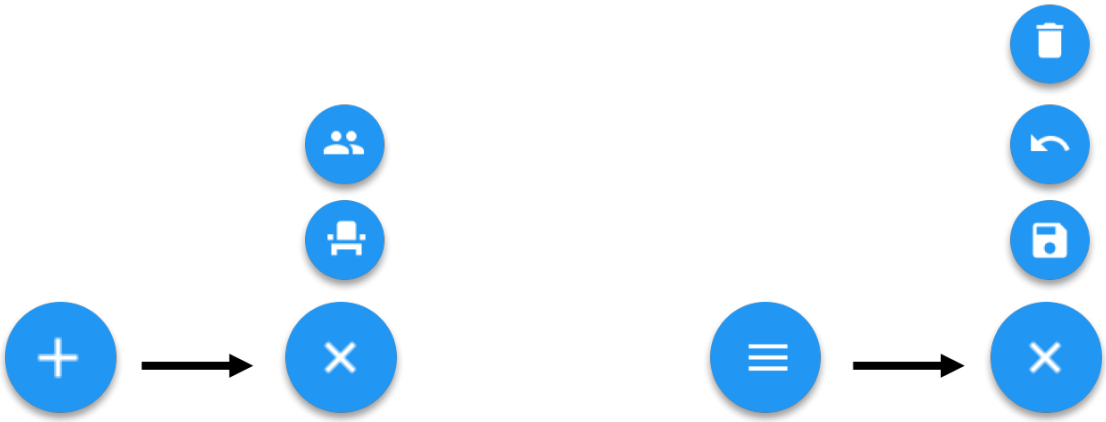

Figure 4.2: Speed Dial Control Add and Edit Modes for Administration UI

4.2 The Mobile Application

In Figure 4.3 the initial UI wireframe of the application which illustrates the viewing application scenarios is shown. These features have been in the scope of the first development iteration. In the following, the interaction workflow with the UI will be described in detail.

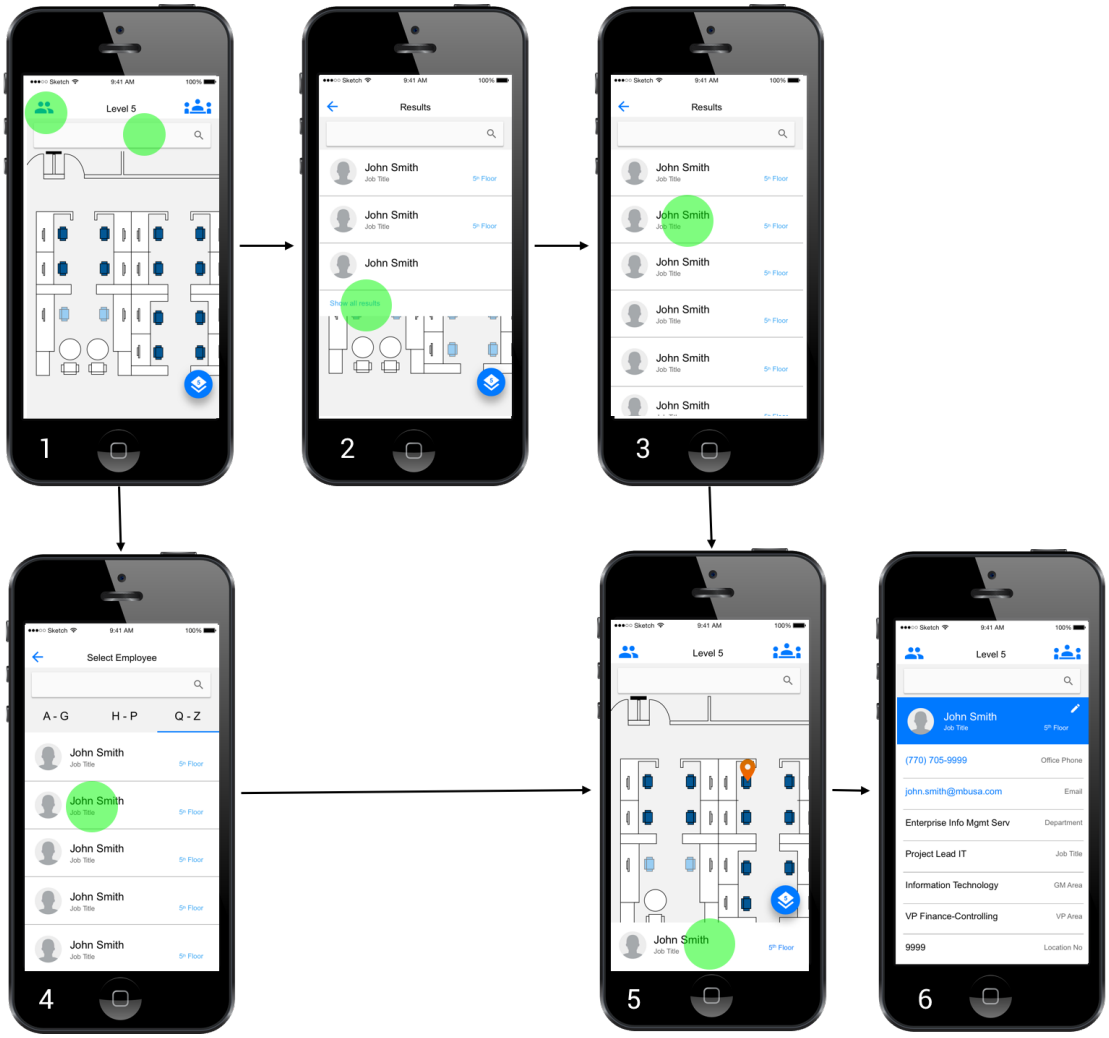

Figure 4.3: Browsing through employees and display employee data

4.2.1 Search in Map

In the top left UI state shown in Figure 4.3 the user is shown the map of the first floor or the last viewed floor. The user can drag the map to the desired position and use a pinch gesture to zoom in and out. In the bottom right corner the user can find a circular speed dial control which informs the user of the currently viewed floor. Touching this control opens a selection of available floors from which the user can select one by touching the entry on the screen.

In the map the user can find conference rooms displayed as blue rectangles. Chairs are displayed as chair symbols and are color coded. A light blue represents an unoccupied chair while dark blue indicates an occupied chair. Occupied chairs and conference rooms can be selected directly from the map by touching to display the information of the occupying user or conference room.

4.2.2 Search by Name

The search field at the top of the user interface offers a free text search to find employees and conference rooms. When typing in the search query, the application immediately offers a list of the three best matching results. The user can expand the list to view all results by tapping on „Show all Results“ at the bottom of the result list. Tapping an entry in the results list selects the chair of the employee.

