HTL Perg für Informatik

Machlandstraße 48,

4320 Perg,

Österreich

Diplomarbeit

Mobile Restaurant and Stampcard Management

Eingereicht von: Andreas Egger
Erik Puchinger

Betreuungslehrer: Dipl.-Ing. Christian Aberger

Auftraggeber: Aberger Software GmbH

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder inhaltlich aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Ort, Datum	*Andreas Egger*
Ort, Datum	*Erik Puchinger*

Danksagung

Danken möchten wir allen Personen, die uns im Laufe der Diplomarbeit stets unterstützt und uns bei sämtlichen auftretenden Fragen und Problemen weitergeholfen haben.

Besonderen Dank gilt unserem Professor, Betreuungslehrer und Auftraggeber Herrn Dipl.-Ing. Christian Aberger. Zum einen für die Idee der Diplomarbeit und zum anderem für die tatkräftige Unterstützung während der Entwicklung.

Des Weiteren möchten wir uns auch bei unseren Familien bedanken, die immer für uns dagewesen sind.

Herzlichen Dank!

Impressum

Schule:	HTL Perg für Informatik Machlandstraße 48 4320 Perg
Schuljahr:	2015/2016
Klasse:	5AHIF
Diplomarbeitstitel:	Mobile Restaurant and Stampcard Management
Diplomarbeitskandidaten:	Andreas Egger Erik Puchinger
Betreuungslehrer:	Dipl.-Ing. Christian Aberger
Auftraggeber:	Aberger Software GmbH

Inhaltsverzeichnis

1 Einleitung

1.1 Kurzbeschreibung

Aufgabenstellung

Kunden von Restaurants müssen Menüpässe bei sich tragen, um nach dem wiederholten Besuch Rabatte bzw. kostenlose Speisen zu erhalten, wie etwa eine gratis Pizza nach dem Kauf von zehn Pizzen. Diese Stempelpässe benötigen viel Platz in der Geldbörse, außerdem ist die Mitführung dieser unkomfortabel.

Das Ziel dieser Diplomarbeit war daher eine mobile Applikation zum einfachen digitalen Verwalten von Menüpässen zu entwickeln, welche Besuchern diverser Restaurants das Mitnehmen von Menüpässen abnimmt. Zusätzlich wurde eine Weboberfläche für Administratoren programmiert, mit welcher sämtliche Verwaltungstätigkeiten durchgeführt werden können.

Realisierung

Die Applikation wurde für die Betriebssystemen iOS und Android entwickelt. Mittels der App können Stempelpässe von im System eingetragenen Gaststätten bequem per Smartphone verwaltet werden. Ein Stempel wird zu einem Stempelpass hinzugefügt, indem der Benutzer einen vom Restaurant erhaltenen QR-Code einscannt. Hat der Kunde den virtuellen Stempelpass befüllt, kann dieser von einem Kellner zurückgesetzt werden. Zusätzlich hat der Benutzer die Möglichkeit, im Vorhinein alle Speisekarten der Restaurants anzusehen.

Über die Weboberfläche kann der Administrator einsehen, welche Produkte wie oft in welchem Zeitraum bestellt wurden. Anhand dieser Informationen ist es möglich, benutzerspezifische Werbung für die Nutzer der mobilen Apps einzuspielen. Außerdem werden mit Hilfe der Weboberfläche administrative Tätigkeiten, wie beispielsweise das Verwalten von Restaurants, Benutzern oder Produkten durchgeführt.

Ergebnis

Das System erleichtert Benutzern das Mitführen von Stempelpässen.

Die Diplomarbeit wird nach Abschluss an den Auftraggeber, die Aberger Software GmbH, übergeben. Diese führt im Anschluss letzte Anpassungen durch und startet dann den Vertrieb dieser Software.

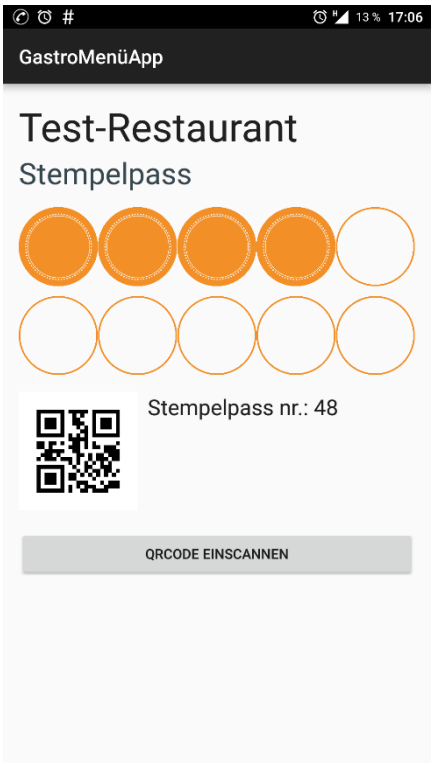


Abbildung 1: Screenshot der Android App (virtueller Stempelpass)

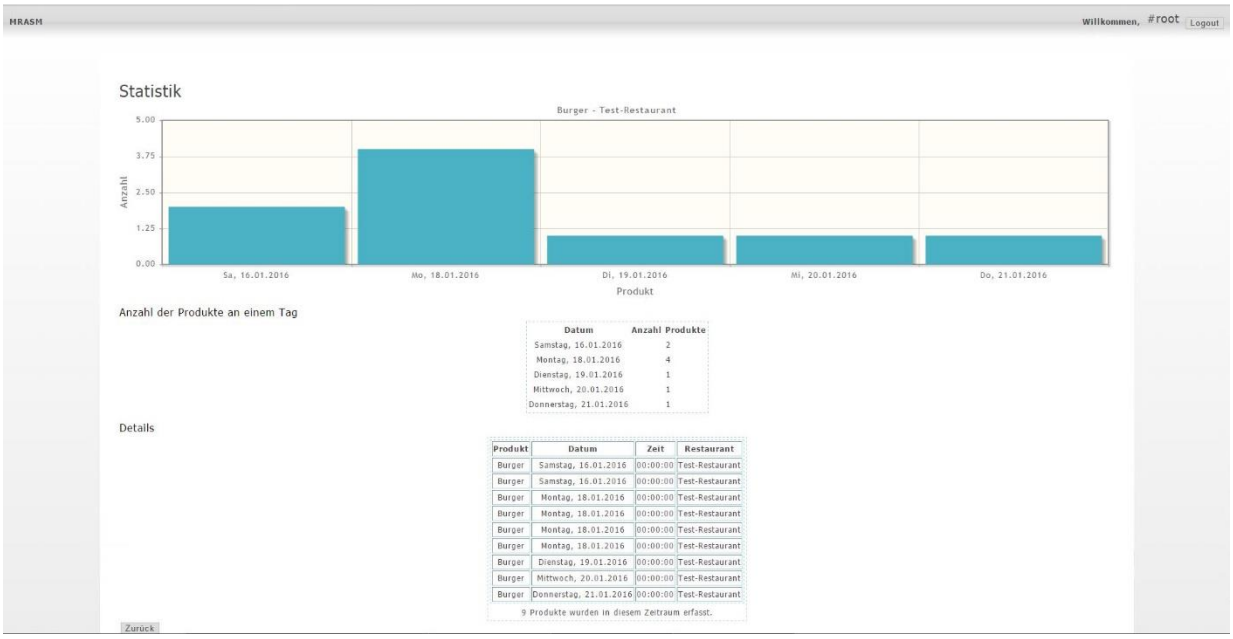


Abbildung 2: Screenshot der Webapplikation (Verkaufsstatistik)

1.2 Abstract

Task

Customers of restaurants have to carry stamp cards in order to gain discounts or free products after the purchase of a certain amount of food (e.g. a free pizza after buying 10 pizzas). These stamp cards take up a lot of space in the wallet and are uncomfortable to carry around.

