Universität Osnabrück

BACHELORARBEIT

Thema:

Prototyp einer mobilen Applikation zur Koordination der Mitarbeiter in einem Catering-Betrieb

Erstellt am 23. Oktober 2018

Vorgelegt von:

Darren Kopatz
Natruper Str. 214
49090 Osnabrück
dkopatz@uos.de

Inhaltsverzeichnis

23. Oktober 2018

Der Catering-Betrieb des VFL Osnabrück, die Eiken & Grosch Stadion GbR, möchte eine mobile Applikation einsetzen, um jederzeit Aufgaben-, Stück- und Inventarlisten zu erstellen, zu verwalten und zu synchronisieren. Dies soll mit dem Ionic-Framework realisiert werden, damit die Betriebssysteme Android und iOS gleichermaßen unterstützt werden können. Für den Datenaustausch der einzelnen Mitarbeiter dient eine zentrale Datenbank. Zusätzlich soll ein weiteres Programm zur Verwaltung dieser Datenbank erstellt werden.

1 Einleitung und Motivation

Das folgende Kapitel soll in das Thema einführen, indem Vorteile einer mobilen Applikation für den Betrieb herausgearbeitet werden.
Im ersten Unterkapitel werden einige Probleme innerhalb des Betriebes dargelegt. Das dritte Kapitel beschreibt Anforderungen, die an die mobile Applikation gestellt werden und das vierte Unterkapitel stellt bereits existierende Applikationen vor und erklärt, wieso die Implementation aller Anforderungen trotzdem sinnvoll ist.

1.1 Problemlage im Betrieb

Die Probleme und Arbeitsabläufe, die im folgenden Unterkapitel beschrieben und erläutert werden, beruhen auf mehreren Monaten Berufserfahrung innerhalb des Betriebes und Gesprächen mit dem Bereichsleiter für Personal sowie Verwaltung und Einkauf von Rohstoffen, Herrn Rene Rosenwald.
Für die Eiken & Grosch Stadion GbR arbeiten an einem Spieltag des VFL Osnabrück zwischen 40 und 60 Mitarbeiter in mehreren Verkaufsständen, davon sind etwa 15 Arbeiter an der Vor- und Nachbereitung dieses Spieltages beteiligt.
Zur Vorbereitung erstellt der Bereichsleiter zunächst eine Liste, auf der die für diesen Spieltag anstehenden Aufgaben festgehalten werden. Diese Liste hat an jedem Spieltag größtenteils denselben Inhalt mit einigen, wenigen Aufgaben, die nur an einem einzigen Spieltag ausgeübt werden müssen. Von diesen sich wiederholenden Aufgaben werden nun gegebenenfalls manche Aufgaben vergessen zu notieren, wodurch entweder der Arbeitsfluss gestört wird oder diese Aufgaben anschließend nachgearbeitet werden müssen.
Die Aufgabenliste, die so erstellt wurde, wird nun im Büro ausgelegt und die verschiedenen Mitarbeiter haken die einzelnen Arbeitspunkte nach dem Erledigen ab. Da die Mitarbeiter sich hierfür immer wieder zum Büro begeben müssen, entstehen dadurch unnötige Laufwege und zusätzliche Personalkosten.
Einer der Punkte auf der Aufgabenliste ist es, die verschiedenen Verkaufsstände zu bestücken. Hierfür werden am Anfang des Arbeitstages mehrere Bestückungslisten ausgeteilt. Allerdings kann sich diese Liste im Laufe des Arbeitstages noch ändern. Diese Änderung findet dann auf einer der Bestückungslisten von einem Mitarbeiter statt und wird dann an die anderen mündlich weitergetragen. Diese Informationen erreichen allerdings meistens nicht alle Mitarbeiter, weshalb es nun zu Fehlbestückungen kommen kann, die dann nachträglich korrigiert werden müssen.
Zur Nachbereitung müssen die übrig gebliebenen Waren nun wieder in die Lager eingeräumt und inventarisiert werden. Die Bestände werden hierfür per Hand auf verschiedenen Vordrucken eingetragen. Die Vordrucke werden anschließend in eine Exceltabelle übertragen. Hierbei treten diverse Übertragungsfehler aufgrund von Lesbarkeit und Tippfehlern auf.

1.2 Anforderungen an die Applikation

Da in Kapitel 1.1 diverse Probleme aufgezeigt wurden, stellt sich nun die Frage, wie sich diese innerhalb einer mobilen Applikation lösen lassen. Dazu werden in den folgenden Unterkapiteln verschiedene Anforderungen an die mobile Applikation vorgestellt. Hierbei wird zuerst die Anforderung beschrieben und erläutert, wieso diese einen Vorteil bietet. Danach wird kurz beschrieben, wie die Anforderung in der mobilen Applikation in etwa umgesetzt werden könnte.

1.2.1 Aufgabenlisten

Die mobile Applikation soll unter anderem die bisherigen Aufgabenlisten ersetzen. Hierzu müssen alle Aufgaben für alle Nutzer synchronisiert angezeigt werden und ungelöste Aufgaben müssen als gelöst markiert werden können. Dadurch werden unnötige Laufwege eingespart und alle können die Aufgabenliste jederzeit einsehen um weitere Arbeitsschritte zu planen. Eine synchronisierte Aufgabenliste kann innerhalb einer Tabelle „aufgabenlisten“ eines Datenbanksystems gespeichert werden. Dafür muss die Aufgabe selbst und der Status dieser z.B. in der Form „gelöst“ oder „ungelöst“ festgehalten werden. Um den Status zu aktualisieren, muss es einen Knopf bei den „ungelösten“ Aufgaben geben, der eine entsprechende Anfrage an die Datenbank weiterleitet. Der Inhalt dieser Tabelle könnte nun innerhalb der mobilen Applikation ausgelesen und die Aufgaben könnten untereinander in Form einer Liste angezeigt werden.
Zusätzlich sollte bei den „gelösten“ Aufgaben die Möglichkeit bestehen, dass man nachvollziehen kann, welcher Mitarbeiter den Status einer Aufgabe verändert hat, damit bei Fehlern der zuständige Mitarbeiter auf diese hingewiesen werden kann. Dies kann umgesetzt werden, indem jedem Mitarbeiter vor dem Benutzen der Applikation ein Nutzername zugewiesen wird, welcher automatisch beim Ändern des Status in einer zusätzlichen Spalte innerhalb der Tabelle „aufgabenlisten“ eingetragen wird.
Um redundante Arbeit zu ersparen, sollten wiederkehrende Aufgaben gespeichert und automatisch zu jeder Aufgabenliste hinzugefügt werden. Da die Spieltage immer in unregelmäßigen Abständen stattfinden, ist es nicht ausreichend, die Aufgaben regelmäßig in vorher definierten Abständen anzuzeigen. Es muss stattdessen möglich sein, immer wieder einen bestimmten Tag zu wählen, an dem die Aufgaben angezeigt werden. Dies kann mit einer zusätzlichen Tabelle „aufgaben“ innerhalb des Datenbankensystems umgesetzt werden. Die wiederkehrenden Aufgaben werden in dieser Tabelle gespeichert und beim Erstellen einer Aufgabenliste wird diese Tabelle ausgelesen und so hinzugefügt. Um diese Aufgaben zu Speichern würde sich ein Java GUI eignen, auf welcher die Aufgaben in ein Textfeld eingegeben und anschließend per Knopfdruck abgesendet werden könnten. Außerdem könnten in dieser GUI auch die Daten für die verschiedenen Arbeitstage ausgewählt und abgeschickt werden.
Des Weiteren sollten auch einmalige oder unregelmäßige Aufgaben zu der Aufgabenliste hinzugefügt werden können, damit die Aufgabenliste vollständig auf dem Smartphone ausgelesen werden kann. Dies kann innerhalb der mobilen Applikation mit einem Textfeld und einem dazugehörigen Button zum Absenden umgesetzt werden. Dadurch wird auch ein spontanes Hinzufügen von unvorhergesehenen Aufgaben ermöglicht.

1.2.2 Bestückungslisten

Da es den Mitarbeitern, wie in Unterkapitel 1.1 dargestellt, bisher nur erschwert möglich ist, ihre eigene Bestückungsliste bei Änderungen auf dem aktuellsten Stand zu halten, erscheint es sinnvoll, innerhalb der Applikation eine synchronisierte Bestückungsliste zur Verfügung zu stellen. Dies kann innerhalb des Datenbanksystems mit einer Tabelle „bestückungslisten“ umgesetzt werden. Der Inhalt der Tabelle kann dann als Liste innerhalb der Applikation angezeigt werden.
Um die Bestückungsliste aktualisieren zu können, sollte es außerdem eine Funktion geben, mit der die Bestückungen direkt am Smartphone geändert werden kann. Einem Nutzer diese Option am Smartphone zu bieten erscheint sinnvoll, weil hier die Bestückung direkt eingesehen und dann gleichzeitig geändert werden kann, wodurch eine höhere Benutzerfreundlichkeit erzielt werden kann. Dies lässt sich umsetzen, indem die verschiedenen Bestückungsmengen als Textbox angezeigt werden, welche sich ändern lassen. Dazu kann man einen Knopf zur Verfügung stellen, welcher die neuen Mengen an die Datenbank weiterleitet.
Da meistens mehrere Stände gleichzeitig bestückt werden, erscheint eine Funktion, die die Bestückungen bestimmter Stände automatisch addiert, sinnvoll. Dies würde den Mitarbeitern die Zeit ersparen, in der diese sonst die Bestückungen selbst addieren müssten. Außerdem werden so mögliche Fehler bei dieser Tätigkeit ausgeschlossen. Umgesetzt werden könnte dies entweder, indem man in der Applikation den Nutzer zu addierende Stände markieren lässt und dieser anschließend die Summe der markierten Stände aus der Datenbank abfragt und ihm die Summe anschließend angezeigt wird. Alternativ kann man dem Benutzer eine Auswahl von Standgruppen geben, die mit einem einzelnen Knopfdruck zusammenaddiert werden. Bei den beiden Optionen stehen sich Flexibilität und Simplizität gegenüber. In diesem Anwendungsfall erscheint die zweite Option sinnvoller, da in der Regel immer dieselben Stände gleichzeitig bestückt werden.

1.2.3 Inventurlisten

Um die Probleme aus Unterkapitel 1.1 in Bezug auf die Inventurlisten zu beheben, bietet es sich an, auch diese zu der mobilen Applikation hinzuzufügen. So können die Probleme mit handschriftlichen Inventuren entgegengewirkt werden. Diese Funktion kann mit einer Tabelle „inventurlisten“ umgesetzt werden. Zunächst sollte diese Tabelle nur die Namen der Produkte ohne eine Menge enthalten. Die Menge muss dann innerhalb der Applikation über Textboxen und einem Knopf zum Aktualisieren der Daten eingefügt werden.
Bei den Inventurlisten für die Lebensmittel kommen häufig auch Produkte auf, die zuvor noch nicht auf der Liste eingetragen waren. Um auch diese Produkte problemlos auf den Inventurlisten festhalten zu können, wäre eine Funktion sinnvoll, die den Nutzern ermöglicht neue Produkte auf der Liste zu notieren. Hierfür wäre ein Knopf denkbar, der dem Nutzer ein Fenster öffnet, in welchem dieser Name und die Menge des Produktes eingeben kann und diese Daten nach anschließender Bestätigung an die Datenbank weitergeleitet wird.
Damit die Eingaben in der Datenbank auch wieder fehlerfrei übertragen werden können, sollte es eine Funktion geben, die die Inventurdaten wieder aus der Datenbank ausliest. Dies könnte in derselben Java Applikation umgesetzt werden, wie aus Unterkapitel 1.2.1. Hier sollte man dem Benutzer einen Pfad auswählen lassen, in dem die Daten gespeichert werden und ihn anschließend per Knopfdruck die Daten auslesen lassen.

1.2.4 Vertraulichkeit der Daten

Da die Inhalte der Applikation vertraulich behandelt werden sollen, dürfen zu den Inhalten nur autorisierte Nutzer Zugang haben. Dies kann ganz klassisch durch eine Passwortsperre geregelt werden. Das Passwort sollte vor dem Benutzen der Applikation eingefordert werden und lediglich nach der erfolgreichen Eingabe des Passwortes darf Zugang zu den Inhalten der Applikation gewährt werden.
Damit den Nutzern diese Rechte wieder entzogen werden können, zum Beispiel für den Fall, dass ein Mitarbeiter aus der Firma ausscheidet, wäre es sinnvoll, jedem Nutzer einen eigenen Benutzernamen mit Passwort zuzuweisen. Das Zuweisen von Benutzerkonten kann man ebenfalls innerhalb der Java Applikation aus Unterkapitel 1.2.1 realisieren.