4.2.3 Search in Alphabetical List

Alternatively to this search, menu icons are available at the top left and right of the user interface. To find a conference room, the user can use the room icon in the top right. This will open a side menu listing all available conference rooms. The list of employees can be opened by touching the user icon in the top left corner of the user interface. This will open a side menu with the alphabetical list of employees grouped in three sublists. The user can select an entity from either of these lists by tapping on the entry.

If a chair or conference room has been selected, the application shows the position of the entity on the map and displays the general details at the bottom of the interface. Touching this details panel expands it and makes all available details visible.

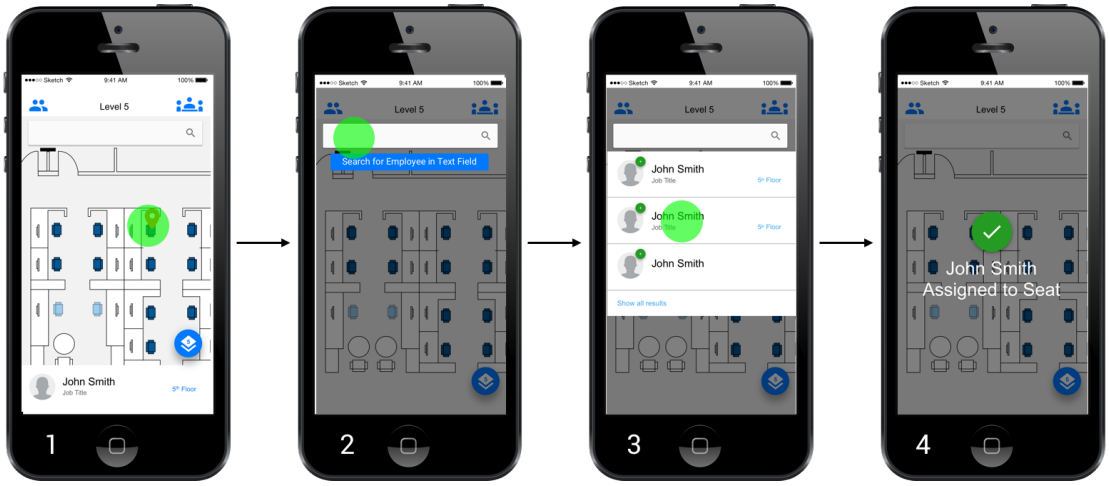

Figure 4.4: Steps how to assign a chair to a user

4.2.4 Claim a Chair

Figure 4.4 shows the application scenario of a user claiming a new chair. Since the data of the application should be managed by the users themselves, this feature offers an easy and fast way to enter a relocation in the app. The user can navigate to his new chair using the pinch and drag features, just as the floor speed dial at the bottom right of the user interface to change to another floor. Once the user has selected an unoccupied chair, a button appears beneath the chair. The user can click onto the button to change the application into a modification mode. A toast beneath the search bar informs the user to use this input to find the user who is to be assigned to the chair. In the result list the user can select the profile of the person to be relocated. The application confirms the change with a confirmation screen as seen in Figure 4.5.

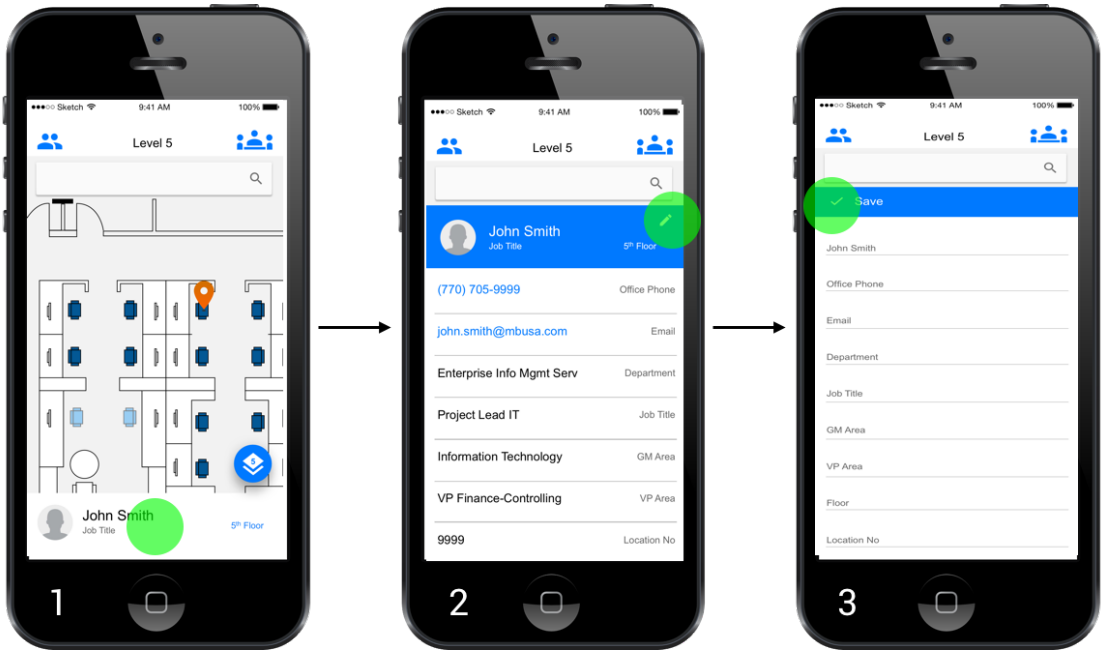

Figure 4.5: Steps how to open user data and execute changes

4.2.5 Edit User Data

Besides claiming a chair, a user needs to be able to update his user data in the floor plan mapper. This process is directly integrated in the detail view of the application, as can be seen in Figure 4.5. Initially the user needs to open the employee profile, which needs to be edited, as described before. After opening the details view of the profile, an edit icon can be found at the top right of the panel. This takes the user to the edit view, which offers a text field for each editable property. Once all changes have been made the save button at the top of the panel submits the changes to the server and updates the local database.

5

Architecture and Implementation

In this chapter the overall architecture, underlying data structure, technologies used and the data exchange mechanisim between the client and server application, are presented. The chapter starts out with characteristics common to the client and server component. In section 5.2 and 5.3, a more specific view on the two tiers making up the floor plan mapper can be found.

The system is composed of a mobile web application and a server component which acts as the interface to the database. Additionally, a simple web interface for general administration is offered. Since a major objective of the project was to minimize the network traffic needed for the application, the major part of this chapter covers the mobile application. The server application is kept simple to enable later integration of other existing databases as data sources.