Therefore the goal of this diploma is to develop a mobile application for the digital management of stamp cards, so that consumers do not have to carry stamp cards anymore. Additionally, a web application for all administrative tasks was designed.

Implementation

The application was developed for the operating systems iOS and Android. With this app, restaurant stamp cards can easily be managed by a smartphone. To add a stamp to the virtual stamp card, the user has to scan a QR-Code. Once a stamp card is filled, the customer receives a free meal and his virtual stamp card gets reset. Additionally, the user can look at the menu cards of the participating restaurants.

The web application allows the administrator to inform himself about which products were sold in a certain time period. Based on this information, it is possible to create user specific advertisements for the users of the mobile applications. Furthermore, tasks like managing restaurants, users or products can be performed with the web application.

Result

This diploma simplifies the process of managing restaurant stamp cards.

After completion, the diploma will be passed to our client, the Aberger Software GmbH. The company will optimize and market the software afterwards.

2 Beteiligte

2.1 Team

Abbildung 3: Diplomarbeitsteam

2.1.1 Andreas Egger

Persönliche Daten

Name:	Andreas Markus Egger
Geburtsdatum:	18. Juni 1997
Adresse:	Wimfeldstraße 19 3691 Nöchling
E-Mail:	andreas-egger97@gmx.at

Abbildung 4: Andreas Egger

2.1.2 Erik Puchinger

Persönliche Daten

Name:	Erik Puchinger
Geburtsdatum:	09. September 1996
Adresse:	Winden 77 4311 Schwertberg
E-Mail:	erik.puchinger@gmail.com

Abbildung 5: Erik Puchinger

2.2 Betreuungslehrer und Ansprechpartner

Der Betreuungslehrer dieser Diplomarbeit ist Herr Dipl.-Ing. Christan Aberger. Er hat uns seit der dritten Klasse in dem Freigegenstand Mobile Computing und seit dem vierten Jahrgang in dem Pflichtgegenstand Java unterrichtet. Im Zuge dessen wurden uns wesentliche Aspekte für diese Diplomarbeit gelehrt. Außerdem haben wir durch ihm die Idee für diese Arbeit erhalten.

Persönliche Daten

Name:	Christian Aberger
Adresse:	Softwarepark 37 4232 Hagenberg
E-Mail:	c.aberger@htl-perg.ac.at

Abbildung 6: Dipl.-Ing. Christan Aberger

2.3 Auftraggeber

Der Auftraggeber dieser Diplomarbeit ist die Aberger Software GmbH mit Sitz in Hagenberg im Mühlkreis. Unser Ansprechpartner und zugleich Betreuungslehrer ist Herr Dipl.-Ing. Christian Aberger.

Abbildung 7: Logo Aberger Software GmbH

Kontaktdaten

Name:	Aberger Software GmbH
E-Mail:	office@aberger.at
Ort:	4232 Hagenberg
Telefon:	+43 720 348 450
Webseite:	www.aberger.at

Das 1991 gegründete Softwareunternehmen, wurde zunächst als Ein-Mann-Betrieb geführt. 2007 übersiedelte das Unternehmen in das amsec Gebäude in Hagenberg. Derzeit sind 12 Personen beschäftigt.

3 Entstehung und Planung

Am Ende des vierten Jahrganges unterbreitete uns Herr Dipl.-Ing. Christan Aberger den Vorschlag eine Diplomarbeit zur digitalen Verwaltung von Restaurant-Stempelpässen im Auftrag seines Unternehmens, der Aberger Software GmbH zu schreiben.

Bereits in den Sommerferien 2015 hat die Vorbereitung und Entwicklung begonnen. Da Herr Andreas Egger einen Ferialjob bei der Aberger Software GmbH angenommen hatte, konnte er bereits große Teile der Diplomarbeit dort erledigen.

3.1 Thema und Aufgabenstellung

3.1.1 Aktueller Zustand

Am Wochenende möchte man wieder einmal ein Restaurant besuchen, jedoch ist man sich nicht sicher ob ein bestimmtes Gericht in diesem geführt wird. Das Restaurant hat jedoch auch keine Webseite um dies zu prüfen. Nach dem Essen möchte man nun bezahlen, doch der volle Stempelpass ist neben den ganzen anderen Stempelpässen in der Geldbörse unauffindbar, somit muss das ansonsten kostenlose Gericht bezahlt werden.

3.1.2 Problembereich

Restaurantbesuchern bleibt es oft, aufgrund einer fehlenden Website, verwehrt, Speisen und Menüs eines bestimmten Restaurants im Vorhinein anzusehen. Auch vergessen Kunden einer Gaststätte wiederholt ihre Stempelpässe, da diese aus Platzgründen nicht immer in der Geldbörse mitgeführt werden können. Somit können keine weiteren Stempel zu dem Pass hinzugefügt und auch keine eventuell kostenlose Speise bestellt werden.

Außerdem sind nicht immer die Allergene oder die konkreten Speisen eines Tagesmenüs in der Speisekarte des Restaurants aufgeführt und es ist mühsam bei jedem Gericht den Kellner um Hilfe zu bitten.

Darüber hinaus ist es Restaurantbetreibern nicht möglich, zielgerichtete Werbung an ihre Kunden zu senden, da bestellte Speisen nicht aufgezeichnet werden.

3.2 Allgemeine Zielsetzung

3.2.1 Technische Ziele

- Die Applikation soll auf Android und iOS Geräten laufen.
    - sämtliche Android Geräte, sowohl Smartphones als auch Tablets (ab Android 4.2)
    - alle iPhones, iPads und iPods (ab iOS 7)
- Die Webapplikation soll auf allen gängigen Browsern laufen.
- Die mobilen Applikationen sowie die Webapplikation sollen einfach und intuitiv zu bedienen sein.
- Die Nutzung neuer Technologien wie z.B.: Swift und PrimeFaces.

3.2.2 Generelle Ziele

- Die mobilen Apps und die Webapplikation sollen reibungslos und performant zusammenarbeiten.
- Der Ablauf soll so schnell und unkompliziert wie möglich sein.
- Kunden und Restaurantbesitzern soll durch die Diplomarbeit Abhilfe geschaffen werden.

3.3 Vorgehensplan

Der Vorgehensplan ist vor allem zu Beginn ein wichtiges Mittel gewesen, um sich einen guten Überblick über den voraussichtlichen Zeitaufwand zu machen.

Ein Vorgehensplan ist bei einem größeren Projekt wie auch bei einer Diplomarbeit unabdingbar. Nach dessen Erstellung hat sich bereits ein guter Überblick über den voraussichtlichen Zeitaufwand ergeben. Eine wesentliche Rolle spielte dabei der Projektstrukturplan. Dieser hat alle durchzuführenden Arbeitspakete übersichtlich und geordnet dargestellt. Somit konnte nach dessen Erstellung bereits eine Auskunft über die voraussichtlichen Arbeitsstunden gegeben werden. Ein genauer Plan darüber, wann diese Arbeitspakete zu erledigen sind, war für die Zeiteinteilung ebenfalls wichtig. In dieser Arbeit erfolgte diese Einteilung immer am Monatsbeginn; durch solch einen agilen Plan konnte gut auf auftretende Probleme reagiert werden.