1.2.5 Import von bereits vorhandenen Listen

Bestückungs- und Inventurlisten existieren bereits von vorherigen Spieltagen in Form von Exceldokumenten. Es wäre sinnvoll, diese ebenfalls in der mobilen Applikation einbinden zu können. Hierfür wäre es am einfachsten, wenn der Nutzer innerhalb der Java Applikation den Pfad zu einem Dokument angeben kann und der Inhalt in diesem Dokument anschließend per Knopfdruck eingelesen und an die Datenbank weitergeleitet wird. Diese Funktion kann ebenfalls dafür verwendet werden, um neue, leere Inventurlisten in die Datenbank hochzuladen, damit diese von den Mitarbeitern einfach in der mobilen Applikation ausgefüllt werden können.

1.3 Existierende Programme

Das vorausgegangene Unterkapitel 1.2 hat unter anderem Gründe genannt, die dafür sprechen, dass eine mobile Applikation den Arbeitsalltag innerhalb des Betriebes vereinfachen könnte. Nun wäre es naheliegend, dass bereits eine mobile Applikation existiert, welche einige der oben genannten Anforderungen bereits vollständig abdeckt. Dann wäre eine Implementation dieser unter Umständen redundant und damit nicht sinnvoll. Deshalb soll das folgende Kapitel andere Applikationen auf diese Punkte untersuchen. Dabei wird sich vor allem auf die Anforderungen der Aufgabenliste beschränkt, da sich die Anforderung der Bestückungsliste und der Inventurliste auf die Inhalte der privaten Exceldokumente aus den Abbildungen 7, 8 und 9 beziehen, weshalb eine Applikation, die diese Ein- und/oder Auslesen kann, nicht möglich erscheint.
Das folgende Kapitel stellt verschiedene Applikationen vor, die ähnliche Funktionen wie die oben beschriebene Aufgabenliste haben. Anschließend wird begründet, warum es trotz der genannten Applikationen sinnvoll ist, eine Aufgabenliste zu implementieren. Hierbei werden die jeweiligen Programme mittels festgelegter Kriterien, die sich aus Unterkapitel 1.2.1 ergeben, verglichen. Zum einen wird geprüft, ob die jeweilige Applikation eine für mehrere Nutzer synchronisierte Liste anbietet. Des Weiteren sollte es möglich sein, eine Liste von wiederkehrenden Aufgaben einmalig zu erstellen und dann in unregelmäßigen Zeitabständen wieder aufzurufen und bei den erledigten Aufgaben muss nachvollzogen werden können, wer diese gelöst hat. Da auch allgemein gefordert wird, dass die Applikation sowohl mit Android als auch mit iOS kompatibel ist, muss dies auch für die hier getesteten Apps der Fall sein. Zuletzt geht noch aus Unterkapitel1.2.1 hervor, dass man den Nutzern die Rechte, die Inhalte der Liste einzusehen, wieder entziehen können muss.

Da alleine der Google Play Store nach Recherche des Verfassers über 100 Ergebnisse bei der Suche nach „Aufgabenlisten“ ausgibt, wurden in dieser Liste nur kostenlose Applikationen mit über 5 000 000 Downloads berücksichtigt. Es ist daher anzumerken, dass es noch weitere Applikationen mit ähnlichen Funktionen gibt, die gegebenenfalls auch mehr oder andere dieser Kriterien erfüllen könnten.

1.3.1 Wunderlist

Wunderlist ist eine Applikation der 6 Wunderkinder GmbH, welche es dem Nutzer erlaubt, mit jedem Gerät synchronisierte Aufgabenlisten zu erstellen, welche auch mit anderen Nutzern geteilt werden können und sowohl auf iOS als auch auf Android verfügbar ist[1]. Durch Herunterladen und Testen der Applikation konnte herausgefunden werden, dass sich auch wiederkehrende Aufgaben einrichten lassen. Allerdings ist dies nur in regelmäßigen Abständen (täglich, wöchentlich, alle drei Tage) möglich. Außerdem konnten hinzugefügte Nutzer auch wieder vom Ersteller der Liste entfernt werden. Allerdings konnte bei eigenen Tests nicht nachvollzogen werden, wer welche Aufgabe gelöst hat.

1.3.2 Any.do

Die Planungs-Applikation Any.do bietet ebenfalls eine Funktion für Aufgabenlisten an. Hier können Listen erstellt werden, die ebenfalls mit Nutzern auf Android und iOS Geräten geteilt werden können[2]. Innerhalb der Applikation kann man außerdem Aufgaben lediglich in einem regelmäßigen Rhythmus wiederkehren lassen und Nutzer konnten weder aus Listen, zu denen sie hinzugefügt wurden, vom Ersteller entfernt werden. Außerdem kann man erneut nicht feststellen, wer eine Aufgabe gelöst hat.

1.3.3 SplenDO

SplenDO der Firma Splend Apps wird lediglich im Google Play Store angeboten und bietet wiederkehrende Aufgaben an[3]. Beim Testen wurde festgestellt, dass die wiederkehrenden Aufgaben auch nur in regelmäßigen Abständen angeboten werden. Da keine Funktion gefunden wurde, um andere Nutzer zu einer Liste hinzuzufügen, entfallen die übrigen Punkte.

1.3.4 Warum sind diese Listen nicht für diesen Betrieb geeignet?

Die folgende Tabelle zeigt die vorher vorgestellten Alternativen, welche der vorgegebenen Anforderungen erfüllt wurden und welche nicht. Wurde ein Feld mit ’-’ ausgefüllt, bedeutet dies, dass das Feld nicht sinnvoll ausgefüllt werden kann. Dies war bei SplenDO der Fall, da diese Applikation nicht mehrere Nutzer unterstützt.

Name	Mehrere Nutzer?	wiederkehrend & unregelmäßiger Abstand?	Android & iOS?	Von wem gelöst?	Nutzer entfernen?
Wunderlist	Ja	Nein	Ja	Nein	Ja
Any.do	Ja	Nein	Ja	Nein	Ja
SplenDO	Nein	Nein	Nein	-	-

Damit ein Programm als Alternative zu der geforderten Aufgabenliste genutzt werden kann, müssen alle Kriterien der Tabelle mit „Ja“ markiert worden sein. Um zu sehen, ob eine Applikation alle Kriterien erfüllt, werden die Kriterien von links nach rechts überprüft.
Das erste Kriterium erfordert, dass dieselbe Aufgabenliste von mehreren Nutzern zeitgleich eingesehen und bearbeitet werden kann. Dies wird sowohl von der Wunderlist als auch von Any.do erfüllt. Das zweite Kriterium besagt, dass es unter anderem wiederkehrende Aufgaben geben muss, welche allerdings in unregelmäßigen Abständen aufgerufen werden sollen. Dies erfüllt keine der vorgestellten Applikationen. Somit kann von diesen keine als Alternative genommen werden. Die Implementierung dieser Aufgabenliste ist somit nicht redundant und deswegen sinnvoll.

2 Technologien

Das folgende Kapitel beschreibt die Technologien, die für die Implementierung dieser mobilen Applikation und der Java GUI genutzt werden.
Zunächst wird das Ionic Framework vorgestellt. Dieses wurde für die Implementation der mobilen Applikation genutzt. Danach werden die Java Foundation Classes beschrieben. Diese wurden genutzt, um die grafische Oberfläche der Java Applikation zu erstellen. Zusätzlich wird hier ein Beispiel implementiert. Anschließend wird MySQL, die Technologie mit der das Datenbanksystem umgesetzt wird, vorgestellt. Um die Kommunikation zwischen der Datenbank und der mobilen Applikation zu ermöglichen, ohne sensible Daten innerhalb des Appcodes zu verwenden, wurde sich für eine REST API, welche mit dem Slim Framework umgesetzt wird, entschieden. Dementsprechend wird in diesem Kapitel dieses Framework vorgestellt und in diesem Zusammenhang erläutert, was eine REST API ist.

2.1 Ionic Framework

Das folgende Unterkapitel stellt das Ionic Framework, welches für die Implementierung der mobilen Applikation genutzt wurde, vor. Dazu wird zunächst die Idee des Framworks vorgestellt, ein Beispielprojekt erstellt und das Testen mit Ionic erklärt.
Dazu werden zunächst die Grundlagen über das Ionic Framework vermittelt.

2.1.1 Was ist das Ionic Framework?

Das Ionic Framework ist ein Open Source Projekt, welches es dem Nutzer ermöglicht, Hybrid-Applikationen zu entwickeln[4]. Um das Konzept von Hybrid-Applikationen zu erläutern, muss etwas ausgeholt werden.
Fast jedes mobile Betriebssystem hat eine API mit einer WebView, um Applikationen zu entwickeln. Eine WebView ist ein Browser, der innerhalb einer mobilen Applikation ausgeführt wird. Dies ermöglicht einem, eine Website mithilfe von HTML, CSS und Java Script zu erstellen, die dann innerhalb der entwickelten Applikation ausgeführt wird. Das bedeutet, dass man einmalig Webcode entwickeln muss, der anschließend mithilfe von einem bereitgestellten, plattformspezifischen Code auf den verschiedenen Betriebssystemen innerhalb einer Applikation angezeigt werden kann[5].
Die Schnittstelle zwischen WebView und systemeigenen Code wird beim Ionic Framwork von Apache Cordova zur Verfügung gestellt. Das Ionic Framework selbst baut zusätzlich auf Angular, einem Webapplikationsframework von Google, auf[6]. Dieses Framework stellt eine robuste Struktur für Frontends, mit der UI-Komponenten erstellt und HTML Templates gesetzt werden können, weshalb dieses eine perfekte Grundlage für Ionic bildet[4].
Insgesamt schafft es das Ionic Framework so, diverse UI-Komponenten zur Verfügung zu stellen, die so schnell eingebunden werden können und damit den Entwicklungsprozess vereinfachen und beschleunigen. Zusätzlich wurde von den Ionic-Entwicklern dafür gesorgt, dass die Komponenten wie die plattformspezifischen Originalen aussehen und sich so verhalten[6].

2.1.2 Erstellen einer Ionic Applikation

Es ist sehr einfach, seine erste Ionic Applikation zu erstellen, nachdem das Framework erst einmal eingerichtet wurde. Durch den Befehl

```
$ ionic start Applikationsname
```

wird ein Ionic Projekt mit dem entsprechendem Applikationsnamen erstellt. Beim Erstellen wird man dazu aufgefordert, eines der offiziellen Templates zu wählen[7]. In Abbildung 1 sieht man ein Projekt, welches mit dem „tabs“ Template erstellt wurde. Dieses Template wird auch später bei der Implementation der mobilen Applikation als Grundgerüst dienen.

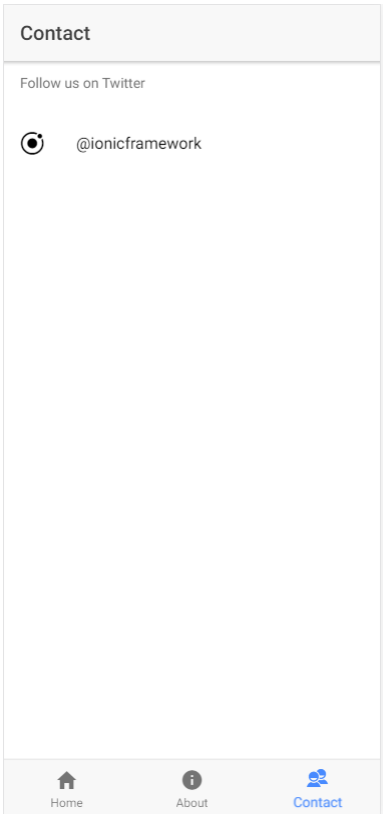

Abbildung 1: Beispiel Ionic Applikation.