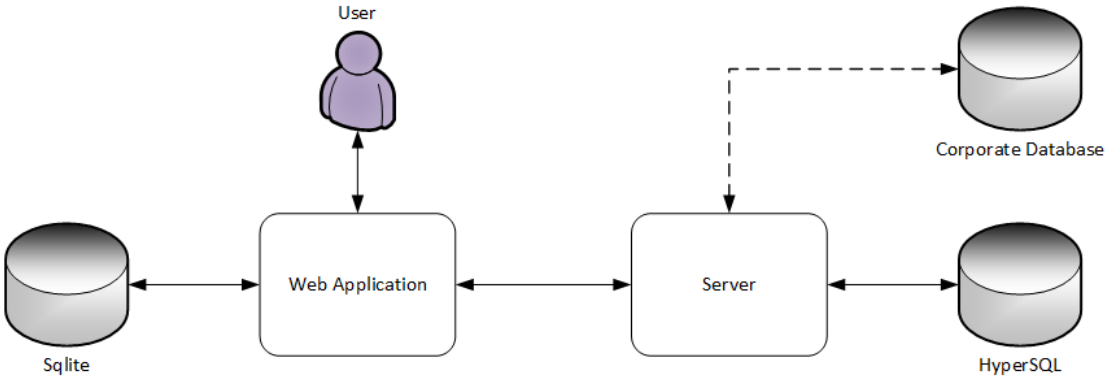

Figure 5.1: General Administrarion UI

5.1 The Data Model

As of the scope of this thesis, the server application will access a dedicated HyperSQL database to serve and update the data. In a later iteration of the project this data will be extended by other corporate databases and thereby integrated into the centralized data management of employee data. This will reduce the administrational effort furthermore and improve data reliability.

After evaluating the future support for the WebSQL API and IndexedDB API on mobile devices, it has been decided to make use of the IndexedDB API on the client side. This enables the application to store the data needed for the floor plan mapper which makes it possible to offer a caching mechanism and enables the viewing feature of the floor plan mapper in the offline mode (see section 5.4.2). As a result, the required server communication is reduced to the initial installation, data updates and data changes.

To fit the needs of the application, the design of the underlying data structure is crucial for the development of an efficient and maintainable mobile application. Due to the continuously improving performance of mobile devices and rising number of users, a number of non-confidential tasks can be run on the end devices in modern applications. This however makes the design of an application-optimized database even more important. The limited processing power and battery life, just as the simplistic design of the IndexedDB API on the client, requires a database design to fit the presentation on the device. Since the datamodel of the server and the mobile application for this thesis match each other, the structure of the database can be unified in one diagram (see Figure 5.2). Implementational differences of the database between the client and the server will only occure because of the nature of the used technologies.

In Figure 5.2 the used data model for the floor plan mapper is shown. The tables *'Room'* and *'Chair'* represent positionable objects which specify a location in the map with the attributes x, y and floor. The chair has an additional rotation property to ensure a correct orientation of the chairs in the user interface. A chair can reference a user

who *occuppies* it in order to link a location to the user. This enables the datamodel to display both occupied and unoccupied chairs. The separation of the user from his current location in the building forces the client to perform a join of the two tables chair and user, but does not cause a major impact on performance because the representation fits the initial view. The need to join the tables is limited to user interactions selecting a chair or a user. All listed tables have an automatically incrementing integer 'id' as primary key. It is crucial that this primary key is final and cannot be reused for other objects, since the synchronisation and modification process relies on these attributes. At that point it is also important to know that the client application can react vulnerable to duplicate identities depending on the implementation of the database service of the used browser.

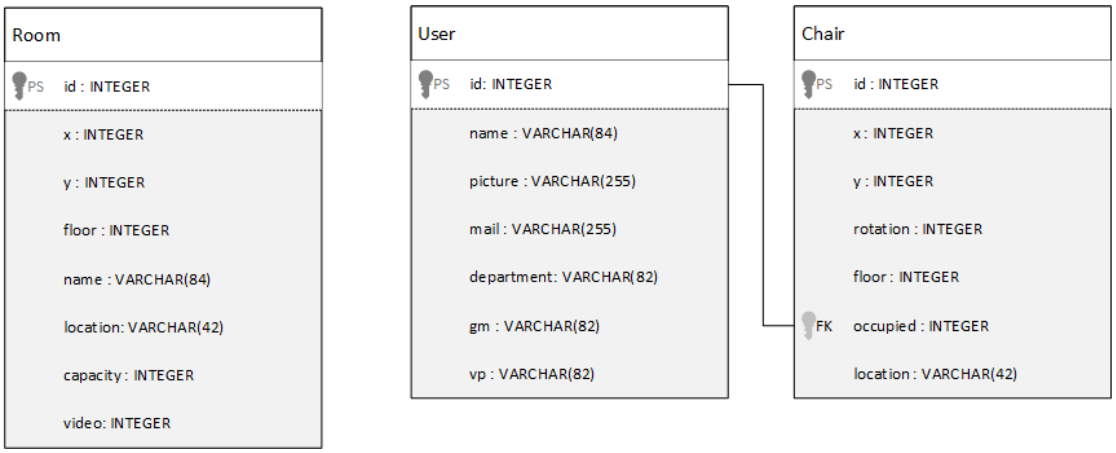

Figure 5.2: Datamodel with Relations

5.2 The Server Architecture

In the following, the programming languages which have been evaluated to fit the requirements of the server implementation best will be introduced. The section is intended to give an overview over the reasons for these decisions, just as the architecture of the implemented solution.

In the obsolete implementation PHP was used for the server processing of the application. Since PHP is a technology generally not present in the MBUSA environment, the server

technology was chosen to be Java Spring Boot. This technology enables a loose coupling of components and therefore improves the reusability and maintainability of the source code.