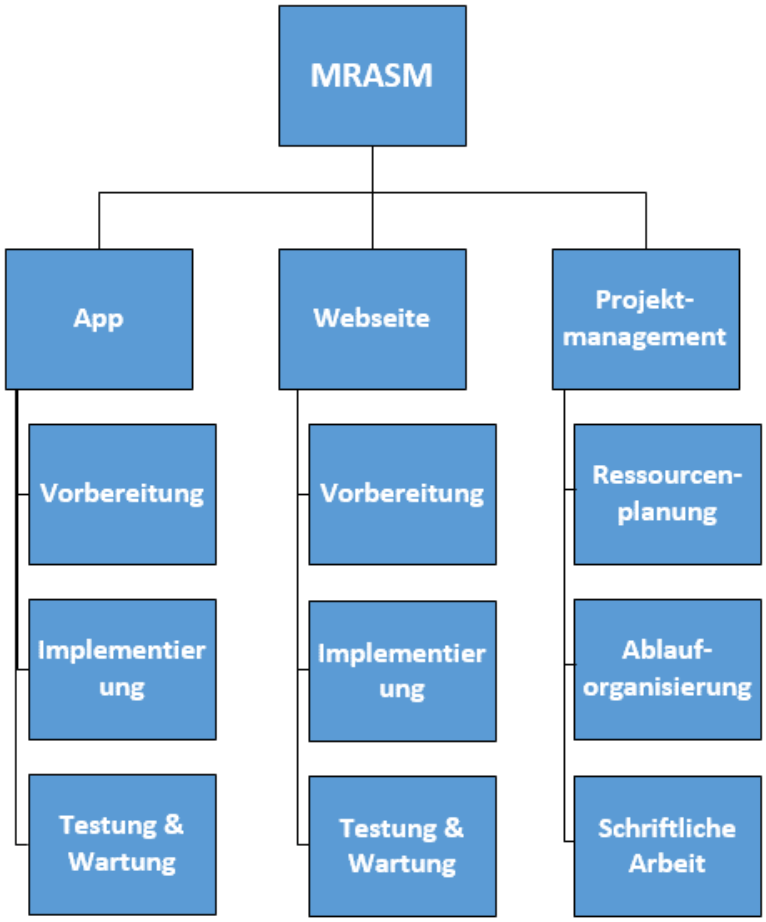


Abbildung 8: Vereinfachter Projektstrukturplan

Der Projektstrukturplan ist in drei Bereiche aufgeteilt. Diese Aufgabenbereiche wurden unter den Projektmitgliedern verteilt. Herr Andreas Egger bekam den Aufgabenbereich der mobilen Applikationen und Herr Erik Puchinger den der Webapplikation. Das Projektmanagement wurde gemeinsam erledigt.

3.3.1 Mobile Applikation

3.3.1.1 Grundlegende Vorgehensweise

Die nachfolgende Grafik veranschaulicht die grundlegende Vorgehensweise für die Realisierung der mobilen Applikationen.

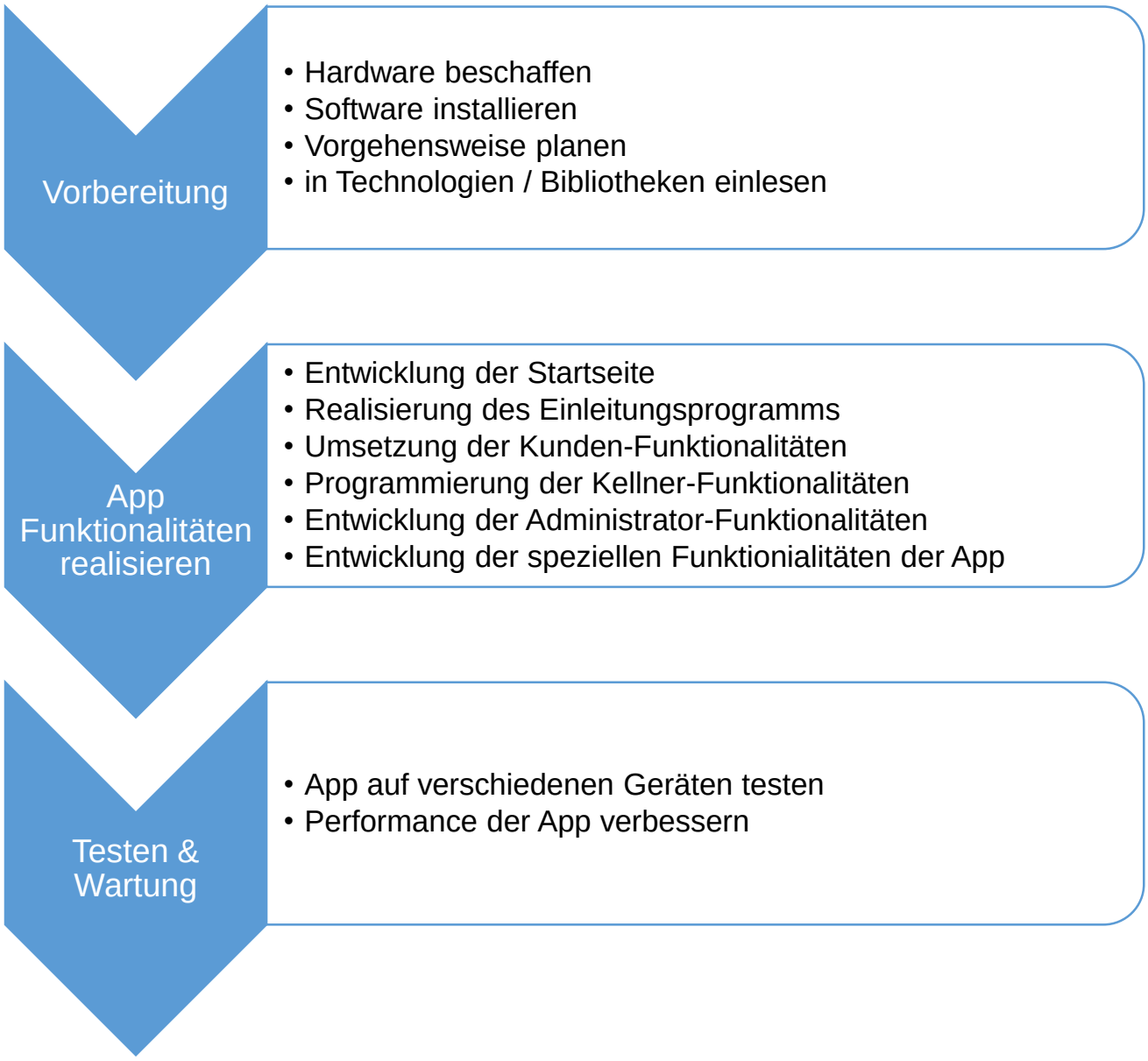


Abbildung 9: Vorgehensplan mobile Applikation

3.3.1.2 Vorbereitung

- Hardware beschaffen
- Software installieren
- Vorgehensweise planen
- in Technologien / Bibliotheken einlesen

3.3.1.2.1 Hardware beschaffen

Für die Entwicklung der Diplomarbeit mussten folgende Komponenten besorgt werden:

Komponente	Anzahl	Anforderungen
Android Mobiltelefon	1	Mindestversion Android 4.4
iPhone oder iPad	1	Mindestversion iOS 7
Server mit Internetzugang	1	2GB Ram aktueller Intel Prozessor (64 Bit, mind. zwei Kerne, Taktfrequenz > 1.6 GHz)
MacBook Air	1	2GB Ram Intel Core i3 11“ Display
Windows Laptop	2	4GB RAM Intel Core i3 15” Display

Die Anschaffung aller in der vorgehenden Tabelle enthaltenen Produkte hätte sich auf größere Kosten belaufen. Wir konnten uns allerdings sämtliche Kosten für die Anschaffung der Hardware ersparen, da wir bereits ein Android Mobiltelefon und zwei Windows Laptops besaßen. Ein iPad, ein MacBook Air und ein Server mit Internetzugang wurde von unserer Lehranstalt zur Verfügung gestellt.