2.1.3 Testen mit Ionic

Ionic stellt diverse Möglichkeiten bereit, um seine Applikation auszuführen und zu testen. Zum Beispiel mit dem Befehl:

```
$ ionic serve
```

Hierbei wird die Applikation im Browser ausgeführt. Dabei wird standardmäßig die Ansicht der Android-Plattform ausgewählt. Mit dem Tag „- -platform=platformname“ können auch iOS und Windows dargestellt werden[6]. Außerdem wird diese Anzeige automatisch aktualisiert, falls innerhalb des Ionic Projektes in den HTML, CSS oder Java Script Dateien Änderungen gespeichert werden[8]. In diesem Modus ist es in neuem Code besonders einfach, kleinere Fehler, z.B. Tippfehler, zu überprüfen, da die Applikation in diesem Modus immer auf dem aktuellsten Stand ist, ohne zwischendurch kompilieren zu müssen.
Wird die Applikation so in Google Chrome getestet, kann mithilfe von Chromes Entwicklertools der Ionic Code auch direkt debuggt werden und Chrome ermöglicht es, die Ansicht der Applikation auf verschiedenen Smartphones anzuzeigen[6].
Mit dem Befehl

```
$ ionic cordova emulate android
```

lässt sich die Applikation direkt in einem vorher vorbereiteten, passenden Simulator ausprobieren[8]. Hier bekommt man ein noch besseres Gefühl dafür, wie sich die Applikation auf dem simulierten Gerät benutzen lässt, ohne dass man dieses besitzen muss. Die Plattform iOS kann allerdings nur in Xcode simuliert werden. Xcode ist allerdings lediglich auf macOS verfügbar.
Besitzt man allerdings ein passendes Gerät, kann man mit

```
$ ionic cordova run android
```

seine eigene App auch direkt auf sein Smartphone laden und dort testen[8].

2.2 Java Foundation Classes

Um die grafische Benutzeroberfläche in Java umzusetzen, wurden die Java Foundation Classes, im Folgenden JFC genannt, genutzt. Die JFC sind ein eigenes Modul von Sun, welches dem Programmierer ermöglichen soll möglichst einfach grafische Benutzerflächen zu erstellen. Das Modul setzt sich aus dem Abstract Window Toolkit (AWT) und den Swing-Komponenten zusammen. Das AWT wurde schon 1985 veröffentlicht und beinhaltet einfache Zeichenmethoden, Methoden zur Ereignisbehandlung und diverse, simple GUI-Komponenten. Da anspruchsvollere Oberflächen immer noch mit viel Arbeit verbunden waren, wurde das AWT 1998 noch durch die neuen Swing-Komponenten erweitert[9].

2.2.1 JFC in Eclipse

Um die grafischen Benutzeroberflächen noch einfacher erstellen zu können, wäre ein Werkzeug nützlich, mit dem die Oberflächen einfach aus den verschiedenen Komponenten per Drag & Drop erstellt werden können. Dies wird von sogenannten GUI-Buildern angeboten. Diese setzen das WYSIWYG-Prinzip um, wodurch es möglich ist, schon beim Erstellen der GUI das aktuelle Ergebnis zu betrachten[9]. In der Entwicklungsumgebung Eclipse ist standardmäßig

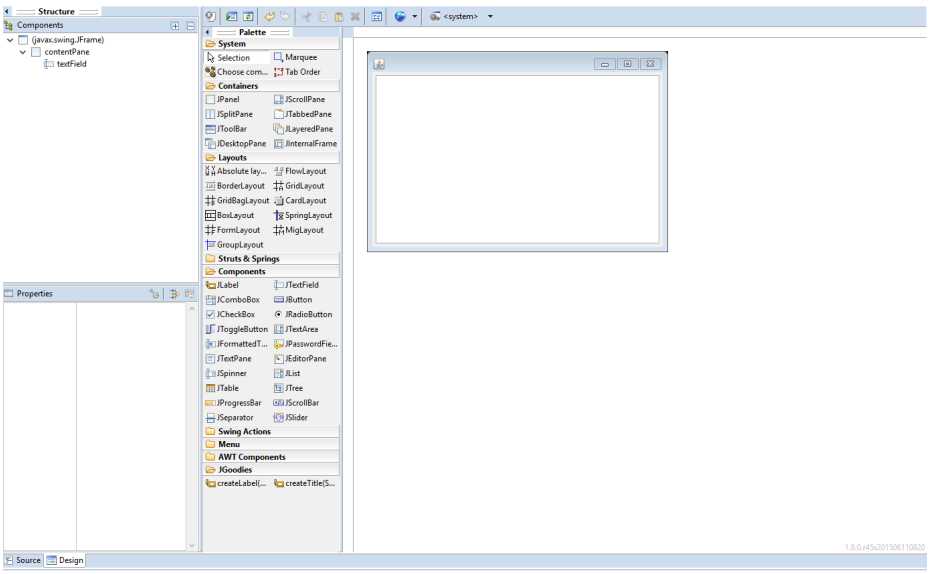

Abbildung 2: Eclipse Window Builder.

kein GUI-Builder enthalten. Dieser kann aber mit dem WindowBuilder-Plugin nachinstalliert werden[9].
Abbildung 2 zeigt ein standardmäßiges Fenster, welches mit dem WindowBuilder beim Erstellen eines JFrames automatisch generiert wurde. Links neben dem Fenster ist das Drag & Drop Menü des WindowBuilders zu erkennen. Das „JLabel“, das „JTextField“ und der „JButton“ sind für die Java Applikation, die in dieser Arbeit implementiert wird, die wichtigsten Komponenten. Deswegen werden diese speziell vorgestellt.

JLabel Das JLabel ist ein einfaches Feld, in dem ein Text oder ein Bild dargestellt werden kann. Das Label kann auf keine Eingaben reagieren[10].

JTextField Ein JTextField ist eine Komponente, in der eine einzelne Zeile Text bearbeitet werden kann. Der Inhalt der Zeile kann auch ausgelesen werde [10].

JButton Ein JButton ist ein einfacher Knopf, welcher gedrückt werden kann. Meistens wird ein JButton zusammen mit einem ActionListener verwendet. So kann durch das Drücken des Knopfes eine beliebige Methode ausgeführt werden[10].

2.3 MySQL

MySQL ist ein relationales Datenbankmanagementsystem, welches von MySQL AB entwickelt wurde. Das System wird als Open Source Software angeboten [11].
Durch ein Datenbankmanagementsystem wird eine Datenbank erst praktikabel, da eine Datenbank alleine nur erlaubt, ein Schema anzulegen, Daten anzulegen und Daten auszulesen. Durch das Managementsystem können Zusammenhänge zwischen verschiedenen Datenschemata berücksichtigt und so die Anzahl an Redundanzen in der Datenbank minimiert werden. Außerdem ist das Managementsystem für den Umgang mit parallelen Aktionen mehrerer

Nutzer zuständig und ermöglicht es, die Rechte einzelner Nutzer für die Erstellung, Löschung und Veränderung von Datenschemata zu definieren[12].

2.4 Slim Framework

Das folgende Unterkapitel behandelt das Slim Framework. Um die Funktion des Slim Frameworks zu verstehen, muss zunächst erläutert werden, was eine REST API ist. Anschließend werden die Vorteile des Frameworks und das Framework an sich vorgestellt.

2.4.1 REST API

REST steht für „REpresentational State Transfer“ und ist ein Architekturstil, der definiert wurde, um einfacher verteilte Systeme zu erstellen und zu planen. Die REST-Architekur baut im Wesentlichen auf die folgenden sieben Prinzipien auf[13]:

- **Client-Server**: REST möchte Client und Server, also Nutzerinterface und Datenhaltung, strikt trennen. Dies ermöglicht zum einen eine einfache Portierbarkeit auf der Clientenseite und steht auf der anderen Seite einer guten Skalierbarkeit des Servers nicht im Weg.
- **Einheitliche Schnittstelle**: Die Interaktionen zwischen den Clients und dem Server basieren auf einer einheitlichen Schnittstelle. Dadurch erhält man eine vereinfachte Architektur und höhere Visibilität von Interaktionen.
- **Layered System**: REST setzt auf mehrschichtige Strukturen. Dies verbessert die Sicherheit, da nur angrenzende Schichten miteinander Kommunizieren müssen. Allerdings führt dies zu einem zusätzlichem Overhead.
- **Caching**: Durch Caching kann eine bessere Erreichbarkeit durch Entlastung des Servers gesichert werden. Dies wird erreicht, indem ein Client die Antworten eines Servers speichert, um diese bei einer erneuten, ähnlichen Frage wiederzuverwenden.
- **Zustandslosigkeit**: Client und Server kommunizieren zustandslos. Dies bedeutet, dass ein Server keine Information eines Clienten bei einer Anfrage speichert und wiederverwendet. Stattdessen muss ein Client bei einer Anfrage alle relevanten Informationen angeben. Das führt dazu, dass die Clienten einen höheren Netzwerkaufwand wegen den größeren Anfragen haben. Allerdings erlaubt es auch dem Server, mehrere Clienten zu bearbeiten, da der Rechenaufwand eines einzelnen Clienten sinkt.
- **Code-on-Demand**:Code-on-Demand ist eine optionale Anforderung, die es dem Client ermöglicht, ausführbaren Code vom Server herunterzuladen.
- **HATEOAS**: HATEOAS steht für „Hypermedia As The Engine Of Application State“, dies lässt sich in etwa in Hypermedia als Motor des Anwendungszustands übersetzen. Das Ganze soll bedeuten, dass in den Antworten des Servers Hyperlinks enthalten sind. So können auch große Mengen an Informationen, auch von verschiedenen Quellen, einfach versendet und verarbeitet werden.

Die eigentliche Idee von REST ist nun, dass anstatt von komplexen Internetdiensten Anfragen einfach mit dem HTTP Protokoll gemacht werden können. Dazu werden lediglich die vier CRUD Operationen von HTTP Anfragen genutzt. Diese sind Erstellen („Create“), Lesen („Read“), Aktualisieren („Update“) und Löschen („Delete“). Dadurch wird REST sehr vielseitig und kann in diversen Programmiersprachen und plattformunabhängig genutzt werden [14].
REST APIs werden mittlerweile in vielen webbasierten Zusammenhängen genutzt. Einige prominente Projekte, die auf REST APIs zurückgreifen, sind unter anderem Twitter, Facebook, YouTube und auch Google[13].

2.4.2 Slim

Slim ist ein PHP Micro Framework, welches einem erlaubt, möglichst schnell und einfach eine REST API zu implementieren [15]. Außerdem ist es eines der schnellsten REST Frameworks und enthält alle Funktionen, die ein solches Framework benötigt[16].

```php
<?php
// Create and configure Slim app
$config = ['settings' => [
    'addContentLengthHeader' => false,
]];
$app = new \Slim\App($config);

// Define app routes
$app->get('/hello/{name}', function ($request, $response, $args) {
    return $response->write("Hello " . $args['name']);
});

// Run app
$app->run();
```

Listing 1: Einfache Slim Applikation [15].

Listing 1 zeigt eine einfache Slim-API, um hieran die Funktionsweise zu verdeutlichen. Zunächst wird die Slim App instantiiert und konfiguriert. Anschließend werden die verschiedenen Routen für die HTTP Anfragen und deren Funktionen definiert. In diesem Fall bekommt man mit einer HTTP GET-Anfrage unter der Adresse „http://www.beispieladresse.de/**hello/Max/**“ eine HTTP Antwort mit „Hello Max“. Zuletzt muss die Applikation noch gestartet werden[15].

3 Implementation

Nachdem in den vorherigen Kapiteln begründet wurde, wieso die Implementation einer mobilen Applikation für den Betrieb sinnvoll ist und die Technologien, die für diese Applikation benutzt werden, erläutert wurden, wird im folgenden Kapitel die mobile Applikation, die Java Applikation und die dazu passende Datenbank mit der REST API implementiert. Hierbei werden die einzelnen Funktionsweisen der Features erklärt und erläutert, wie diese implementiert wurden. Die Reihenfolge der einzelnen Unterkapitel entspricht der chronologischen Reihenfolge, in der die Teile des Features implementiert wurden, um zu verdeutlichen, wie diese aufeinander aufbauen. Der komplette Code und Screenshots der mobilen Applikation und der Java Applikation können unter der Seite
https://github.com/dkopatz/bachelor_dkopatz/tree/master/ eingesehen werden.

3.1 Login

Zunächst wurde die Autorisierung für den Nutzer innerhalb der Applikation, wie in 1.2.4 beschrieben, eingerichtet, da dies auch der erste Teil ist, den ein Nutzer sieht, wenn er die mobile Applikation startet. Dazu wurde zunächst eine entsprechende Tabelle in der Datenbank erstellt und sodann eine passende HTTP-Request für die REST API erstellt. Anschließend wurde mit dem Ionic Framework die Applikation so modifiziert, dass ein Nutzer zunächst auf einer Startseite gültige Logindaten eingeben muss, um zu den restlichen Funktionen zugelassen zu werden. Zuletzt wurde noch innerhalb der Java Applikation eine Möglichkeit geschaffen, um neue Nutzer zu registrieren.

3.1.1 Datenbank

Um die Zugangsdaten von Nutzern in der Datenbank speichern zu können, wurde eine Tabelle „login“ erstellt. Die Tabelle hat zwei Spalten des Typen „text“, um dort einen Nutzernamen und ein Passwort zu speichern.