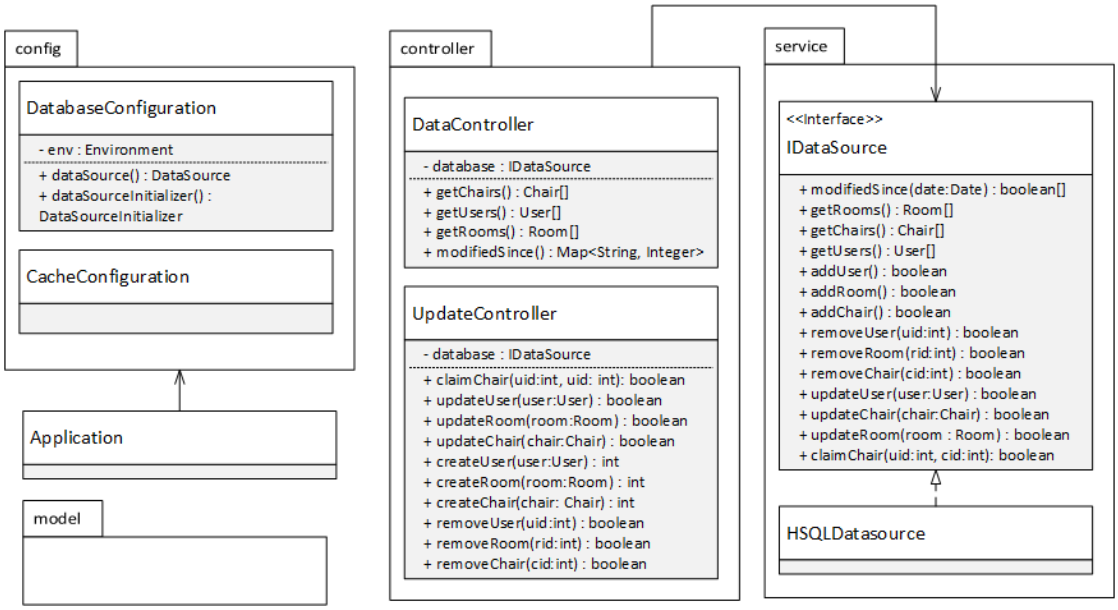

Figure 5.3: Class Structure of Server Implementation in UML 2.0

Figure 5.3 shows the implementation of the server. In the *model package* the beans used for the communication between the *controller* and *service package* can be found. These classes contain the attributes as defined in the datamodel in Figure 5.2 and offer both getters and setter. The *'User'* object is an exception having an additional property *'location'*, which refers to the foreign key *'Chair.location'*. This attribute is used for the user edit feature to execute changes on both the user and the location number of the current chair.

The *controller package* contains the definition of the REST interface using Java Annotations. Using the *Programming to Interfaces Paradigm* [20] these REST controllers are auto-wired to the matching implementation of the inteface *IDataSource* in the *service package* at server startup. This enables to interchange the used implementation to connect to the database at configuration, instead of making changes in the code itself. To access the HyperSQL database the Class *HSQLDataSource* implements the *IDataSource* interface.

The *config package* contains the configuration files which are loaded by the *Application* class defined in the root package. The *CacheConfiguration* initializes the Cacheable feature offered by Spring Boot. This helps minimizing the stress on the application server. The *DatabaseConfiguration* uses the database properties in the configuration file to initialize the connection to the database. Using an external configuration file makes it possible to change the used database without the need to rebuild the entire project. For security, the access data to the database can be encrypted on the file system.

5.3 The Mobile Application

Due to the simple server implementation the mobile application contains the complex logic of the project. The system component represents the tool implemented for the users to discover and maintain the data. In the following sections the general architectur and user interface design are presented.

5.3.1 The Architecture

On the client side it was decided to use the web framework AngularJS [14]. The framework is based on the MVVM design pattern (see section 2.1), which helps with the separation of concerns. For each task a service is implemented which handles the data modification and structures the code by its functional components. The synchronisation between the data model and the data representation on the user interface is handled by AngularJS through data binding. To implement this data binding, AngularJS uses a dirty checking algorithm (see Definition 2).

Definition 2. *Dirty checking is a mechanism, which is used to detect changes by scanning through a list of watched variables. [21]*

The structure of the implemented floor plan mapper can be seen in Figure 5.4. Because of AngularJS's way of structuring different components in services, directives and

controllers, a regular UML 2.0 class diagram has been used to model the applications' components. In the diagram each entity represents a controller, service, filter or directive (see section 2.4).

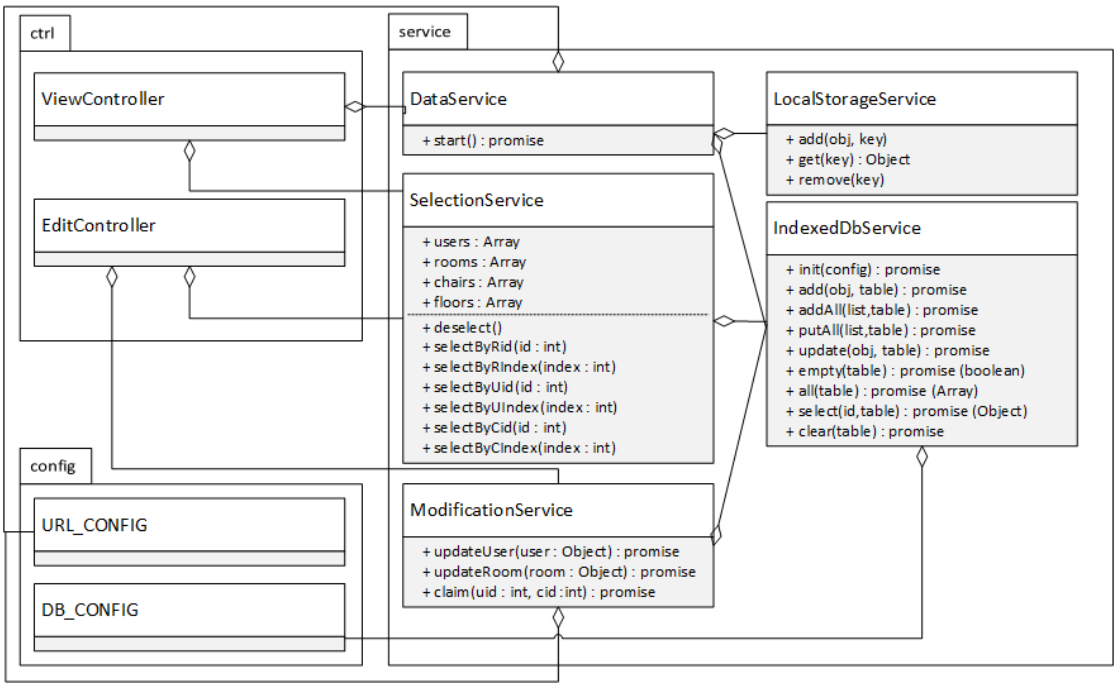

Figure 5.4: Class Structure of Floor Plan Mapper App Implementation

As the impact of AngularJS filters on the application only affects the UI representation, the filters have been left out in the diagram. However, the need for these filters was not negligible. For the scope of this thesis the following three filters were developed:

- InitialFilter : returns a list of contained name initials filtered by specified initials
- ConcatFilter : concatinates two array to execute ng-repeat on multiple arrays
- SoundexFilter : filters array of objects by soundex of their name property

The architecture of the application is constructed of two main packages containing the controllers and the services implemented for the application. The controllers are used to offer the interface for the front end to trigger the processes implemented in the services of the packages. Thereby two controllers each dedicated to either read or update features, were created. The data displayed to the user is managed through the *DataService*,

the *SelectionService* and the *ModificationService*, which access the data saved in the application's global arrays. To access the client's database API the *IndexedDbService* offers an AngularJS conform API.

5.3.2 The User Interface

Due to the single page nature of the floor plan mapper, a classical dialog structure diagram cannot properly describe the dialog flow of the application. Figure 5.5 shows the structure of the UI in an adapted dialog structure diagram, each node representing a UI state. The states are each assigned to either the *ViewController* or the *EditController* (see section 5.3.1).