3.3.1.2.2 Software installieren

Nach der Beschaffung der benötigten Hardware wurde unmittelbar mit der Installation der Software begonnen.

Zunächst musste auf dem Server das Betriebssystem Windows Server 2012 R2 installiert werden. Anschließend wurde auf den Windows Laptops die Entwicklungsumgebung Android Studio und auf dem MacBook XCode installiert. Auf dem Server wurde das Datenbankmodellierungsprogramm MySQL Workbench, sowie das Programm XAMPP, welches einen Webserver zur Verfügung stellt, installiert.

Für den Datenaustausch zwischen einer Android Applikation und einem Datenbankserver bzw. einer iOS Applikation und einem Datenbankserver wird üblicherweise eine XML[1] oder JSON[2] Schnittstelle verwendet. Die Auswahl fiel bei dieser Diplomarbeit auf eine XML-Schnittstelle. Da aber bereits existierende Lösungen für diese die Funktionalität des Webservers beeinträchtigt hätten, musste sie selbst entwickelt werden. Die Wahl der Programmiersprache fiel dabei auf PHP[3], da diese in XAMPP bereits enthalten ist, welches auch für die Gewährleistung der Webseitensicherheit benötigt wird.

Der Programmcode der Schnittstelle befindet sich im Abschnitt 4.4.2.

3.3.1.2.3 Vorgehensweise planen

Bei der Planung der Vorgehensweise war es besonders wichtig ein Pflichtenheft zu verfassen. Damit konnte ein Gesamtüberblick über die zu entwickelnden Funktionalitäten geschaffen werden.

Nach der Fertigstellung des Pflichtenheftes wurde mit der Erarbeitung der Arbeitspakete aus dem Projektstrukturplan begonnen.

Als die Arbeitspakete definiert waren, begannen wir festzulegen wie diese Aufgaben abgearbeitet und verteilt werden. Dabei wurde vereinbart, dass alle eineinhalb Wochen eine Skype-Konferenz abzuhalten ist, bei welcher immer zwischen ein bis drei Arbeitspakete jedem Teammitglied zugeteilt werden. Diese Arbeitspakete mussten bis zur nächsten Konferenz fertiggestellt werden. Jede dieser Konferenzen wurde zusätzlich in Form eines Protokolls schriftlich festgehalten.

3.3.1.2.3.1 Vorgehensweise für die Entwicklung der App planen

Zwar vermittelte der Projektstrukturplan bereits einen groben Überblick darüber welche Aufgaben mit der Entwicklung der App verbunden sind, allerdings musste noch genauer definiert werden, in welcher Reihenfolge die einzelnen Arbeitspakete abzuarbeiten sind.

Dabei kam es zu dem Entschluss, dass zuerst das gesamte Grundgerüst der mobilen Applikation realisiert wird. Erst im Anschluss werden alle spezifischen Funktionalitäten wie beispielsweise der QR-Code[4] Scanner oder die Benachrichtigungen entwickelt.

[1] XML: Extensible Markup Language, ein Speicherformat welches sowohl von Menschen als auch von Maschinen leicht lesbar ist.
[2] JSON: JavaScript Object Node, ein Speicherformat welches sowohl von Menschen als auch von Maschinen leicht lesbar ist, allerdings etwas komprimierter als XML ist.
[3] PHP: Hypertext Preprocessor, eine Skriptsprache welche für die Entwicklung von Webseiten verwendet wird.
[4] QR-Code: Quick Response Code, eine Weiterentwicklung des Barcodes mit einer größeren Speicherkapazität.

3.3.1.2.4 In Technologien / Bibliotheken einlesen

Bevor mit der Entwicklung der mobilen Applikationen begonnen werden konnte, mussten wir uns zuerst in diverse neue Technologien und Bibliotheken einlesen. Die nachfolgende Tabelle enthält dabei alle wesentlichen Technologien und Bibliotheken, welche in den mobilen Applikationen verwendet wurden.

Technologie / Bibliothek	Verwendung
Java	Die Android Applikation wurde mit der Programmiersprache Java entwickelt.
Android API	Bei der Entwicklung der Android Applikation wurde die offizielle Android API verwendet.
Swift	Die iOS Applikation wurde mit der Programmiersprache Swift entwickelt.
ZBar	Die Bibliothek ZBar kam in der Android App zum Einsatz, um das Einscannen von QR-Codes zu realisieren.
AEXML	AEXML ist eine Bibliothek für die Programmiersprache Swift. Diese wurde verwendet um den Zugriff auf die Datenbankschnittstelle in der iOS Applikation umzusetzen.

3.3.1.2.4.1 Technische Vorbereitung

Das Ziel der technischen Vorbereitung war es eine funktionierende Verbindung zwischen einem Smartphone und der Datenbank herzustellen. Dafür wurde zuerst ein Datenbankmodell benötigt, wodurch sich die Frage stellte ob eher ein code-first oder ein model-first Ansatz gewählt werden soll.

Beim code-first Ansatz wird zuerst der Programmcode entwickelt und erst danach wird das Datenbankmodell automatisch generiert.

Beim model-first Ansatz wird zuerst das Datenbankmodell entwickelt. Im Anschluss werden die Teile des Programmcodes, welche die Datenbank abbilden, automatisch generiert.

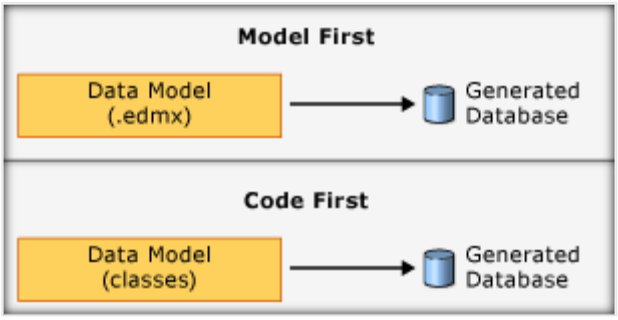


Abbildung 10: Model First vs. Code First

Da eine grafische Benutzeroberfläche für die Erstellung und Wartung des Datenbankmodells bevorzugt wurde, fiel die Wahl auf den model-first Ansatz.

Daher wurde mit der Erarbeitung des Datenbankmodells begonnen, wobei das Pflichtenheft als praktische Unterstützung zur Verfügung stand.