3.1.2 REST API

In der REST API wurde unter der Route „/api/login/Nutzername/Passwort“ eine HTTP Request eingerichtet, welche eine SQL-Anfrage an die Datenbank sendet. Für die Anfrage wird aus der Route Nutzername und Passwort ausgelesen und sucht in der Tabelle „login“ nach einem Eintrag mit diesem Nutzernamen und diesem Passwort und gibt, wenn diese vorhanden sind, ein JSON-Objekt mit diesem Nutzername und Passwort wieder zurück.

3.1.3 mobile Applikation

Zunächst muss eine neue Seite „login“ eingerichtet werden. Damit diese beim Starten der Applikation angezeigt wird, muss in der „app.component.ts“ die „LoginPage“ als „root-Page“eingerichtet werden. Die Login Seite besteht im Wesentlichen aus drei Teilen: zwei Eingabefeldern für Nutzername und Passwort (wobei die Zeichen im Passwortfeld natürlich unkenntlich gemacht sind) und einem Knopf, mit dem man sich anmelden kann. Die Abblidung 3 zeigt den Aufbau der Methode login(), welche beim Drücken des Knopfes ausgeführt wird, als Programmablaufplan. Zunächst wird der SHA256-Wert des Textes aus

dem Passwortfeld generiert. Dies soll zusätzliche Sicherheit erzeugen, weil das Passwort so nicht als Klartext in der folgenden HTTP-Request versendet wird und damit nicht problemlos von einem möglichen Angreifer abgehört werden kann. Anschließend wird die HTTP-Request an die in Unterkapitel 3.1.2 erstellte Route mit Inhalt des Nutzernamenfeldes und dem generierten Hash gesendet. Nachdem 800 ms gewartet wurde, um auf eine mögliche Antwort der API zu warten, wird zunächst überprüft, ob eine HTTP-Response erhalten wurde. Ist dies nicht der Fall, wird eine Fehlermeldung gesendet, weil dies bedeutet, dass die REST-API nicht erreicht werden kann. Ansonsten wird überprüft, ob die Response ein passendes JSON-Objekt enthält. Wenn ein leeres JSON-Objekt empfangen wurde, bedeutet dies, dass kein Nutzername mit dem angegebenen Passwort gefunden wurde und es wird erneut eine entsprechende Fehlermeldung ausgegeben. Ansonsten wird der Inhalt des JSON-Objektes gespeichert, damit ein Nutzer beim nächsten Start automatisch eingeloggt werden kann und er wird zur nächsten Seite weitergeleitet.

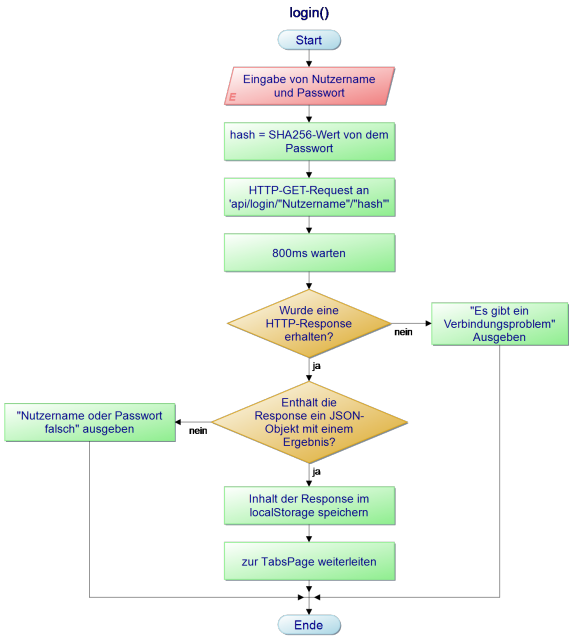

Abbildung 3: Programmablaufplan der Methode login().

3.1.4 Java Applikation

Um mit der Java Applikation neue Nutzer zu erstellen, wurde in dem Fenster aus Kapitel 2.2.1 ein Feld für Nutzernamen und ein Feld für Passwörter erstellt. Durch das Drücken eines Knopfes wird eine SQL-Abfrage ausgeführt, bei der Nutzername und Passwort aus den Feldern in der Tabelle „login“ gespeichert werden.

3.2 Aufgabenliste

Im folgenden Unterkapitel wird die Funktion der Aufgabenliste mit allen Anforderungen aus Kapitel 1.2.1 implementiert. Dafür werden zunächst die nötigen Tabellen in der Datenbank hinzugefügt. Anschließend werden die REST API um die nötigen Requests erweitert und die Aufgabenliste in der mobilen Applikation implementiert. Zuletzt wird die Java Applikation um Elemente und Funktionen zum Hinzufügen von Aufgaben und Aufgabenlisten erweitert.

3.2.1 Datenbank

In der Datenbank wurden zwei Tabellen hinzugefügt. Die Tabelle „aufgaben“ dient dazu, die wiederkehrenden Aufgaben zu speichern. Diese hat zwei Spalten. Eine Spalte „Aufgabe“, um die eigentliche Aufgabe speichern zu können, und eine Spalte „Priorität“, mit der es später möglich sein soll, die Aufgaben in eine übersichtlichere Reihenfolge zu bringen. Außerdem wurde die Tabelle „todoliste“ erstellt. Diese hat zusätzlich zu den zwei Spalten der Tabelle „aufgaben“ noch eine Spalte „Datum“, mit der eine Aufgabe zu einem Spieltag zugeordnet werden kann. Die Spalte „Status“ ist vom Typ „enum“ und erlaubt die Werte „gelöst“ und „ungelöst“. Zuletzt gibt es noch die Spalte „gelost_von“. Hier sollen Nutzernamen eingetragen werden, die eine Aufgabe gelöst haben.

3.2.2 REST API

Die Aufgabenlistenfunktion soll in mobilen Applikation mit zwei Seiten umgesetzt werden. Auf der ersten Seite sollen die verschiedenen Daten der Arbeitstage angezeigt werden. Dazu wurde eine Route „/api/tododat“hinzugefügt. Diese führt eine SQL-Abfrage aus bei der alle Daten aus der Tabelle „todoliste“ ausgegeben werden. Mit dem Zusatz „GROUP BY Datum“ wird verhindert, dass Daten doppelt erscheinen. Gleichzeitig werden so die Daten sortiert.
Auf der zweiten Seite sollen gelöste und ungelöste Aufgaben nacheinander angezeigt werden, der Status von Aufgaben soll aktualisiert werden können und es sollen neue Aufgaben zu der Tabelle hinzugefügt werden können. Dafür wurde unter der Route „/api/todo/Datum“ eine Abfrage ausgeführt, die alle Aufgaben mit dem Status „ungelöst“ und dem passenden Datum geordnet nach der Priorität als JSON-Objekt zurückgibt. Unter der Route „/api/todone/Datum“ wird eine ähnliche Abfrage für Einträge mit dem Status „gelöst“ ausgeführt. Mit dem Unterschied, dass die Aufgaben nicht nach der Priorität geordnet werden und stattdessen noch die Spalte „gelost_von“ mit zurückgegeben wird. Außerdem wurde noch eine HTTP-PUT-Request unter der Route „/api/todo/update/Datum/Aufgabe/Nutzer“ eingerichtet. Mit dieser wird der Status der Aufgabe mit dem passenden Datum zu „gelöst“geändert und in der Spalte „gelost_von“ wird der Nutzer eingetragen. Zuletzt wurde noch eine HTTP-POST-Request unter der Route „/api/todo/add/Datum/Aufgabe“ eingerichtet. Bei dieser wird eine SQL-Abfrage ausgeführt, mit der ein Eintrag mit dem Datum und der Aufgabe aus der Route erstellt wird. Als Status wird „ungelöst“ eingetragen und und die Spalte „gelost_von“ bleibt leer.

3.2.3 mobile Applikation

Die erste Seite dieses Features ist die „TodoPage“. Bei dieser wird im Konstruktor die Request der Route „/api/todo/Datum“ aufgerufen. Das so empfangene JSON-Objekt aus Daten wird in der Applikation als Liste aus Buttons angezeigt. Drückt man auf einen dieser Buttons, wird die Methode „objectSelected(object)“ ausgeführt. Diese ruft die nächste Seite „TododetailPage“ auf und übergibt noch zusätzlich das Datum des gedrückten Knopfes an die Seite. Zusätzlich wurde noch ein Refresher hinzugefügt. Der Refresher ist eine Komponente, welche eine definierte Methode ausführt, wenn der Nutzer auf dem Bildschirm einen Finger von oben ausreichend nach unten bewegt[17]. Hier führt der Refresher eine Methode aus, welche dieselbe HTTP-Request aus dem Konstruktor erneut ausführt, um gegebenenfalls neue Daten anzeigen zu können.
Im Konstruktor der „TododetailPage“ werden zunächst zwei GET-Requests ausgeführt. Aus dem JSON-Objekt, das nach der Request an die Route „/api/todo/Datum“ mit dem Datum, das von der vorherigen Seite übergeben wurde, empfangen wird, werden die Aufgaben ausgelesen. Diese werden in Form einer Liste angezeigt. Die einzelnen Aufgaben lassen sich von links nach rechts bewegen, wodurch ein Knopf offengelegt wird. Wird dieser gedrückt, wird die HTTP-PUT-Request aus dem vorherigen Kapitel ausgeführt mit dem vorherig gedrücktem Datum, der gedrückten Aufgabe und dem zum Login genutzten Nutzernamen als Parameter in der Route. Das JSON-Objekt, das nach der Request an die Route „/api/todone/Datum“ mit demselben Datum, wie bei der vorherigen Request empfangen wurde, enthält Paare aus Aufgaben und die Namen von Nutzern, die diese gelöst haben. Die Paare werden nebeneinander in einer Liste aus Textfeldern unter der andere Liste ausgegeben.
Zwischen den Listen wurde ein Eingabefeld mit einem Knopf hinzugefügt. Durch Drücken wird die HTTP-POST-Request aus dem vorherigen Kapitel mit dem übergebenen Datum und dem Inhalt des Feldes als Aufgabe ausgeführt. Damit die neue Aufgabe auch angezeigt werden kann, werden anschließend noch einmal dieselben Requests wie aus dem Konstruktor ausgeführt.

3.2.4 Java Applikation

Um in der Java Applikation neue wiederkehrende Aufgaben in der Datenbank einzutragen, wurden ein Eingabefeld und ein JSpinner in dem Fenster hinzugefügt. Ein JSpinner ist eine Komponente, in der ein Nutzer Zahlen eingeben kann. Rechts von dem Eingabefeld befinden sich zwei Pfeile mit der die Zahl inkrementiert und dekrementiert werden kann[10]. Durch das Drücken eines Knopfes wird eine SQL-Abfrage ausgeführt, die die beiden Eingaben einliest und mit diesen einen Eintrag in der Tabelle „aufgaben“ erstellt. Um die wiederkehrenden Aufgaben nun in die Tabelle „todoliste“ zu übertragen, wurde noch ein Textfeld für ein Datum und ein Knopf hinzugefügt. Durch Drücken des Knopfes werden die wiederkehrenden Aufgaben zunächst mit der Priorität aus der Tabelle „aufgaben“ ausgelesen und beide werden in separaten Arrays gespeichert. Anschließend wird ein SQL-Statement vorbereitet, bei der alle wiederkehrenden Aufgaben mit dem Datum aus dem Textfeld, der ausgelesenen Priorität und mit dem Status „ungelöst“ eingetragen werden. Zuletzt wird dieses noch an die Datenbank weitergeleitet.

3.3 Bestückungsliste

Im folgenden Unterkapitel wird die Funktion der Bestückungsliste mit allen Anforderungen aus Kapitel 1.2.2 implementiert. Zunächst wurde wieder eine Tabelle in der Datenbank eingerichtet. Anschließend wurde die Java Applikation um nötige Felder und Methoden zum Einlesen der Bestückungslisten erweitert, um das Testen der dazugehörigen HTTP-Requests und der mobilen Applikation zu ermöglichen. Eine Dokumentenvorlage der Bestückungsliste ist in Abbildung 7 zu erkennen. Um zu verdeutlichen, welche HTTP-Requests benötigt werden, wird, bevor diese implementiert werden, das Aussehen des Features in der Applikation beschrieben. Zuletzt werden die eigentlichen Funktionen der Applikation implementiert.