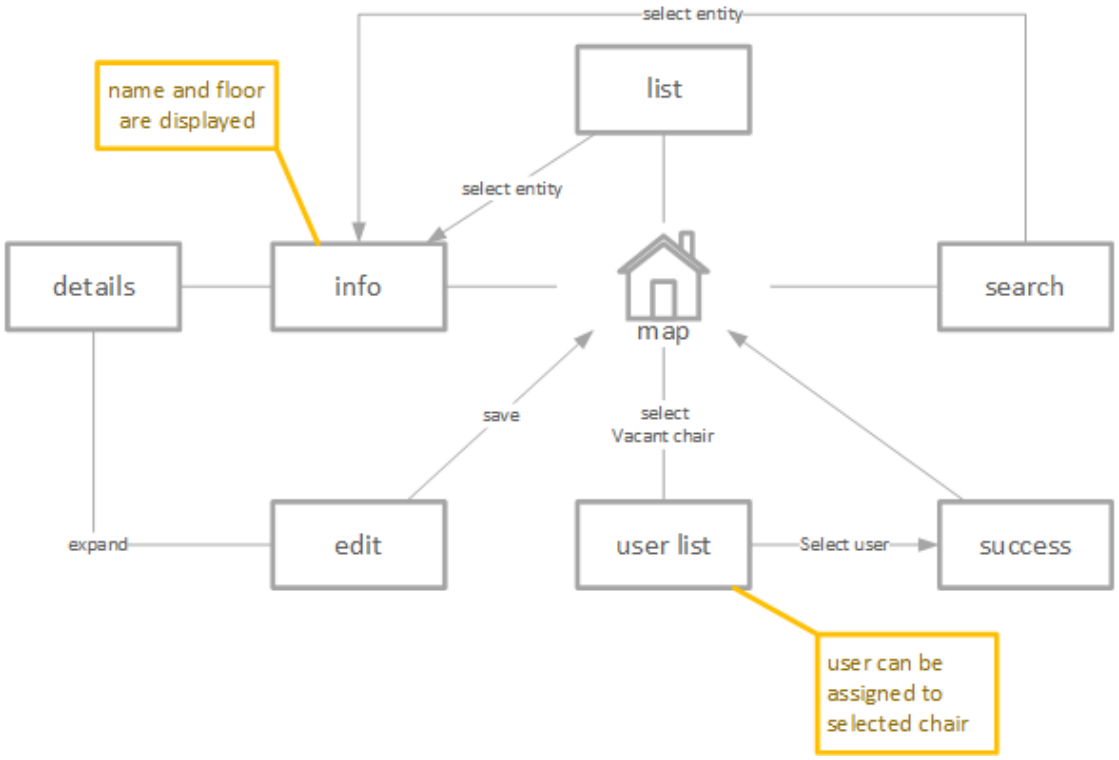

Figure 5.5: Dialog Structure of the floor plan mapper

Used Technologies

For the implemenation of the application's user interface two technologies have been reviewed: Angular Material [22] and MaterializeCSS[23]
Because of issues with touch gestures on iOS devices with Angular Material, MaterializeCSS was used to create the user interface of the floor plan mapper. Materialize is a front-end library using jQuery as dependency and allowing easy adaption of the offered material design components using SASS [24]. The administration interface was built without major modifications of the components, since the focus of the project was put on creating a self maintained system. On the user side of the application, the floor plan mapper was modified to fit the project's requirements and to offer a fluid user experience.

In the following section the challenges and the used concepts to solve these challenges during the UI desgin phase will be presented. It describes how the application's general design was created to improve the overall user experience.

UI Design

Since google's material design guideline [25] has gained a major acceptance from the targeted user group [26], it was chosen to be used as base of the design for the floor plan mapper. The demanded simplicity, given color and font scheme were maintained, without the need to deviate from the corporate identity. The *Material Design* guidelines allow a color scheme of two colors: a primary color and an accent color. For this application the primary color *blue* and the accent color *red* where chosen from the *Material Design* color palette. Only being used to indicate selected chairs and conference rooms, the accent color only has a minor impact on the UI appearance. The interactive components follow the material paradigm of flat surfaces with minimalistic animations. Therefore the used buttons are limited to the circular speed-dial control to select a floor and icons provided by the material icon library [27].

In the UI the text font *Roboto* and the dynamic text size ensures the readability of text on all devices. The constrain of a minimum text size however results in overlowing issues on

longer data values. To solve this issue, the application design avoids having dynamic text values on the same line as absolute positioned components. This enables linebreaks in the datafields without affecting the overall design (see Figure 5.6).

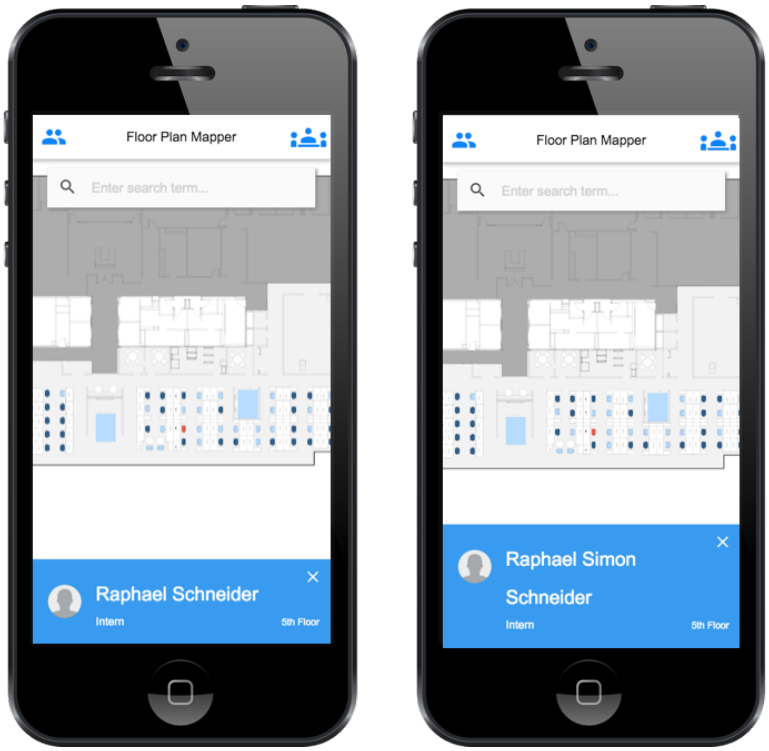

Figure 5.6: Overflowing of data fields

When launching the application, a limit of loaded images was decided. Once the user interacts with the application, more images will be loaded. In the case of missing images, a default image will be shown. This helps minimize the network traffic and improves the application's startup time.

In the edit mode of the application the user has the possibility to edit the data for a given employee or conference room. To prevent the user from entering too long or syntactically incorrect values in the textfields, inputs are properly restricted to the specified input types triggering the user's device to open either a text, mail or number keyboard, when selecting a specific input in the form.