3.3.1.3 App Funktionalitäten realisieren

- Entwicklung der Startseite
- Realisierung des Einleitungsprogramms
- Umsetzung der Kunden-Funktionalitäten
- Programmierung der Kellner-Funktionalitäten
- Entwicklung der Administrator-Funktionalitäten
- Entwicklung der besonderen Funktionalitäten der App

3.3.1.3.1 Entwickeln der Startseite

Nach Abschluss der Vorbereitungsphase wurde mit der Entwicklung der App begonnen.

Zuerst wurde die Startseite der mobilen Applikation entwickelt. Diese besteht aus einer Landkarte auf der alle teilnehmenden Restaurants abgebildet sind.

3.3.1.3.2 Realisieren des Einleitungsprogramms

Im Anschluss wurde ein Einleitungsprogramm erstellt, um neuen Benutzern die Funktionalitäten der App näher zu bringen. Dieses Einleitungsprogramm wurde bis zum Abschluss der Entwicklung der Kunden-Funktionalitäten laufend erweitert.

3.3.1.3.3 Umsetzen der Kunden-Funktionalitäten

Da die Entwicklung der Kunden-Funktionalitäten alleine ein zu großes Arbeitspakete gewesen wäre, wurde es in folgende Pakete unterteilt:

- entwickeln einer Restaurantspeisekarte
    - Die Restaurantspeisekarte bildet eine Speisekarte ab, welche alle Produkte das jeweilige Restaurant enthält.
- realisieren einer Tagesmenü-Übersicht
    - Die Tagesmenü-Übersicht stellt alle verfügbaren Tagesmenüs des jeweiligen Restaurants dar.
- umsetzen eines virtuellen Stempelpasses
    - Der virtuelle Stempelpass stellt alle Funktionen, die auch ein herkömmlicher Stempelpass bietet, bereit.
- verlinken zur jeweiligen Restauranthomepage
    - In der mobilen Applikation kann der Benutzer ohne größere Umwege auf die jeweilige Restauranthomepage gehen. Dies wurde mittels eines Links zur Webseite des jeweiligen Restaurants realisiert.
- entwickeln der Werbungsanzeige
    - In der jeweiligen Speiseansicht, sowie Tagesmenüansicht wird direkt Werbung angezeigt.

- realisieren der Allergenübersicht
    - Das Anzeigen der Allergene eines Produktes wurde ebenfalls in der mobilen Applikation realisiert. Die Angabe dieser Information ist seit dem 13. Dezember 2014 innerhalb der EU verpflichtend.

3.3.1.3.4 Entwicklung der Kellner-Funktionalitäten

Die Entwicklung der Kellner-Funktionalitäten wurde ebenfalls aufgrund seines Umfangs in mehrere Arbeitspakete unterteilt.

- entwickeln des Stempelkartenscanners
    - Die Entwicklung des Stempelkartenscanners wurde erst nach der Umsetzung des QR-Code Scanners realisiert. Da der Stempelkartenscanner auf die Funktionalität dessen zugreift.
- realisieren der Stempelkartenverwaltung
    - Bei der Entwicklung der Stempelkartenverwaltung wurde darauf geachtet, dass diese die wesentlichen Informationen eines Stempelpasses wiedergibt. Damit Kunden ihre Stempelpässe einlösen können ist eine Funktion enthalten, mit welcher Stempelpässe zurückgesetzt werden können.

3.3.1.4 Umsetzung der Administrator-Funktionalitäten

Die Hauptfunktionalität für Administratoren ist die Verwaltung von Tagesmenüs. Diese besteht aus folgenden Teilfunktionalitäten:

- anzeigen
- hinzufügen
- bearbeiten
- löschen

3.3.1.4.1 Umsetzung der speziellen App-Funktionalitäten

Die Umsetzung der speziellen App-Funktionalitäten besteht aus der Entwicklung aller Funktionen, welche nicht unmittelbar in der Anwendung wahrgenommen werden.

Darunter werden folgende Funktionalitäten verstanden:

- ORM Tool[5]
    - Da für das Datenbankmodell ein model-first Datenbankansatz verwendet wurde, musste zunächst nach einer kostenlosen Bibliothek gesucht werden welche diesen Ansatz unterstützt. Da aber keine passende gefunden wurde, musste selbst ein ORM Tool entwickelt werden.

[5] ORM Tool: Ein Object Relational Mapping Tool realisiert die objektrelationale Abbildung von Datenbankobjekten in Objekten der jeweiligen Programmiersprache.

- QR-Code Scanner
    - Der QR-Code Scanner wurde ebenfalls selbst entwickelt. In der Android Applikation musste jedoch auf die ZBar Bibliothek zurückgegriffen werden, da die Android API selbst keine Schnittstelle zum Scannen des QR-Codes bereitstellt.
- Push Benachrichtigungen[6]
    - Push Benachrichtigungen wurden entwickelt, um Benutzern mittels benutzerspezifischer Werbung über die Angebote der Restaurants zu informieren.

3.3.1.5 Testen und Wartung

- App auf verschiedenen Geräten testen
- Performance der App verbessern

3.3.1.5.1 Mobile Applikationen auf verschiedenen Geräten testen

Nachdem die mobilen Applikationen entwickelt waren, musste gewährleistet werden, dass diese Applikationen problemlos auf verschiedenen Geräten funktionieren.

Allerdings waren wir nur in Besitz von zwei unterschiedlichen Android Mobiltelefonen sowie einem iPad. Daher wurden für weitere Testzwecke Emulatoren verwendet, welche diverse mobile Geräte simulieren können.

Sobald es auf einem Gerät zu Problemen kam, wurden diese korrigiert. Im Anschluss musste diese Applikation wieder auf allen Geräten getestet werden um neue Fehler zu vermeiden.

Liste der getesteten Geräte:

Gerät	Betriebssystem(e)
OnePlus One	Android: 4.4; 5.0; 5.1; 6.0
Moto G 1. Generation	Android: 5.1
Samsung Galaxy S4	Android: 4.4; 5.0
Samsung Galaxy S4 LTEA	Android: 4.4; 5.0
Google Nexus 5 Emulator	Android: 5.1
Google Nexus 6 Emulator	Android: 5.1; 6.0
iPad 2	iOS: 9.0
iPad 3 Emulator	iOS: 8.4; 9.0; 9.2
iPhone 6 Emulator	iOS 8.4; 9.0
iPhone 5S Emulator	iOS 8.4; 9.0

[6] Push Benachrichtigungen: Diese werden verwendet um Benutzern mobiler Applikationen Benachrichtigungen mitzuteilen.

3.3.1.5.2 Performanceverbesserung der App

Ein weiterer wichtiger Punkt der Wartung der App war es dessen Performance zu verbessern. Das meiste Verbesserungspotential ergab sich bei der Kommunikation mit der Datenbank.

Daher wurde UTF-8[7] als Zeichenkodierung für die Datenbank sowie deren Schnittstelle verwendet.

Darüber hinaus hatten die selbst entwickelten DAOs[8] das Problem, dass sie sich bereits bei ihrer Initialisierung mit der Datenbank verbunden haben und daraufhin alle Daten der jeweiligen Abfrage heruntergeladen haben. Dieses Problem konnte allerdings durch geringe Quellcodeänderungen behoben werden.