3.3.1 Datenbank

Für die Bestückungsliste wurde eine Tabelle „bestueckungs-items“ erstellt. Aufgrund von Abbildung 7 wurden dann die Spalten „Stand_nr“, „Sorte“ und „Menge“ erstellt. Um die verschiedenen Spieltage unterscheiden zu können, wurde außerdem noch eine Spalte „Datum“ hinzugefügt. Um in der späteren mobilen Applikation eine bessere Übersicht zu schaffen und das Einfügen der Mengen in die Liste einfacher automatisieren zu können, wurde noch eine Spalte „Position“ hinzugefügt. In dieser Spalte wird die Stelle festgehalten, an welcher sich die zugehörige Sorte in der Excelvorlage befunden hat. Diese Möglichkeit erfordert zwar zusätzlichen Speicherplatz, ermöglicht es aber auch innerhalb kürzester Zeit weitere Produkte in der Vorlage hinzuzufügen, ohne dass Probleme entstehen könnten.

3.3.2 Java Applikation

Zunächst wurden in der Java Applikation zwei Textfelder hinzugefügt. In das eine soll das Datum der Liste eingetragen werden und in das andere Testfeld kann mithilfe eines danebenliegenden Knopfes ein Pfad zu einem Exceldokument herausgesucht werden. Es wurde außerdem ein Knopf hinzugefügt, welcher eine Bestückungsliste unter dem angegebenen Pfad einliest und mit dem angegebenen Datum zu der Datenbank hinzufügt. Abbildung 6 zeigt ein Programmablaufsplan, welcher die Methode zu diesem Knopf beschreibt. Zunächst werden aus der ersten Zeile die verschiedenen Produkte ausgelesen und gespeichert, um aus den folgenden Zeilen die Werte dem richtigen Produkt zuordnen zu können. In allen folgenden Zeilen wird zunächst überprüft, ob die erste Spalte einen String mit vier oder weniger Zeichen enthält, da dies das Format der Standnummern ist. Wenn eine Standnummer erkannt wurde, wird die Standnummer in einem Array gespeichert und aus den anderen Spalten in dieser Zeile werden die Mengen ausgelesen. Alle so eingelesenen Mengen werden in einer zweidimensionalem Matrix gespeichert. Alle Mengen eines Standes werden in einer Zeile gespeichert, damit diese zugeordnet werden können. Enthält eine Zelle, in der eine Menge enthalten sein sollte, keinen Wert, wird in der Matrix stattdessen der Wert Null gespeichert. Nachdem so alle Zeilen des Exceldokumentes gelesen wurden, wird eine SQL-Abfrage erstellt, in der sämtliche, eingelesen Werte gleichzeitig in die Datenbank eingefügt werden.

3.3.3 REST API

Das Feature soll in der mobile Applikation aus drei Seiten bestehen. Zunächst sollen alle Daten angezeigt werden, für die eine Bestückungsliste hochgeladen wurde. Dazu wurde eine Request unter der Route „api/bestdat“ hinzugefügt. Diese führt eine Abfrage aus bei der alle Daten aus der Tabelle „bestueckungs_items“ ausgegeben werden. Dies wurde Analog zu der Route „api/tododat“ aus Kapitel 3.2.2 umgesetzt. Anschließend sollen zu einem Datum alle Stände angezeigt werden. Dafür wurde unter der Route „/api/best/Datum“ hinzugefügt. Hier wurde wieder der Zusatz „GROUP BY“ benutzt, um Dopplungen zu vermeiden. Für die dritte Seite sollen dann noch die verschiedenen Produkte mit den dazugehörigen Mengen angezeigt werden. Dafür wurde eine Route unter „/api/besten/Datum/Stand“ eingerichtet. Da laut Kapitel 1.2.2 auch die Mengen von mehreren Ständen addiert angezeigt werden sollen, wird zunächst eine Substitution ausgeführt, die die Himmelsrichtungen in einen Ausdruck ändert, der auf alle entsprechenden Stände zutrifft, zum Beispiel „Norden“ in „N%“. Bei der SQL-Anfrage werden alle Einträge mit passendem Datum und Stand ausgegeben. Diese werden nach ihrer Position gruppiert. Als JSON-Objekt zurückgegeben werden dann die verschiedenen Produkte mit ihren Mengen. Die Funktion „sum()“ wurde genutzt, um die Mengen einer Position zusammen zu addieren. Zuletzt wurde noch eine HTTP-PUT-Request unter der Route „/api/best/update/Datum/Stand/Sorte/Menge“ hinzugefügt, um das Aktualisieren von Mengen aus Kapitel 1.2.2 zu ermöglichen.

3.3.4 mobile Applikation

Die HTTP-Request der Route „/api/best/Datum“ wird in der „HomePage“ im Konstruktor aufgerufen. Die Daten aus dem so empfangenen JSON-Objekt werden analog zu dem aus der „TodoPage“ aus Kapitel 3.2.3 angezeigt. Durch Drücken eines Datum wird allerdings die „BestdetailPage“ aufgerufen. Außerdem enthält die „HomePage“ auch einen Refresher, der ebenfalls analog zu dem von der „TodoPage“ implementiert wurde.
Bei der „BestdetailPage“ wird im Konstruktor zunächst das Datum aus dem übergebenen Objekt ausgelesen und eine HTTP-Request an die Route „/api/best/Datum“ mit dem ausgelesenen Datum gesendet. Die Stände aus der Antwort werden erneut wieder als Liste aus Knöpfen angezeigt. Durch Drücken eines dieser Knöpfe werden zum einen das Datum und der gedrückte Stand übergeben und zum anderen die „StandPage“aufgerufen. Alternativ gibt es auch noch einen FabButton. Der Floating Action Button ist ein Knopf der über den normalen Inhalten an einer festen Position sitzt und weitere Optionen aufrufen kann[17]. Hier wird dieser genutzt, um die Mengen von mehreren Ständen aufzuaddieren. Beim Drücken des FAB's erscheint ein Menü, in dem man noch die verschiedenen Himmelsrichtungen auswählen kann, um dann die „StandPage“mit der entsprechenden Himmelsrichtung und dem Datum aufzurufen.
Bei der „StandPage“ wird aus dem übergebenen Objekt Datum und Stand ausgelesen und mit diesen die HTTP-Request an die Route „/api/besten/Datum/Stand“ ausgeführt. Das empfangene JSON-Objekt enthält nun mehrere Paare aus Sorten und Mengen, welche in einer Liste angezeigt. Ein Paar wird immer in einer Zeile angezeigt. Sorten als Text, wie es bei den vorherigen Listen auch der Fall war, und die Mengen werden in einer Textbox angezeigt, welche bearbeitet werden kann. Über der Liste befindet sich ein Knopf, welcher einen Bestätigungsdialog aufruft, um ein versehentliches Ausführen der Methode zum

Aktualisieren zu verhindern. Die Aktualisierung wird erst nach einer Bestätigung ausgeführt. Die Methode führt die HTTP-PUT-Request aus Kapitel 3.3.3 für jedes angezeigte Sorten-Mengen Paar aus und ermöglicht so die Anforderung aus Kapitel 1.2.2, Bestände aktualisieren zu können. Anschließend wird die HTTP-Request der Route „/api/besten/Datum/Stand“ erneut ausgeführt, um die neuen Mengen auch anzuzeigen.

3.4 Inventurlisten

Im folgenden Unterkapitel wird die Funktion der Inventurlisten mit allen Anforderungen aus Kapitel 1.2 implementiert. Es gibt zwei verschiedene Typen von Inventurlisten. Abbildung 8 zeigt die Excelvorlage für die Lebensmittelinventur und Abbildung 9 die Vorlage für die Getränkeinventur. Da die Vorlage für die Getränkeinventur sehr ähnlich zu der Vorlage der Bestückungsliste aus Abbildung 7 ist, kann die Funktion der Inventurliste für Getränke fast komplett analog zu der Bestückungsliste implementiert werden. Deshalb wird auf weitere Erläuterungen zur Implementation der Getränkeinventur an dieser Stelle verzichtet. Die Lebensmittelinventur weist auch einige Parallelen auf, auf diese wird dann hingewiesen.
Für die Funktion der Getränkeinventur wird zunächst die Tabelle in der Datenbank angelegt. Als Nächstes wird dann das Einlesen und Beschreiben der Inventurlisten in der Java Applikation implementiert. Beim Implementieren der nötigen HTTP-Requests wird wieder ein grober Aufbau der Applikation erläutert, um den Nutzen deutlich zu machen. Zuletzt wird das Feature in der mobilen Applikation implementiert.

3.4.1 Datenbank

Die Lebensmittelinventur wird in einer Tabelle „inventurlisten_items“ realisiert. Zum einen wurde eine Spalte „Datum“ hinzugefügt, um die verschiedenen Spieltage unterscheiden zu können. Aus dem Exceldokument von Abbildung 8 wurden die Spalten „Kategorie“, „Produkt“, „Anzahl“, „Einheit“ und „Bemerkung“ entnommen und ebenfalls zu der Tabelle hinzugefügt.

3.4.2 Java Applikation

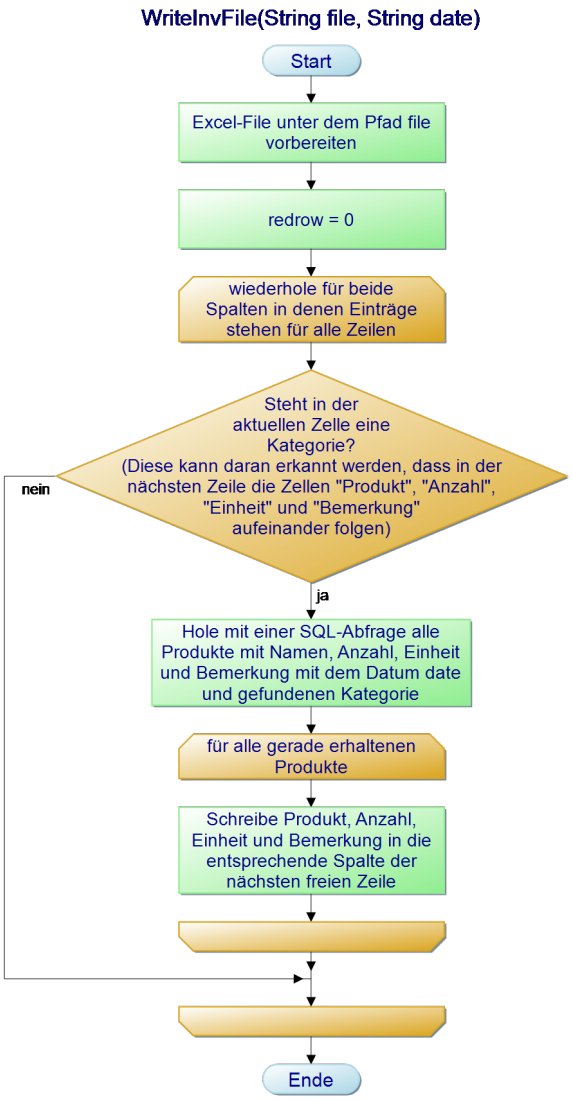

Abbildung 4: Programmablaufplan der Methode WriteInvFile(String file, String date).

In der Java Applikation wurden für die Getränkeinventur zwei Eingabefelder hinzugefügt. Ein Feld ist zum Eingeben eines Datums gedacht und in das andere Feld soll ein Dateipfad zu einer Inventur, die eingelesen werden soll, oder zu einer Vorlage, die beschrieben werden soll, angegeben werden. Durch Drücken des Knopfes „Inventur abschicken“ wird die Methode ReadInvFile ausgeführt, diese ist in Abbildung 4 in Form eines Programmablaufplans dargestellt, der Ablaufplan wird folgend noch erläutert. Zunächst wird in der Exceldatei unter dem Pfad „file“ eine Zeile gesucht, in der eine Kategorie steht. Eine Kategorie wird erkannt, wenn unter der Zeile der Kategorie eine Zeile steht, die ein bestimmtes Format hat. Die so gefundene Kategorie muss gespeichert werden. Solange in den nachfolgenden

Zeilen ein Produktname steht, werden zu diesem Produkt Kategorie, Name, Anzahl, Einheit und die Bemerkung in einer Matrix gespeichert. Dies wird für das ganze Dokument und alle Kategorien wiederholt. Anschließend werden alle Produkte aus der Matrix ausgelesen und mit einer SQL-Abfrage zu der Datenbank hinzugefügt, dabei ist der Inhalt der Spalte „Datum“ bei allen Einträgen das Datum, welches im Methodenkopf übergeben wurde.
Beim Drücken des anderen Knopfes wird die Methode WriteInvFile ausgeführt, diese wird durch den Programmablaufplan in Abbildung 4 beschrieben. Bei der Methode werden erneut wieder alle Kategorien in der Exceldatei unter dem Pfad „file“ nacheinander gesucht. Wenn eine Kategorie erkannt wurde, werden mit einer SQL-Abfrage alle Produkte dieser Kategorie und dem Datum „date“ aus der Datenbank herausgesucht. Jeder Eintrag wird nun nacheinander in die Zeilen unter der Kategorie eingetragen.