5.4 Implementation Concepts

In order to integrate the features defines in the requirements (see section 3.10) a number of concepts needed to be developed. This section covers these concepts and evaluates their qualities.

5.4.1 Polyfill for Client Database

Since the support for the IndexedDB in the *iOS and Safari browser* is limited the intended single *IndexedDbService* had to be wrapped in another service to polifill the features needed for the application. To solve this issue the service *PolifillDatabase* (see Figure 5.7) has been implemented. It detects iOS and Safari devices using the JavaScript library modernizr and returns the unified API of either the *IndexedDbService* or the *WebSqlService*. This enables the application to use the *PolifillDatabase* to access the local database, independent of the used Web API.

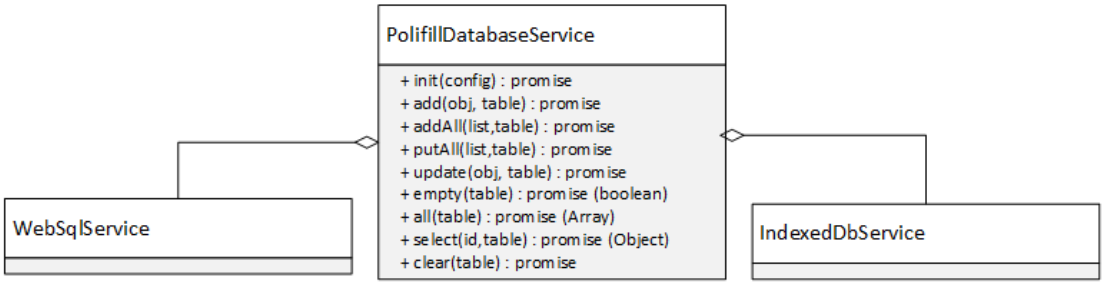

Figure 5.7: Implementation of PolifillDatabase

In order to offer a uniform API interface the versioning mechanism build into the *IndexedDB API* needed to be reproduced in the *WebSqlService*. As reccommended in Jannuary 2015 by the W3C [28] the IndexedDB offers a versioning system for changes on the database structure. The implementation uses a version number, which is used to detect a change in the database version and trigger a database reinitialisation. To incorporate this feature into the *WebSqlService*, a version number was added to the client's local storage using the *LocalStorageService* (see Figure 5.4). The service checks the local storage attribute on service initialisation and reacts accordingly.

5.4.2 Caching Strategy and Offline Mode

A data-driven mobile application requires a well-thought-out caching strategy, in order to offer an optimal and fluid user experience. In the case of the floor plan mapper a combination of server and client caching has been applied. On the server side Java Spring Boot's native CacheService offers a simple API to cache separate requests. This can be used for the fetching of data from the server and eliminates unnecessary requests to the server database.

In Addition to the server side caching, the mobile application uses the *Web Storage API*, *IndexedDB API* and the *WebSql API* to implement a client side cache. This enables the application to minimize data trafic and occasionally run without connecting to the data server at all. Depending on the used browser, the application uses either the *WebSQL API* or the *IndexedDB API* to save the fetched data sets persistently. In Addition to the persisting of data, the application saves a timestamp of the last update using the *LocalStorageAPI*. On startup the application sends the timestamp of the last update to the server, who returns a list of tables, which have been changed since the given timestamp.

Since a daily application, which requires a permanent network connection leaks practicality the implementation of the caching algorithm additionally enables the offline usage of the floor plan mapper. As intended in the requirements the application uses the local database as a fallback, if a connection to the server fails. The provided manifest file contains a list of files, which are needed for the offline usage of the application. This enables the used browser to keep these files in cache.

5.4.3 The Search Algorithm

To support the search process of both conference rooms and users, the requirements (se chapter 3.10) ask for a free-text search. However, a simple search field filtering the list of conference rooms and users can lead to search failure. Due to different spellings of multicultural names, the search results need a suitable filtering algorithm.

The floor plan mapper solves this issue with a custom implementation of an Angular filter. The filter utilizes the soundex algorithm [29] to find the entries with similarly sounding names instead of checking for exact spelling. After the relevant subset of entries has been found, the edit distance between the search term and the entry names is evaluated. This value is then used to sort the result list and display the closes matches first. Figure 5.8 illustrates the dataflow of this algorithm.

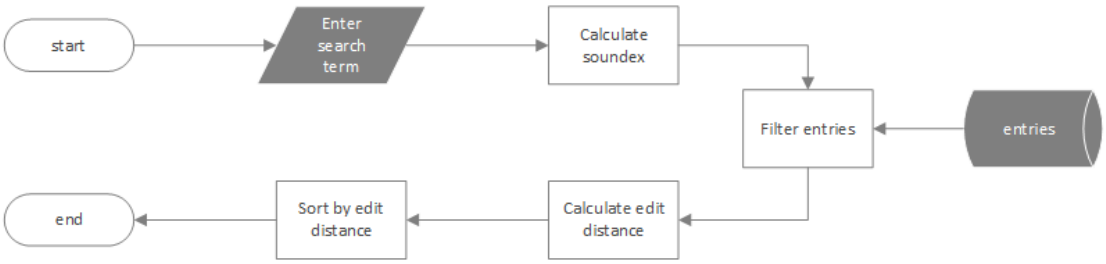

Figure 5.8: Search Algorithm using Soundex and Edit Distance

5.4.4 Modeling of UI states of components

As AngularJS does not nativly offer support for a multi state container a new concept had to be developed in order to implement the one-page interface. For this purpose the *StateContainer* directive was developed. It represents a container, which references a number of *State* directives and only displays the currently active state. The use of Angular transclusion enables the nesting of the *State* tags in the *StateContainer* tags. To change the state of the container, the *StateContainer* directive defines a bindable function, which enables the controller access it.

5.4.5 Device Rotation Indicator

The user's position in the building can be separated in location coordinates and orientation in relation to the floor plan. The device orientation feature has been implemented using the *DeviceOrientation* event specified by the W3C [30]. It is used to rotate the map shown in the application to the angle given in the *DeviceOrientation* event object (see Figure 5.9). To hide the noise in the values given by the device sensors the rotation steps were limited to a 45 degree intervals.