Beispiel für ein altes Datenbankzugriffsobjekt:

```
public class ProductDao{
    //Variables
    private MySQLConnection mySQLConnection = null;
    private ArrayList<Product> products = new ArrayList<Product>();

    //Constructor
    public ProductDao (MySQLConnection mySQLConnection){
        this.mySQLConnection = mySQLConnection;
        this.setProducts(new ArrayList<Product>());
        ArrayList<MySQLRow> rows = this.mySQLConnection.
        executeQuery("select * from product");
        for (int i = 0; i < rows.size(); i++){
            this.getProducts().add(new Product(rows.get(i)));
        }
    }

    //Operative Methods
      public Product findProductByRestaurant (Restaurant restaurant){
            for (int i = 0; i < this.products.size(); i++){
                  if (this.products.get(i).getRestaurantIdRestaurant() ==
                       restaurant.getIdRestaurant()){
                         return this.products.get(i);
                  }
            }
            return null;
      }
}
```

Wie man an diesem Quellcode klar erkennen kann, würde sich das Mobiltelefon bei dem folgenden Aufruf den gesamten Inhalt der Produkttabelle herunterladen:

```
ProductDao productDao = new ProductDao(this.mySQLConnection);
List<Product> products = productDao.findProductByRestaurant(restaurant);
```

[7] UTF-8: Unicode Transformation Format-8 ist ein Zeichenkodierungsformat, welches weit verbreitet ist und einen sehr geringen Speicherbedarf pro Zeichen hat.

[8] DAOs: Data Access Objects bzw. Datenzugriffsobjekte, werden verwendet um in einer Applikation auf eine Datenbank zuzugreifen.

Beispiel für ein neues Datenbankzugriffsobjekt:

```
public class ProductDao extends Dao{
    //Variables
    private ArrayList<Product> products = new ArrayList<Product>();

    //Constructor
    public ProductDao (MySQLConnection mySQLConnection){
        super(mySQLConnection);
    }

      public ArrayList<Products> runQuery(String query){
        this.setProducts(new ArrayList<Product>());
        for (MySQLRow row : super.runMySQLQuery(query)){
            this.getProducts().add(new Product(row));
        }
            return this.products;
      }

    //Operative Methods
      public Product findProductByRestaurant (Restaurant restaurant){
            this.runQuery("select * from product where
                          product.restaurant_idrestaurant="
                          + restaurant.getIdRestaurant());
            if (this.product.size() > 0){
                  return this.product.get(0);
            }
            return null;
      }
}
```

Wenn man mit diesem verbesserten Datenbankzugriffsmodellen den vorigen Aufruf wiederholt, werden nur die benötigten Daten aus der Datenbank geladen.

3.3.2 Webapplikation

In der nachfolgenden Grafik ist die grundlegende Vorgehensweise der Umsetzung der Webapplikation dargestellt.

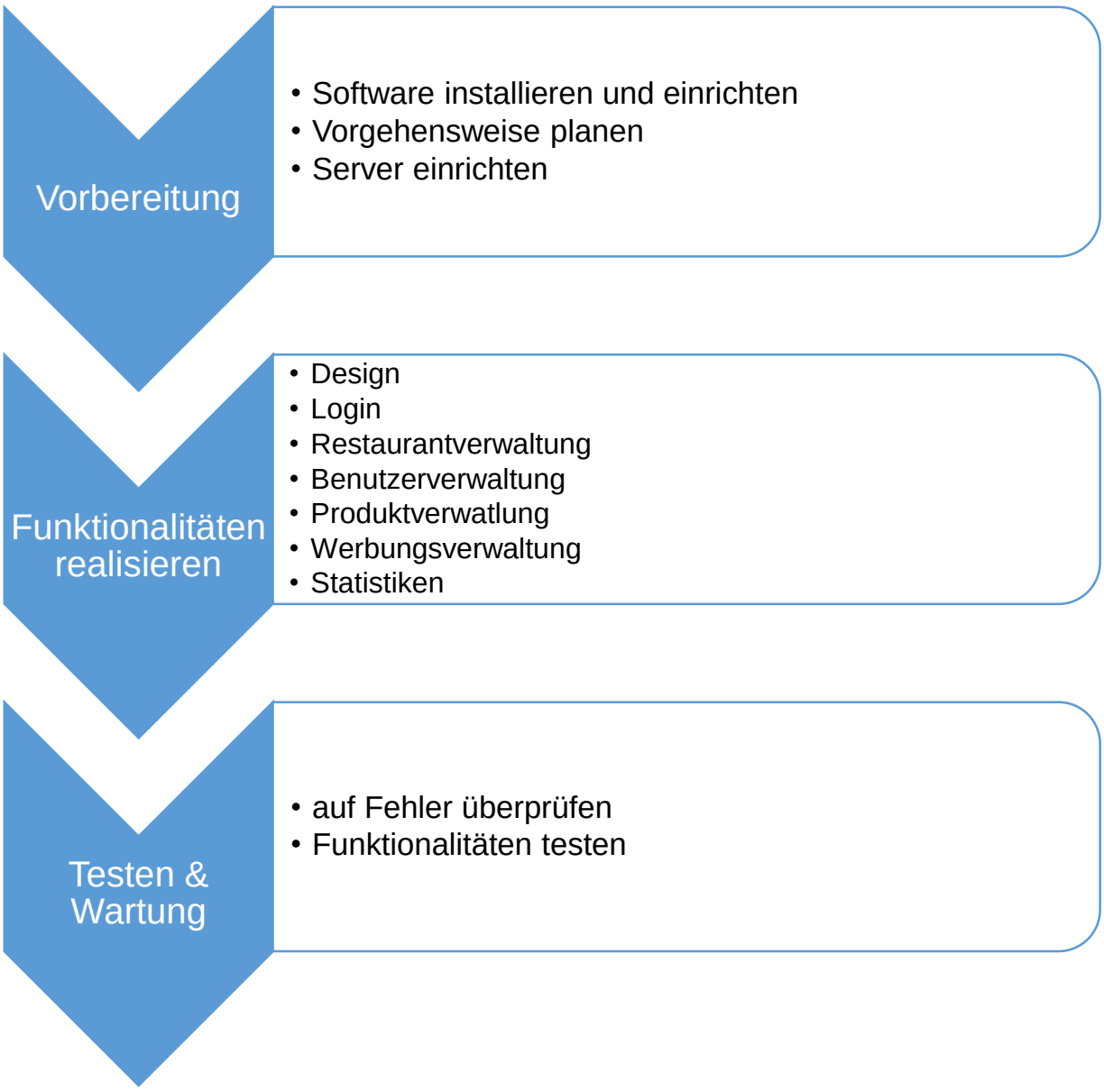


Abbildung 11: Vorgehensplan Webapplikation

3.3.2.1 Vorbereitung

- Vorgehensweise planen
- Server vorbereiten

Die Vorbereitung war einer der wichtigsten Aspekte der Diplomarbeit, da später verwendeten Technologien ausgewählt und eingerichtet wurden.

Des Weiteren musste man sich im Vorhinein mit den jeweiligen Technologien vertraut machen, um Komplikationen möglichst zu vermeiden.

3.3.2.1.1 Vorgehensweise planen

Unter diesen Punkt fallen unter anderem die Auswahl der passenden Programmiersprache und Entwicklungsumgebung.

3.3.2.1.1.1 Technologien auswählen

Einer der ersten Schritte in der Planung der Diplomarbeit war die Auswahl passender Technologien. Für die Programmierung der Webapplikation wurde JSF, eine Webtechnologie der Java Platform, Enterprise Edition und der WildFly Server gewählt, da diese Technologien bereits im Schulunterricht behandelt wurden und sämtliche Ansprüche zur Erstellung der Webapplikation erfüllten. Als Entwicklungsumgebung wurde Eclipse gewählt, da diese speziell auf Java ausgelegt ist.