3.4.3 REST API

Die Lebensmittelinventur besteht wieder aus drei Seiten. Auf der ersten Seite sollen die verschiedenen Inventurdaten angezeigt werden. Dazu wurde eine HTTP-Request unter der Route „api/invdat“ analog zu der Route „api/tododat“ aus Kapitel 3.2.2 für die Tabelle „inventurlisten_items“ eingerichtet. Auf der zweiten Seite sollen zu einem Datum die Kategorien angezeigt werden. Dazu wurde eine Route „/api/inv/Datum“ eingerichtet, die eine entsprechende SQL-Abfrage ausführt und die Kategorien als JSON-Objekt zurückgibt. Die Ergebnisse wurden noch nach der Kategorie gruppiert, um Dopplungen zu vermeiden. Auf der dritten Seite sollen dann die konkreten Einträge der Kategorien dargestellt und aktualisiert werden können. Außerdem soll es möglich sein, neue Produkte hinzuzufügen. Um die Einträge aus der Datenbank auslesen zu können, wurde eine HTTP-GET-Request unter der Route „/api/inv/Datum/Kategorie“ eingerichtet. Bei der SQL-Abfrage werden „Produkt“, „Anzahl“, „Einheit“ und „Bemerkung“ der Einträge als JSON-Objekt zurückgegeben, die das Datum und die Kategorie aus der Route haben. Zum Aktualisieren wurde eine HTTP-PUT-Request unter der Route „/api/inv/update/Datum/Produktnamen/Anzahl/-Bemerkung“ hinzugefügt. Bei dieser Request wird der Eintrag mit dem Datum und dem Produktnamen aus der Route aktualisiert, indem bei diesem Eintrag die Spalten „Anzahl“ und „Bemerkung“ auf die entsprechenden Werte aus der Route gesetzt werden. Um Produkte hinzufügen zu können, wurde eine HTTP-POST-Request eingerichtet. Diese erstellt einen neuen Eintrag in der Tabelle „inventurlisten_items“ aus den Argumenten der Route „/api/inv/add/Datum/Kategorie/Produkt/Anzahl/Einheit/Bemerkung“

3.4.4 mobile Applikation

Für die Inventurlisten wurde zunächst die „InvWahlPage“ erstellt. Auf dieser werden zwei Knöpfe angezeigt. Durch Drücken des einen Knopfes wird man zu der „GinvPage“ weitergeleitet. Auf dieser und den folgenden Seiten wurde die Getränkeinventur umgesetzt, diese kann wie bereits am Anfang dieses Unterkapitels erwähnt komplett analog zu der Bestückungsliste implementiert werden. Wird der andere Knopf gedrückt, wird man zu der „InventurPage“ weitergeleitet. Diese Seite wurde analog zu der „HomePage“ aus Kapitel 3.3.4 implementiert. Die beiden Seiten unterscheiden sich nur darin, dass bei der „InventurPage“ die Route „api/tododat“ verwendet wird und dass die Seite „InvKategoriePage“aufgerufen wird.
Im Konstruktor dieser Seite wird die HTTP-Request an die Route „/api/inv/Datum“ mit

dem Datum von der vorherigen Seite aufgerufen. Die Kategorien aus dem empfangenen JSON-Objekt werden als Liste aus Knöpfen in der Applikation dargestellt. Wird einer der Knöpfe gedrückt, wird die „InvItemsPage“aufgerufen und dabei werden das Datum und die gedrückte Kategorie übergeben.
Bei dieser Seite wird eine HTTP-GET-Request an die Route „/api/inv/Datum/Kategorie“ im Konstruktor abgeschickt. Der Inhalt des JSON-Objektes wird in einer Liste angezeigt. Dabei werden alle Attribute eines Eintrages aus der Datenbank in einer Zeile angezeigt. Die Spalten „Produkt“ und „Einheit“ werden als einfacher Text angezeigt. Die Spalten „Anzahl“ und „Bemerkung“ werden stattdessen in einem Textfeld angezeigt, welches bearbeitet werden kann. Außerdem wurde ein FabButton in der rechten Ecke hinzugefügt. Durch Drücken des FAB's wird ein Menü mit den Optionen „Wiederherstellen“, „Inventur abschicken“, „Artikel hinzufügen“ und „Cancel“ aufgerufen. Durch das Drücken von „Cancel“ wird lediglich das Menü wieder geschlossen, dies geschieht auch zusätzlich bei den anderen Menüpunkten. Wird „Wiederherstellen “ gedrückt, wird die HTTP-Request aus dem Konstruktor dieser Seite erneut ausgeführt, dabei gehen Daten, die in den Textfeldern eingegeben wurden, verloren. Wird der Menüpunkt „Inventur abschicken“ gewählt, wird für jede Zeile die HTTP-PUT-Request unter der Route „/api/inv/update/Datum/Produktnamen/Anzahl/Bemerkung“ ausgeführt. Anschließend wird noch die HTTP-Request aus dem Konstruktor ausgeführt, damit mögliche Änderungen auch angezeigt werden. Bei der Option „Artikel hinzufügen“ wird nach dem Drücken ein Dialog geöffnet, bei dem man vier Textfelder ausfüllen kann. Wird anschließend der Knopf „Speichern“ in diesem Dialog gedrückt, wird eine HTTP-POST-Request an die Route „/api/inv/add/Datum/Kategorie/Produkt/Anzahl/Einheit/Bemerkung“ gesendet. Die Parameter Datum und Kategorie werden aus den übergebenen Daten des vorherigen Fensters gesetzt. Die übrigen Parameter werden mit den Inhalten der Textfelder aus dem Dialog beschrieben. Anschließend wird wieder die GET-Request aus dem Konstruktor ausgeführt, um den neuen Eintrag anzuzeigen.

4 Auswertung und Evaluation

Nachdem im vorherigen Kapitel die mobile Applikation implementiert wurde, stellt sich die Frage, ob die Applikation den Anforderungen gerecht wird. Dieser Aspekt wird unter anderem im folgenden Kapitel beleuchtet. Außerdem konnte die Applikation nach der Implementierung in dem Catering-Betrieb getestet werden. Anschließend wurde eine Evaluation mit den Mitarbeitern, die die Applikation getestet haben durchgeführt. Diese wird anschließend ausgewertet. Zuletzt werden noch gegebenenfalls vorhandene Fehler besprochen und sinnvolle Erweiterungen für die Applikation erläutert.

4.1 Anforderungen erfüllt?

synchronisierte Listen
Bei allen Features wurde im Kapitel 1.2 gefordert, dass die verschiedenen Listen synchronisiert für alle Nutzer zur Verfügung stehen. Diese Anforderung wurde erfüllt, da sämtliche Inhalte der Applikation über HTTP-Requests aus der Datenbank eingelesen werden und alle Änderungen über HTTP-POST und HTTP-PUT-Reuqests in der Datenbank festgehalten werden. Somit haben alle Nutzer Zugang zu denselben Informationen aus der Datenbank und damit ist die Anforderung erfüllt.

Vertraulichkeit der Daten
In Kapitel 1.2.4 wurde gefordert, dass auf die Inhalte der Applikation nur von autorisierten Personen zugegriffen werden soll. Dafür wurde in Kapitel 3.1.3 eine Seite eingerichtet, die sich bei jedem Start öffnet und den Nutzer nur zu den weiteren Inhalten weiterleitet, wenn dieser einen gültigen Nutzernamen mit dem passenden Passwort eingibt oder hinterlegt hat.

Import der Exceldokumente
In Kapitel 1.2 wurde bei allen Listen gefordert, dass die Inhalte aus den bereits vorhandenen Exceldokumenten, die in Abbildung 7, 8 und 9 dargestellt sind, in die Datenbank eingelesen werden können. Dies wurde für die Bestückungsliste in Kapitel 3.3.2 und für die Inventurlisten in Kapitel 3.4.2 innerhalb der Java Applikation implementiert.

4.1.1 Aufgabenliste

wiederkehrende Aufgaben
In Kapitel 1.2.1 wurde gefordert, dass für die Aufgabenliste wiederkehrende Aufgaben gespeichert werden können. Die wiederkehrenden Aufgaben sollen auf jeder Aufgabenliste erscheinen und die Listen müssen in beliebigen Abständen erstellt werden können. In Kapitel 3.2.4 wurde implementiert, dass wiederkehrende Aufgaben in der Datenbank gespeichert werden und dass die wiederkehrenden Aufgaben bei dem Erstellen einer Aufgabenliste automatisch zu dieser hinzugefügt werden.

Hinweis wer eine Aufgabe gelöst hat
In Kapitel 3.2.3 wurde implementiert, dass Nutzer die Aufgaben auf einer Aufgabenliste als gelöst markieren können. Bei der dazugehörigen HTTP-Request wird zusätzlich der Nutzername mitgesendet. Auf derselben Seite werden außerdem alle gelösten Aufgaben mit den dazugehörigen Nutzernamen angezeigt. Somit gilt diese Anforderung aus Kapitel 1.2.1 ebenfalls als erfüllt.

einmalige Aufgaben in der Applikation hinzufügen
Zwischen den gelösten und ungelösten Aufgaben auf derselben Seite, wurde in Kapitel 3.2.3 außerdem implementiert, dass nach einem Knopfdruck eine Textbox ausgelesen wird und eine Aufgabe mit dem Inhalt der Textbox zur aktuellen Aufgabenliste hinzugefügt wird. Da die Aufgabe nur mit dem passenden Datum in der Tabelle „todoliste“ und nicht in der Tabelle „aufgaben“ eingefügt wird, wird durch diese Funktion eine weitere Anforderung aus Kapitel 1.2.1 erfüllt.

Nutzer entfernen
Die letzte Anforderung aus Kapitel 1.2.1, Nutzer wieder entfernen zu können, wurde innerhalb der Aufgabenliste nicht direkt umgesetzt. Stattdessen können Nutzer dadurch entfernt werden, dass der Eintrag des Nutzers, der entfernt werden soll, aus der Tabelle „login“ ausgetragen wird. Somit kann sich der entsprechende Nutzer nicht mehr beim Starten der Applikation autorisieren und erhält keinen Zugang zu der Aufgabenliste. Dies ist ausreichend, um diese Anforderung zu erfüllen.

4.1.2 Bestückungsliste

Bestückungsliste aktualisieren
Um die Anforderung aus Kapitel 1.2.2, die Mengen einer Bestückungsliste aktualisieren zu können, zu erfüllen, wurde in Kapitel 3.3.4 ein Knopf hinzugefügt, der, wenn er gedrückt wird, eine Funktion ausführt, die die Mengen aus der Bestückungsliste einliest und an der Datenbank sendet. Damit ist die Anforderung erfüllt worden.

Mengen von Standgruppen automatisch aufaddieren
Diese Anforderung wurde dadurch erfüllt, dass man auf der „BestdetailPage“, welche in Kapitel 3.3.4 implementiert wurde, mithilfe des FABs die Standgruppen auswählen kann und auf der folgenden Seite die Summen der Mengen der ausgewählten Stände angezeigt wird.

4.1.3 Inventurliste

Inventurlisten erstellen
Die Funktion Inventurlisten zu erstellen, wurde in den Kapiteln 3.4.2 und 3.4.4 implementiert. Dazu wurde in der Java-Applikation in Kapitel 3.4.2 die Funktion implementiert, mit der eine Inventurliste eingelesen wird. Somit kann auch eine leere Vorlage eingelesen werden, die nur noch ausgefüllt werden muss. Weiter wurde in dem Kapitel 3.4.4 eine Funktion in der mobilen Applikation implementiert, mit der die Menge und die Bemerkung eines Eintrages in der Datenbank bearbeitet werden kann. Dadurch wurde diese Anforderung ebenfalls erfüllt.

neue Produkte Hinzufügen
Die Anforderung aus Kapitel 1.2.3, in den Lebensmittelinventuren Produkte hinzuzufügen, wurde umgesetzt, indem Nutzer die verschiedenen Eigenschaften eines Artikels in mehreren Textboxen angeben und die Inhalte dann per Knopfdruck absenden.

Exceldokument beschreiben
Zuletzt wurde in Kapitel 1.2.3 noch gefordert, dass die Daten der Inventurlisten aus der Datenbank ausgelesen werden sollen, um diese anschließend in die Excelvorlage zu übertragen. Dazu wurde in Kapitel 3.4.2 eine Methode implementiert, die die genannte Anforderung per Knopfdruck durchführt.