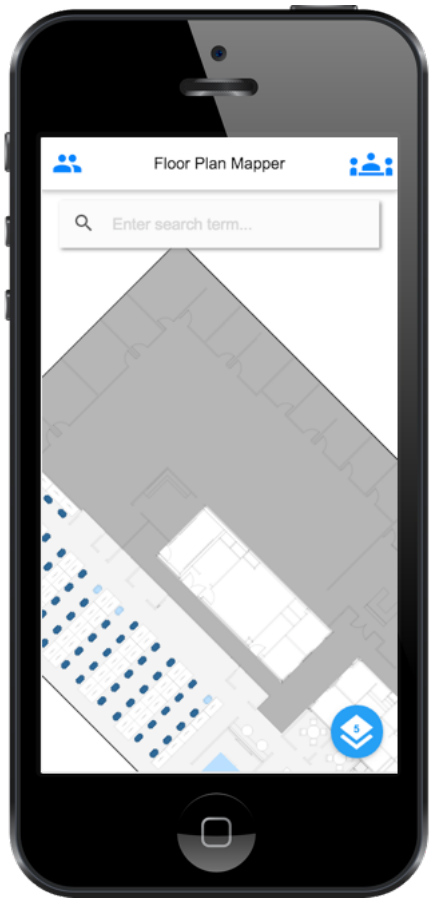

Figure 5.9: Dynamic Map Orientation adapted to Device Orientation

6

Requirements Evaluation

In this chapter the finished application and implementation concepts are reviewed and the requirements coverage will be evaluated. During the course of development some unexpected issues and challenges occured which needed to be solved and effected the outcome of the project covered by this thesis. The goal of the chapter is to give a detailed view on the solved just as unsolved challenges during the project and the changes they caused on the requirement coverage. Additionally, chapter 7 summarizes the insights gained during the development.

In table 6.1 the coverage of functional and non-functional requirements is evaluated. In general, a high coverage of the functional requirements can be found. Most issues occured due to platform specific behaviour, which could not be handeled properly because of the limited time frame and test user base.

Requirement	Fulfilled in %
FR00 View Floors	100%
FR01 (Un-)occupied Seats	100%
FR02 View Conference Rooms	80%
FR03 View Entity Room Data	100%
FR04 Free-text Search	100%
FR05 Employee & Room List	99%
FR06 Offline Mode	100%
FR07 Indicator for Device Rotation	70%
FR08 Administrational Tool	50%
FR09 Community-based Seat Management	100%
FR10 Community-based Data Management	100%
NR01 High Compatibility	100%
NR02 Accessibility	50%
NR03 Fluid UI and Performance	90%
NR04 Data Safety	100%
NR05 Data Security	100%
NR06 Usage of the MVVM Pattern	100%
NR07 Usage of Dependency Injection	100%
NR08 Intuitive User Interface	99%
NR09 Device Specific Gestures	100%
NR10 Corporate Design	100%
NR11 Minimize Data Traffic	100%

Table 6.1: Summarizing Table with Requirement Evaluation

FR02 View Conference Rooms

The conference rooms are displayed in the user interface as rectangles of uniform size. These rectangles have been varied in size throughout the development and testing phase to find the ideal size to minimize the number of wrong selections on touch devices. Ideally the size of rooms should be individually adjustable to fit the represented room. However this feature was not implemented due to the high data volume and higher maintenance effort.

FR05 Employee & Room List

The list of employees has been separated into two sublists as proposed in the requirements. As intended in the UI design a separation in three sublists has been applied. However, it was discovered that the large number of employees requires finer partitioning in order to offer a usable browsing tool. The implementation of a sublist per existing initial would therefore be an obvious improvment.

FR07 Indicator for Device Rotation

The implementation of the map rotation according to the device rotation caused usability issues, due to the sudden rotation events. Intended to help with the orientation in the building the 45 degree rotation steps were discovered to cause the users to loose track of their position instead. To solve these issues it would be possible to decrease the angle intervals. However, this results in a bad user experience on devices with lower quality sensors due to the noise in the measurements. Instead a small orientation indicator as shown in the bottom left corner of Figure 6.1 can solve this issue.

FR08 Administrational Tool

Apart from the comunity-based administration a separate administrational interface was requested by the initial requirements. However, this feature had to be deferred due to

Figure 6.1: Device Rotation Indicator

time restrictions. As of the point of this thesis, a REST interface has been implemented supporting all administrational functionalities. The userinterface incorporating these features has not been implemented.

NR02 Accessibility

Because of privacy concerns and updated requirements the application has only been made available to a limited set of test users. In future the application will be accessible to all devices on the corporate network. To achieve the requirement of accessibility for all devices in the building, more work needs to be put in evaluating the confidentiality level of the data displayed.

NR03 Fluid UI and Performance

In General the application offers a fluid user experience and UI performance on mobile devices. In contrast to mobile platforms, Internet Explorer 11 shown some performance issues in comparison to Chrome or Safari.

NR08 Intuitive User Interface

The application has recieved positive user feedback during the testing phases. However, continuous adaptions to user feedback is crucial to maintain the user base.

7

Summary

After having covered the aspects of the concepts and technical implementation of the floor plan mapper, this chapter is dedicated to summarise the general challenges throughout the project and gives a proposal for future improvments of the process. It starts out with a general overview of the development related challenges which occured during the project. This aspect is evaluated furthermore in section 7.2. Finally, an outlook over possible future features of the floor plan mapper will be introduced and evaluated in prospect to the current organizational and technical environment.

7.1 Organizational and Technical Challenges

Due to the dynamic process model used in the development for this thesis the major challenge was to organise the communication with the supervisors. The on-going move of the facility therefore caused delays on agreements and the development progress. To counteract this leak of personel availability a focus had to be set on the maintainablility, extendability and documentation of the application. In Addition the variable behaviour across the used browsers and devices demanded additional work on testing and debugging the application. During the time of pending descisions the focus has been set to these issues. Finally, after completing the core features of the application, the privacy constrains had to be reevaluated. Pubilshing the data shown in the floor plan mapper as requested by the requirements, demands the evaluation of the privacy level needed for the data and authorisation, which has not been given within the time frame of this thesis. As a result the application has only been published for users within the facility.

7.2 Lessons Learned

As the work of this thesis has shown, the concept of a mobile floor plan application has some definite advantages compared to the solution of printed floor plans. The offered interactivity has shown to offer a multitude of unseen use cases and can be extended to be a useful tool for many user groups. The concept of comunity based data management can highly improve the update times of data and potentially delegate certain organizational tasks to the users. From a user perspective the feedback on the self maintainable user information has been positive, confirming this statement furthermore. Unfortunatelly organizational barriers slowdown the process of integrating this data management concept in the corporate environment, as the concept violates the hierarchical organisation found in large organisations.

7.3 Project Outlook and Related Use Cases

During the work on this thesis and the related web application, new features and use cases have been introduced and requested for future evaluation. In this section these new proposals will be presented to give an outlook in the project's future development and possible new application areas on which the concepts can be applied to.