3.3.2.1.2 Server vorbereiten

Da die Diplomarbeit auf einen Server für die MySQL Datenbank und die Weboberfläche angewiesen ist, war es wichtig einen passenden Server anzuschaffen. Somit musste die MySQL Datenbank und die Weboberfläche nicht lokal am Rechner betrieben werden.

Um dies zu bewerkstelligen, wurde ein Schulserver beantragt und Windows Server 2012 darauf installiert. Anschließend musste auf diesem eine MySQL Datenbank über XAMPP aufgesetzt und der WildFly Server sowie Eclipse eingerichtet werden. Dabei war es nötig die aktuellste Java Version herunterzuladen und zu installieren. Weiters wurde zur Verwaltung der MySQL Datenbank die Software MySQL Workbench benötigt. Um die Website auf dem Server zu testen, ist der Browser Google Chrome installiert worden. Für den Fernzugriff auf den Server wurde Remote Desktop auf dem Windows Server 2012 aktiviert.

3.3.2.2 Implementierung

- Programmierung des Logins
- Entwicklung der Restaurantverwaltung
- Entwicklung der Benutzerverwaltung
- Entwicklung der Produkt- und Tagesmenüverwaltung
- Programmierung der Werbungserstellung
- Entwicklung der Statistik-Anzeige

3.3.2.2.1 Design

Das grundlegende Design wurde den Funktionen entsprechend entworfen und in CSS umgesetzt. Dazu wurde ein CSS-Template geschrieben und mittels eines JSF-Templates in den entsprechenden Seiten eingebunden. So ist der Header beispielsweise auf allen Seiten gleich. Das bringt den Vorteil mit sich, dass keine Codeduplizierung vorkommt und Änderungen auf allen Seiten sofort übernommen werden.

Das Design wurde im Laufe der Arbeit immer wieder angepasst. So wurde z.B. im Nachhinein das Layout des Startmenüs grundlegend verändert und der Hintergrund auf allen Seiten ausgetauscht.

Für nähere Informationen zum Design siehe Abschnitt 4.2.2.

3.3.2.2.2 Login

Der Login bildet den Zugang auf die Webseite. Je nach Benutzergruppe (Administrator oder Restaurant-Administrator) hat der Benutzer unterschiedliche Sichten auf die Weboberfläche.

Der Administrator kann sämtliche Benutzer, Restaurants und Produkte verwalten. Der Restaurant-Administrator hat nur die Möglichkeit, Benutzer und Produkte des eigenen Restaurants zu verwalten.

3.3.2.2.2.1 Anmeldung und Sessions

Jeder Benutzer bekommt bei der Anmeldung eine Session zugeteilt. Mittels Sessions können auf der Seite unterschiedliche Benutzer differenziert werden und Benutzer mit falschen Zugangsdaten ausgesperrt werden.

Dazu musste zuerst die Login XHTML-Seite programmiert werden, welche zwei Eingabefelder für das Passwort und den Benutzernamen bietet.

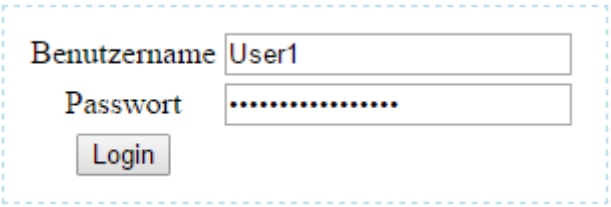


Abbildung 12: Login der Webapplikation

```
<h:outputLabel value="Benutzername" />
<h:inputText value="#{userBean.username}" required="true" requiredMes-
sage="Benutzername Eingabe!" id="username"/>
<h:outputLabel value="Passwort" />
<h:inputSecret value="#{userBean.password}" required="true" requiredMes-
sage="Passwort Eingabe!" id="password"/>
<h:commandButton action="#{userBean.login()}" value="Login" />
```

Die Seite überprüft ob alle Eingabefelder ausgefüllt sind. Ist das nicht der Fall, erscheint eine Fehlermeldung. Bei erfolgreicher Validation des Benutzernamen und Passworts werden die Benutzerdaten in der *UserBean*-Klasse abgeglichen. Dabei erfolgt eine Überprüfung ob der angegebene Benutzer im System eingetragen ist und welcher Benutzergruppe dieser angehört.

```
public String login() throws NoSuchAlgorithmException {
        if (checkUser(username, encode(password))==1){ //Restaurant-Admin
            return "welcome?faces-redirect=true";
        }
        if (checkUser(username, encode(password))==2){ //Administrator
            HttpSession session = SessionBean.getSession();
            session.setAttribute("username", "#"+username);
            session.setAttribute("userid", "0");
            return "welcome?faces-redirect=true";
        }
        return null;
}
```

Bei erfolgreicher Prüfung wird eine *HttpSession* erstellt, um später den eingeloggten Benutzer zu identifizieren.

```
public static HttpSession getSession() {
        return (HttpSession) FacesContext.getCurrentInstance()
                             .getExternalContext().getSession(false);
}
//gibt den Namen des eingeloggten Users zurück
public static String getUserName() {
        HttpSession session = (HttpSession) FacesContext
                             .getCurrentInstance()
                             .getExternalContext().getSession(false);
        return session.getAttribute("username").toString();
}
```

In der Klasse *AuthorizationFilter* werden Benutzer der Webapplikation die nicht eingeloggt sind, daran gehindert ohne Anmeldung Zugriff auf die Seite zu erlangen. Dazu wird die bestehende Methode *doFilter* überschrieben. Bei dem Versuch direkt auf eine Seite innerhalb des Logins zuzugreifen, wird ein unautorisierter Benutzer automatisch auf die Seite *login.jsf* weitergeleitet.

```
@WebFilter(filterName = „AuthFilter“, urlPatterns = {“*.jsf“,“*.xhtml“ })
public class AuthorizationFilter implements Filter {
    @Override
    public void doFilter(ServletRequest request, ServletResponse re-
sponse, FilterChain chain) throws IOException, ServletException {
        try {
            HttpServletRequest reqt = (HttpServletRequest) request;
            HttpServletResponse resp = (HttpServletResponse) response;
            HttpSession ses = reqt.getSession(false);
            String reqURI = reqt.getRequestURI();
            if (reqURI.indexOf(“/login.jsf”) >= 0||
               (ses != null && ses.getAttribute(“username”) != null)||
                reqURI.indexOf(“/public/”) >= 0||
                reqURI.contains(“javax.faces.resource”)){
                  chain.doFilter(request, response);
            }
            else{
                  resp.sendRedirect(reqt.getContextPath() + “/login.jsf”);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Beim Klicken auf den Logout-Button, der sich im Header jeder Seite befindet, wird die Session auf *invalidate* gesetzt und der Benutzer wird zur Login Seite weitergeleitet.