4.2 Evaluation

Nachdem die Applikation vollständig implementiert wurde, konnte die Applikation im Arbeitsalltag des Caterings des VFL Osnabrück getestet werden. Mit den Mitarbeitern, die die Applikation benutzt haben, wurde anschließend eine Evaluation durchgeführt. Die komplette Evaluation kann unter
https://github.com/dkopatz/bachelor_dkopatz/tree/master/Evaluation eingesehen werden. Im Folgenden werden die Ergebnisse der Evaluation vorgestellt.

4.2.1 Auswertung der Fragebögen

An der Evaluation haben sieben Personen teilgenommen. Zunächst sollten die Befragten die Applikation allgemein, die Bedienbarkeit und zuletzt die einzelnen Features jeweils auf einer Skala von eins bis fünf bewerten, wobei eins die negativste und fünf die positivste Bewertung darstellt. Bei den verschiedenen Features konnte außerdem noch angegeben werden, dass das Feature nicht benutzt wurde, da Personen ohne Umgang mit einem Feature zu diesem keine sinnvolle Bewertung machen können. Anschließend wurde den Befragten die Möglichkeit gegeben, persönliches Feedback zu den Features oder der Applikation im Allgemeinen zur Verfügung zu stellen.
Die mobile Applikation wurde im Durchschnitt mit 4,429 Punkten bewertet. Damit kann man sagen, dass die Applikation insgesamt positiv aufgenommen wurde und von keinem Ablehnung erfahren hat.
Die Bedienbarkeit wurde mit einem Durchschnittswert von 4,428 ähnlich gut bewertet. Allerdings wurden bei der letzten Frage ausschließlich gute und sehr gute Bewertungen und bei Frage zu der Bedienbarkeit wurde eine neutrale Stimme abgegeben. Auf diese Stimme wird später noch einmal eingegangen.
Die Aufgabenliste wurde im Durchschnitt mit 4,714 Punkten bewertet und wurde damit sogar sehr positiv aufgenommen. Dies spiegelt sich auch darin wieder, dass bei dem zusätzlichen Feedback keine Kritik genannt wurde, sondern lediglich ein weiteres Feature vorgeschlagen wurde. Es wurde vorgeschlagen, statt einer einzigen wiederkehrenden Liste, mehrere wiederkehrende Listen zu erlauben.
Die Funktion der Getränkeinventur wurde nur von einer Person bewertet und zwar mit einem gut, da diese auch nur von einer Person durchgeführt wird. Als zusätzliches Feedback wurde genannt, dass das Eintragen der verschiedenen Mengen zu lange dauere. Als Lösungsvorschlag

wurde zusätzlich noch angegeben, dass automatisch von einem Textfeld zu einem anderen weitergeleitet werden sollte und dieses vor der Eingabe leer sein sollte, anstatt dass man den vorherigen Wert löschen muss.
Die Funktion der Lebensmittelinventur wurde von zwei Personen bewertet und zwar von beiden mit einem gut. Als zusätzliches Feedback wurde ein ähnliches Problem wie bei der Getränkeinventur geschildert. Außerdem wurde auf einen Fehler hingewiesen, wodurch beim Hinzufügen von neuen Artikeln vorherige Inventurdaten verloren gehen, wenn diese nicht abgeschickt wurden. Dieser Fehler wurde überprüft und konnte reproduziert werden. Im nächsten Kapitel wird auf diesen Fehler noch näher eingegangen.
Die Bestückungsliste wurde wieder von allen Teilnehmern bewertet und erhielt eine sehr gute Rückmeldung mit im Durchschnitt 4,714 Punkten. Zu dieser Liste gab es außerdem kein zusätzliches Feedback, deswegen wird insgesamt davon ausgegangen, dass dieses Feature sehr zufriedenstellend angenommen wurde.
Die neutrale Bewertung in Bezug auf die Bedienbarkeit lässt sich am einfachsten auf das Feedback bei den Inventurlisten zurückführen, da hier offenbar Unannehmlichkeiten bei der Eingabe der Mengen aufgetreten sind. Insgesamt ist das Feedback in Bezug auf die mobile Applikation gut ausgefallen. Deshalb ist ein weiterer Einsatz an einem Spieltag vorstellbar und eine Umsetzung der Vorschläge wäre damit denkbar und sinnvoll. Deswegen werden diese Vorschläge im nachfolgenden Kapitel noch einmal aufgegriffen.

4.3 Weitere Arbeiten

Obwohl die Implementierung alle gestellten Anforderungen erfüllt hat, kann die Applikation, wie schon aus dem vorherigen Kapitel hervorgeht, dennoch ausgebessert werden, etwa um Fehler zu eliminieren oder um die Benutzerfreundlichkeit zu verbessern. Im Folgenden werden verschiedene Aspekte besprochen, an denen weiter gearbeitet werden kann. Hierbei beschäftigt sich das erste Unterkapitel mit noch vorhandenen Fehlern und das zweite Unterkapitel mit zusätzlichen Funktionen, die noch ergänzt werden können.

4.3.1 Noch vorhandene Fehler

Auch wenn die einzelnen Funktionen der Applikation einsatzfähig sind und weitestgehend fehlerfrei funktionieren, sind trotzdem einige kleine Fehler vorhanden. Diese werden im Folgenden zusammen mit einem möglichen Lösungsansatz vorgestellt.

Verlieren von nicht gespeicherten Inventurdaten beim Hinzufügen eines neuen Artikels
In Kapitel 4.2.1 wurde ausgewertet, dass bei der Inventurliste Einträge, die in der Applikation geändert wurden, verloren gehen, wenn ein neuer Artikel hinzugefügt wird, bevor die Inventureinträge abgeschickt wurden. Dieser Fehler kann behoben werden, indem die Methode, die Einträge abzuschicken, automatisch durch das Hinzufügen eines neuen Artikels ausgeführt wird.

Fehler beim Auslesen der Lebensmittel-Inventurdaten aus der Datenbank bei zu vielen neuen Artikeln
Da in der Inventurliste beliebig viele neue Artikel hinzugefügt werden können, kann der Fall eintreten, dass so viele Artikel hinzugefügt wurden, dass auf der Excelvorlage zwischen zwei Kategorien nicht ausreichend viele, freie Zeilen vorhanden sind, um alle neuen Artikel eintragen zu können. Um dieses Problem zu lösen, wäre es denkbar, alle neuen Produkte einfach einer eigenen Kategorie zuzuordnen, welche am Ende der Tabelle angehängt wird. Übersichtlicher wäre es aber voraussichtlich, wenn man die Methode, mit der die Excelvorlage beschrieben wird, so ändern würde, dass anstatt der Vorlage eine leere Exceldatei beschrieben wird und die Abstände der verschiedenen Kategorien somit dynamisch sind.

4.3.2 Mögliche Erweiterungen

Im Folgenden werden noch weitere Ergänzungen vorgestellt, um die das System erweitert werden kann. Die Erweiterungen dienen hauptsächlich dazu, die Benutzerfreundlichkeit zu verbessern.

Detaillierte Fehlermeldungen in der Java Applikation
Zum Zeitpunkt der Veröffentlichung dieser Arbeit werden in der Java Applikation die verschiedenen Fehlermeldungen lediglich ausgeworfen. Besser wäre es aber, dass sich, wenn ein Fehler auftritt, stattdessen in dem Fenster der Applikation ein Pop-up öffnet. In dem Pop-up sollte dann eine genauere Nachricht, weshalb der Fehler aufgetreten ist, enthalten sein. Zum Beispiel „Die Datei ist bereits geöffnet“ oder „Fehler beim Einlesen der Datei“.

Einfachere Eingabe bei den Inventur und Bestückungslisten in der mobilen Applikation
Aus den Rückmeldungen der Mitarbeiter aus dem Kapitel 4.2.1 hat sich ergeben, dass die Eingabe von Mengen innerhalb der mobilen Applikation etwas unhandlich gestaltet ist. Deshalb sollten die Textfelder, die bearbeitet werden sollen, sich vor der neuen Eingabe automatisch leeren. Damit könnte ein Arbeitsschritt gespart werden.
Außerdem wurde erwähnt, dass es wünschenswert wäre, wenn nach einer Eingabe der Nutzer automatisch zu dem nächsten Textfeld für die nächste Eingabe weitergeleitet würde. Dies würde die Eingabe bei mehreren Produkten flüssiger machen.

verschiedene wiederkehrende Aufgabenlisten Eine weitere Rückmeldung eines Mitarbeiters aus Kapitel 4.2.1 hat ergeben, dass es verschiedene Listen mit wiederkehrenden Aufgaben geben sollte, weil zum Beispiel im Winter andere Aufgaben erledigt werden müssen, als im Sommer. Dies könnte man umsetzen, indem man zu der Tabelle „aufgaben“ eine Spalte hinzufügt, in der definiert ist, zu welcher Vorlage diese Aufgabe gehört.

Möglichkeit einzelne, wiederkehrende Aufgaben nach Bedarf aussetzen zu können
Zusätzlich zu der vorherigen Erweiterung könnte man die Aufgabenlisten insgesamt individualisierbarer gestalten. Beim Erstellen einer Aufgabenliste könnte sich ein Dialog öffnen, bei welchem man zunächst eine der Vorlagen auswählt. Anschließend könnten alle Aufgaben mit einem Markierungsfeld angezeigt werden. Dabei sind die Aufgaben der ausgewählten

Vorlage bereits markiert. Bevor die Aufgabenliste nun erstellt wird, könnten noch zusätzliche Aufgaben außerhalb der Vorlage markiert werden oder es könnten bei bereits markierten Aufgaben die Markierung entfernt werden.

5 Fazit

5.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde eine mobile Applikation mit allen dazu nötigen Komponenten implementiert und innerhalb eines Betriebes getestet. Die mobile Applikation erfüllt alle gewünschten Anforderungen: Die Applikation ist für Android und iOS verfügbar, in der App können Aufgabenlisten angezeigt werden, zu der mithilfe einer Java Applikation wiederkehrende Aufgaben hinzugefügt werden können, die verschiedenen Inventur- und Bestückungslisten können innerhalb der Applikation angezeigt und bearbeitet werden, die Inventurlisten können wieder aus der Datenbank ausgelesen werden, auf die Inhalte innerhalb der Applikation kann nur nach einem erfolgreichen Login zugegriffen werden und die verschiedenen Exceldokumente können in die Datenbank eingelesen und damit in der Applikation angezeigt werden. Die mobile Applikation konnte erfolgreich im Arbeitsalltag des Catering-Betriebes getestet werden und anschließend konnte konstruktives Feedback eingeholt werden und deshalb kleinere Schwachstellen erkannt werden. Somit kann insgesamt gesagt werden, dass die wichtigsten Schritte getan wurden, um die aktuelle Problemlage innerhalb des Betriebes zu verbessern.

5.2 Ausblick

Die implementierten Features sind komplett funktionsfähig. Allerdings ging aus dem Testlauf in dem Betrieb hervor, dass einige Elemente optimiert werden könnten, um einen schnelleren Umgang mit der Applikation zu ermöglichen. Deshalb soll in den folgenden Monaten das Feedback der Mitarbeiter teilweise umgesetzt werden. Die Eiken & Grosch Stadion GbR kann es sich sehr gut vorstellen, die Applikation nach den Verbesserungen regelmäßig bei dem Catering-Betrieb des VFL Osnabrück zu verwenden und auch in anderen Stadien die Applikation anzutesten. Deshalb verfolgt die Eiken & Grosch Stadion GbR eine Weiterentwicklung der Applikation mit großem Interesse.