As already suggested in section 3.8, the conference room reservation system is planned to be integrated into the floor plan mapper in a future iteration. This will offer a fast way to execute this recurring task and give the users another reason to come back to the application in future.
Additionally, an integration of social networks to enable messaging and better information about the employees was proposed. The LinkedIn API [31] can be used to create an employee search by skill to help find the right contact person in case of any issues.
Concering the privacy issues, the public availability for guest users is still in converstation as covered in section 7.1. However the goal is set to make the application available to guest users just as employees, to maximize the user group.
From an organizational point of view the in door locationing as discussed in section 3.6

is still a much demanded feature. Until the technology needed for this feature reaches an acceptable price to equip an entire facility, this requirement needs to be layed back however. As an alternative ideas such as giving orientation points throughout the building, such as QR tags in the elevators and a redesign of a orientation indicator have been in discussion.
Finally the possibility to assign a person to a seat for a limited and optionally recurring time interval has been put into the future score. This feature will enable the efficient management of cubicals for commuting employees and may replace the centralized cubical assignment in future releases.

7.4 Conclusion

As of today, employees working in offices are faced with more and more frequent changes in the workplace. To support employees cope with adjustments in the floor layout or even relocation to another building, this thesis is intended to introduce the concept of a interactive, mobile floor plan. As described in the preceding chapters, the floor plan mapper implemented for this thesis aims to provide the support needed to adapt to these changes. With the application users can find collegues and conference rooms in the building. To maintain the data displayed in the application the concept of comunity driven data management was developed. It enables the individual employee to self maintain the own data.

Form an organizational point of view the purpose of this thesis is to simplify facility management and enable a dynamic use of the office area. The application successfully proves, that the concepts used can help create a more dynamic workspace and produce possibilities to build further features to increase the flexibility in the office.

Bibliography

[1] Apple Inc.: iPhone-Modell bestimmen. (2015) https://support.apple.com/de-de/HT201296 (Last accessed on 2016-13-04).

[2] PewResearchCenter: U.S. Smartphone Use in 2015. (2015) http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/ (Last accessed on 2016-12-04).

[3] Bitkom e.V.: 44 Millionen Deutsche nutzen ein Smartphone. (2015) https://www.bitkom.org/Presse/Presseinformation/44-Millionen-Deutsche-nutzen-ein-Smartphone.html (Last accessed on 2016-12-04).

[4] Schickler, M., Reichert, M., Pryss, R., Schobel, J., Schlee, W., Langguth, B.: Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health. Springer-Verlag (2015)

[5] Dahlman, E., Parkvall, S., Skold, J., Beming, P.: 3G Evolution: HSPA and LTE for Mobile Broadband. Number 2. Academic Press (2008)

[6] Schickler, M., Pryss, R., Schobel, J., Reichert, M.: An engine enabling location-based mobile augmented reality applications. In: 10th International Conference on Web Information Systems and Technologies (Revised Selected Papers). Number 226 in LNBIP. Springer (2015) 363–378

[7] Schobel, J., Schickler, M., Pryss, R., Nienhaus, H., Reichert, M.: Using vital sensors in mobile healthcare business applications: Challenges, examples, lessons learned. In: 9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps. (2013) 509–518

[8] Nah, F.F.H., Siau, K., Sheng, H.: The value of mobile applications: a utility company study. Communications of the ACM - Medical image modeling **48** (2009) 85–90

[9] Geiger, P., Schickler, M., Pryss, R., Schobel, J., Reichert, M.: Location-based mobile augmented reality applications: Challenges, examples, lessons learned. In: 10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps. (2014) 383–394

[10] eMarketer: Internet usage (engagement) growth solid. (Technical report) http://de.slideshare.net/kleinerperkins/internet-trends-v1/14-14Internet_Usage_Engagement_Growth_Solid11 (Last accessed on 2016-14-04).

[11] John Gossman: Introduction to Model/View/ViewModel pattern for building WPF apps. (2005) http://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/ (Last accessed on 2016-05-04).

[12] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE (2000)

[13] Douglas Crockford: RFC 7159: The JavaScript Object Notation (JSON) Data Interchange Format. (2014) https://tools.ietf.org/html/rfc7159 (Last accessed on 2016-14-04).

[14] Google Inc.: AngularJS Framework. (2010) https://angularjs.org (Last accessed on 2016-28-02).

[15] Schobel, J., Schickler, M., Pryss, R., Reichert, M.: Process-driven data collection with smart mobile devices. In: 10th International Conference on Web Information Systems and Technologies (Revised Selected Papers). Number 226 in LNBIP. Springer (2015) 347–362

[16] Schobel, J., Schickler, M., Pryss, R., Maier, F., Reichert, M.: Towards process-driven mobile data collection applications: Requirements, challenges, lessons learned. In: 10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps. (2014) 371–382

[17] Galen Gruman: What you need to know about using Bluetooth beacons. (2014) http://www.infoworld.com/article/2608498/mobile-apps/what-you-need-to-know-about-using-bluetooth-beacons.html (Last accessed on 2016-13-04).

[18] Estimote, Inc: How do I get started? (2012) http://estimote.com (Last accessed on 2016-14-04).

[19] PAAS, F., RENKL, A., SWELLER, J.: Cognitive load theory: Instructional Implications of the interaction between information structures and cognitive Architecture. Instructional Science (2004)

[20] Fisher, P.T., Murphy, B.D.: 2. In: Spring Persistence with Hibernate. (2010) 27–28

[21] Lerner, A.: ng-book. FULLSTACK.io (2013)

[22] Google Inc.: Angular Material Design. (2014) https://material.angularjs.org/latest/ (Last accessed on 2016-28-02).

[23] Alvin Wang and Alan Chang and Alex Mark and Kevin Louie: Materialize. (2014) http://materializecss.com (Last accessed on 2016-28-02).

[24] Hampton Catlin and Natalie Weizenbaum and Chris Eppstein: Syntactically Awesome Style Sheets. (2006) http://sass-lang.com (Last accessed on 2016-28-02).

[25] Google Inc.: Material Design Guidelines. (2014) https://www.google.com/design/spec/material-design/introduction.html (Last accessed on 2016-23-01).

[26] Android: Anteil der verschiedenen Android-Versionen an allen Geräten mit Android OS weltweit. (2016) http://de.statista.com/statistik/daten/studie/180113/umfrage/anteil-der-verschiedenen-android-versionen-auf-geraeten-mit-android-os/ (Last accessed on 2016-10-04).

[27] Google Inc.: Google Material Icons. (2014) https://design.google.com/icons/ (Last accessed on 2016-23-01).

[28] World Wide Web Consortium: Indexed Database API Specification. (2015) `https://www.w3.org/TR/IndexedDB/` (Last accessed on 2016-12-04).

[29] Russell, R., Odell, M.: Soundex. US Patent **1** (1918)

[30] World Wide Web Consortium: DeviceOrientation Event Specification. (2011) `https://www.w3.org/TR/orientation-event/` (Last accessed on 2016-12-04).

[31] LinkedIn Inc.: LinkedIn API. (2016)

List of Figures

Name: Raphael Schneider Matrikelnummer: 792210

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Raphael Schneider