Anhang

A Abbildungsverzeichnis

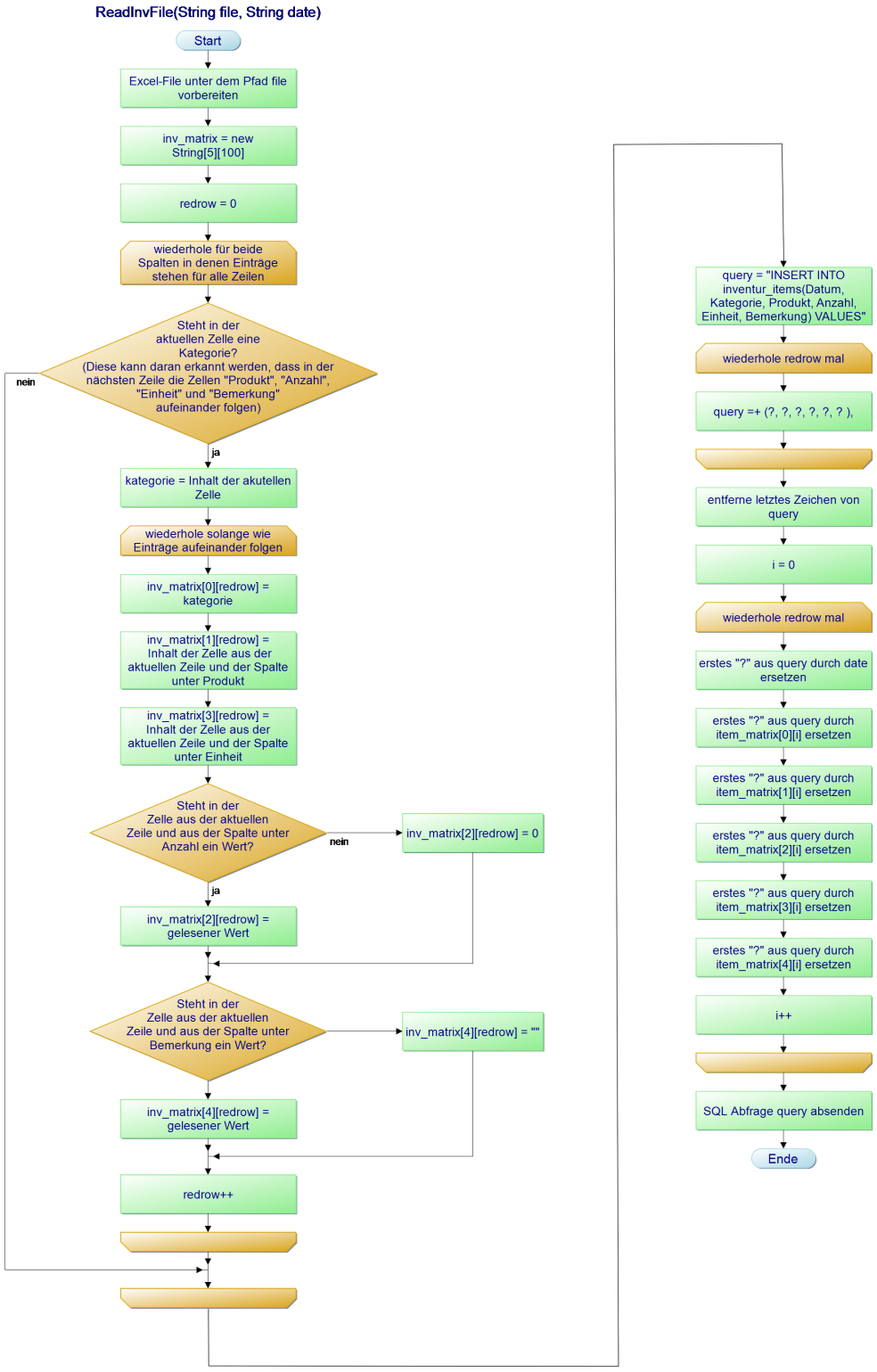

Abbildung 5: Programmablaufplan der Methode ReadInvFile(String file, String date).

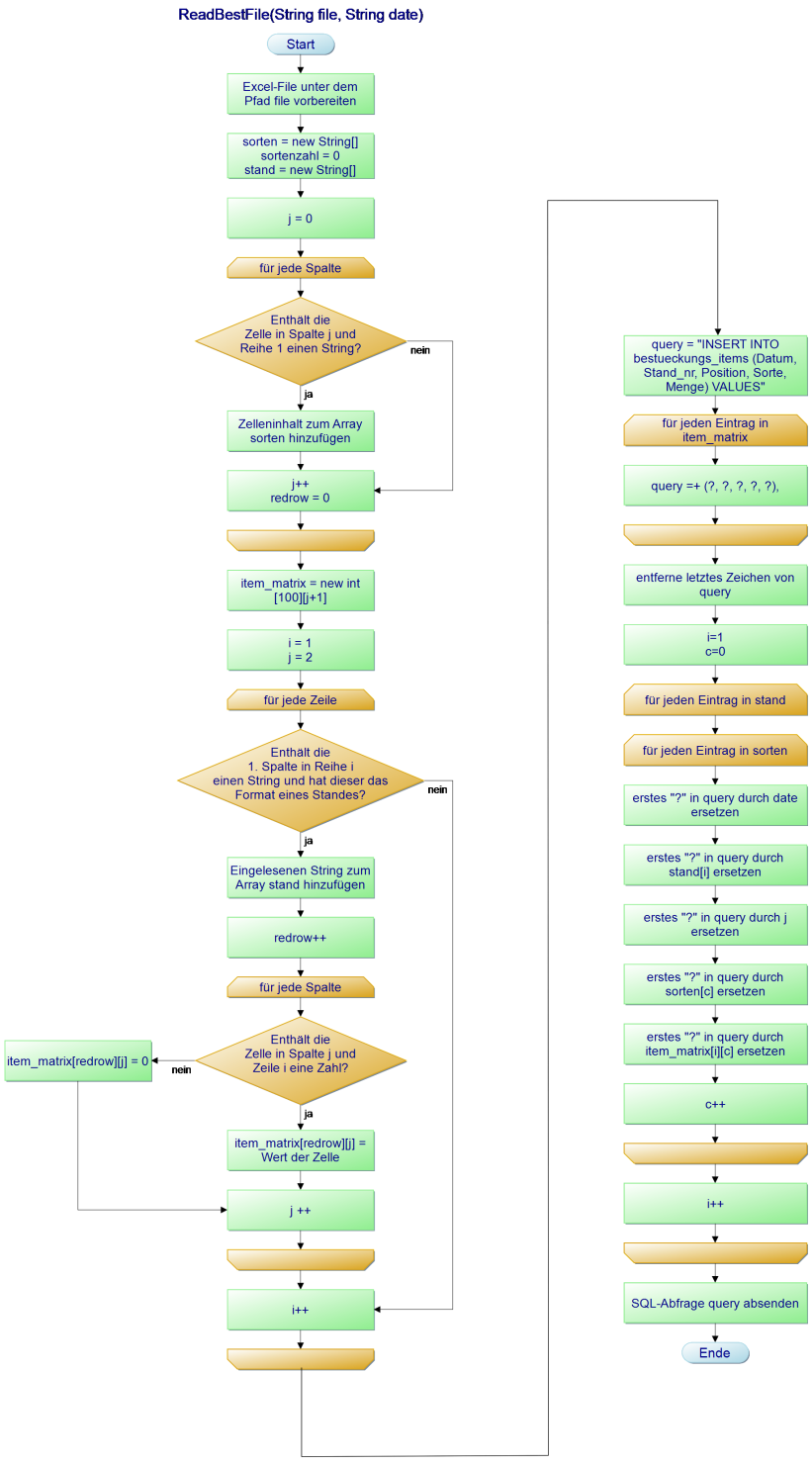

Abbildung 6: Programmablaufplan der Methode ReadBestFile(String file, String date).

Bremen II

Bestückungsliste VfL Osnabrück Stadioncatering
SOLL

05.05.2018

	Lagerort	Pils	Cola	Orange	Zitrone	Zero	Wasser	A-Schorle	Sport	Alkfrei	Wurst	Krakauer	Steaks	Suppe	Pommes	Fleischk.
o12																
o13																
o14																
o16																
o17																
o20																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s3																
s5																
s6																
s7																
s10																
s11																
s43																
s43a																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w1																
w2																
w36																
w37																
w39																
w42																
w42a																
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 7: Vorlage der Bestückungsliste.

Kühlhausinventur

Eiken & Grosch Stadioncatering GbR, Zillestrasse 4, 49134 Wallenhorst

Datum: _____

Fleisch-/Wurstwaren			
Produkt	Anzahl	Einheit	Bemerkung
Bratwurst Giersch		Stk.	
gebratene Wurst Giersch		Stk.	
gebratene Wurst Giersch in Currysauce		kg	
Kohlwurst Giersch		Stk.	
Bratwurst Kinnius		Stk.	
Krakauer		Stk.	
Pfeffernacken		Stk.	
Fleischkäse (Scheiben)		Stk.	
Fleischkäse (Laib)		kg	
Backschinken roh		kg	
Backschinken gegart		kg	

Backwaren			
Produkt	Anzahl	Einheit	Bemerkung
Baguettebrötchen		Stk.	
Baguettebrötchen (gebacken)		Stk.	
Kaiserbrötchen		Stk.	
Softbrötchen		Stk.	
Riesenbrezel (Teigling)		Stk.	
Riesenbrezel (gebacken)		Stk.	

Impulseis			
Produkt	Anzahl	Einheit	Bemerkung
Oreo Sandwich		Stk.	
Oreo Stieleis		Stk.	
Milka Stieleis		Stk.	
Daim Stieleis		Stk.	
10 for two		Stk.	
Nucki Erdbeer		Stk.	
Kaktus Stieleis		Stk.	

Suppen & Saucen			
Produkt	Anzahl	Einheit	Bemerkung
Gulaschsuppe		Liter	
Grünkohl (veg.)		Liter	
Grünkohl (mit Wurst)		Liter	
Kartoffelsuppe		Liter	
Mayonaise im Eimer		kg	
Ketchup im Eimer		kg	
Senf im Eimer		kg	
Mayotube		Stk.	
Ketchuptube		Stk.	
Senftube		Stk.	
Currysauce (Hela)		kg	
Currysauce (AVO)		kg	

Kartoffelprodukte & Salate			
Produkt	Anzahl	Einheit	Bemerkung
CN-Pommes (geschlossen)		kg	
CN-Pommes (Anbruch)		kg	
Aviko Super Crunsh (geschlossen)		kg	
Aviko Super Crunsh (Anbruch)		kg	
Krautsalat		kg	
Krautsalat (gepimpt)		kg	

Abbildung 8: Vorlage der Lebenmittelinventurliste.

Getränkeinventur

Eiken & Grosch Stadioncatering GbR, Zillestrasse 4, 49134 Wallenhorst

Datum: _____ Unterschrift: _____

Lagerort / Produkt	Fassbier Liter	Dunkelbier Liter	Coca Cola Liter	Orange Liter	Zitrone Liter	Zero Liter	Cola Light Liter	Classic Liter	Medium Liter	Naturell Liter	Sportscap Stk.	Schorle Liter	Sport Liter	Alkfrei 0,33	Alster 0,33	Pils 0,33	Glühwein Liter	Rum Flasche	Salvus Cola Flasche	Weißwein Flasche	Sekt Flasche	Rosé Flasche	Rotwein Flasche	O-Saft Flasche
W36																								
W37																								
W39																								
S3																								
S7																								
S10																								
O12																								
O14																								
O20																								
N25																								
N26																								
N27																								
N29																								
N30																								
N31																								
N33																								
Kühlwagen Ost																								
Tandem Ost																								
Kühlhaus																								
FK Nord																								
LT																								
gesamt:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 9: Vorlage der Getränkeinventurliste.

B Literatur

[1] 6 Wunderkinder GmbH. Wunderlist. https://www.wunderlist.com/de/, Accessed:04.10.2018.

[2] Any.do, 2018. https://www.any.do/, Accessed:04.10.2018.

[3] Splend Apps. Splendo, 2018. https://play.google.com/store/apps/details?id=com.splendapps.splendo&hl=de, Accessed:04.10.2018.

[4] Rahat Khanna; Sani Yusuf; Hoc Phan. *Ionic : Hybrid Mobile App Development*. Packt Publishing, June 14, 2017.

[5] Arvind Ravulavaru. *Learning Ionic - Second Edition*. Packt Publishing, April 28, 2017.

[6] Chris Griffith. *Mobile App Development with Ionic, Revised Edition*. O'Reilly Media, Inc., August 28, 2017.

[7] Tutorial. https://ionicframework.com/docs/intro/tutorial/, Accessed:11.10.2018.

[8] Testing your app - ionic framework, 2018. https://ionicframework.com/docs/v1/guide/testing.html, Accessed: 05.10.2018.

[9] Christian Ullenboom. *Java ist auch eine Insel - Einführung, Ausbildung, Praxis*. Rheinwerk Verlag, 2018.

[10] Jcomponent - java platform se7. https://docs.oracle.com/javase/7/docs/api/javax/swing/JComponent.html, Accessed:16.10.2018.

[11] Steve Suehring; Tim Converse; Joyce Park. *PHP6 and MySQL 6 Bible*. John Wiley & Sons, 2009.

[12] Stephan Kleuker. *Grundkurs Datenbankentwicklung*. Springer Vieweg, 2013.

[13] Sanjay Patni. *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS*. Apress, 2017.

[14] Sufyan bin Uzayr. *Learning WordPress REST API : a practical tutorial to get you up and running with the revolutionary WordPress REST API*. Packt Publishing, 2016.

[15] Slim framework. https://www.slimframework.com/, Accessed:11.10.2018.

[16] Pablo Solar Vilarino Carlos Perez Sanchez. *PHP Microservices*. Packt Publishing Ltd, 2017.

[17] Ionic platform documentation. https://ionicframework.com/docs/, Accessed:11.10.2018.