```
HttpSession session = SessionBean.getSession();
session.invalidate();
return “login?faces-redirect=true”;
```

3.3.2.2.2.2 Passwort Verschlüsselung

Bei der Abspeicherung des Passworts in der Benutzerverwaltung und bei der Abgleichung des Passwortes beim Login wird eine Verschlüsselung verwendet. Dazu wurde eine PBKDF2WithHmacSHA1-Methode in Java implementiert, die einen Klartext mittels einer Hashfunktion verschlüsselt.

Für weiter Informationen dieser Verschlüsslung siehe Abschnitt 6.2.3.11.

3.3.2.2.3 Restaurantverwaltung

Die Restaurantverwaltung bietet die Möglichkeit Restaurants zu bearbeiten.

3.3.2.2.3.1 Restaurantübersicht

In der Restaurantübersicht werden alle Restaurants des jeweiligen Benutzers angezeigt.

Um dies zu realisieren wurde ein eigenes Session-Attribut *userid* erstellt, indem die ID des aktuell eingeloggten Benutzers abgespeichert wird. Mit Hilfe dieses Attributes kann von jeder Stelle im Programm auf die ID zugegriffen werden und somit die dem Benutzer zugeteilten Restaurants herausgesucht werden.

```
//sucht zu dem eingeloggtem User einer bestimmten Usergruppe alle Restau-
rants
public List<UsersToRestaurant> findByUser(int usergroupId) {
            return entityManager.createQuery("from UsersToRestaurant
             where usergroup_idusergroup=:idGroup user_iduser=:idUser",
             UsersToRestaurant.class)
             .setParameter("idGroup", usergroupId)
             .setParameter("idUser", Integer.parseInt(SessionBean
                                                  .getUserId()))
             .getResultList();
}
```

3.3.2.2.3.2 Restaurant hinzufügen und bearbeiten

Die Funktion des Restaurant Hinzufügens steht nur dem Administrator offen. Restaurant-Administratoren können lediglich ihre eigenen Restaurants bearbeiten.

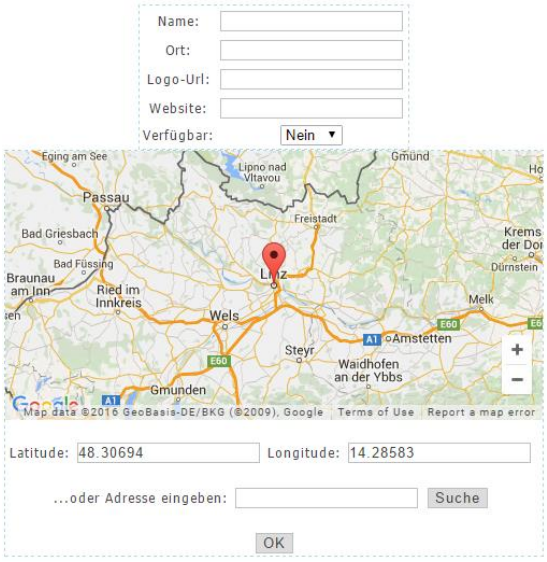


Abbildung 13: Restaurant hinzufügen Funktion

Um den Standort eines Restaurants möglichst einfach zu setzten, wurde der jQuery Latitude and Longitude Picker for Google Maps verwendet und an die Seite angepasst.

Wurde alles eingegeben und mit OK bestätigt, werden die Daten samt den Koordinaten des Restaurants in der Datenbank abgespeichert.

3.3.2.2.4 Benutzerverwaltung

Administratoren haben die Möglichkeit alle Benutzer zu bearbeiten. Restaurant-Administratoren können nur ihr eigenes Profil und die Verknüpfungen der Benutzer, die mit ihren Restaurants verbunden sind, ändern.

Damit Administratoren und Restaurant-Administratoren Benutzer zu Restaurants zuordnen können, wurde die Drop-Down-Liste *p:selectCheckboxMenu* von der JSF-Erweiterung PrimeFaces verwendet.

```
<h:outputLabel for="restaurant">
      Administrator bei:
</h:outputLabel>
<p:selectCheckboxMenu id="restaurant" value="#{userBean.adminRestau-
rantList}" label="auswahl" filter="true">
      <f:selectItems value="#{userBean.restaurants}" var="restaurant"
       itemLabel="#{restaurant.restaurantName}"
       itemValue="#{restaurant.idrestaurant}" />
</p:selectCheckboxMenu>
```

Die Drop-Down-Liste erlaubt es mehrere Restaurants gleichzeitig auszuwählen und nach diesen zu suchen. Dabei werden die ausgewählten Restaurants in eine ArrayList gespeichert und danach in die Datenbank transferiert.

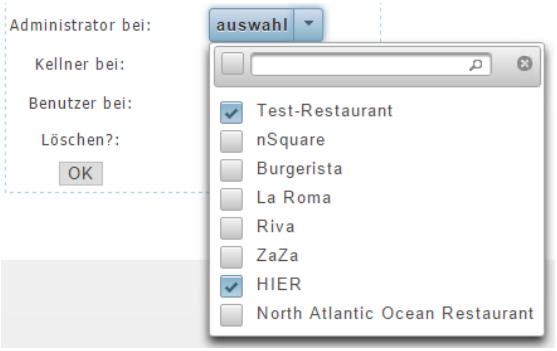


Abbildung 14: Restaurantzuordnung

3.3.2.2.5 Produkt- und Tagesmenüverwaltung

Die Produktverwaltung bietet die Möglichkeit neue Produkte für Restaurants zu erstellen. Mittels der Tagesmenüverwaltung können diese einem Tagesmenü zugeordnet werden.

3.3.2.2.6 Werbungsverwaltung

Mit der Werbungsverwaltung kann der Restaurant-Administrator benutzerspezifische Werbung für Kunden seines Restaurants generieren.

Dieser gibt hierfür die gewünschten Parameter ein. Die Werbung ist im Anschluss auf den Smartphones der Gäste zu sehen.

Für nähere Information über die Werbungserstellung siehe Abschnitt 4.5.3.

3.3.2.2.7 Statistiken

Für die Auswertung von Produktverkäufen muss der Restaurant-Administrator einen Zeitraum und ein Produkt auswählen. Die Auswertung erfolgt in einem Balkendiagramm nach verkauften Tagesmenüs oder Produkten pro Tag.

Für die Auswahl des Datums und die Anzeige des Diagramms wurde PrimeFaces verwendet.

```
<p:calendar value="#{productQRCodeBean.date}" locale="de"
navigator="true" pattern="dd.MM.yyyy"/>
```

In diesem Beispiel ist zu sehen wie der PrimeFaces Kalender implementiert wurde, um ein Datum auszuwählen.

```
<p:chart type="bar" model="#{productQRCodeBean.barModel}"
style="height:300px" responsive="true" />
```

Hier sind die Implementierungen des PrimeFaces BarCharts in JSF (Beispiel oben) und die entsprechende *ProductQRCodeBean-Methode* (Beispiel unten) dargestellt.

```
public BarChartModel getBarModel() {
        BarChartModel model = new BarChartModel();
        ChartSeries products = new ChartSeries();
        products.setLabel("Produk"); //Überschrift
        for(int i=0; i<sortDates.size(); i++) {
            DateFormat df = new SimpleDateFormat("EE, dd.MM.yyyy");
            products.set(df.format(sortDates.get(i).getDate()),
                        sortDates.get(i).getProducts()); //Balkenwerte
        }
        model.addSeries(products);
        barModel = model;
        barModel.setTitle(list.get(0).getProductQrcode().
                          getProduct().getProductName()
                          + " - "+list.get(0).getProductQrcode()
                          .getProduct().getRestaurant()
                          .getRestaurantName()); //Titel
        Axis xAxis = barModel.getAxis(AxisType.X);
        xAxis.setLabel("Produkt"); //x-Achsen Beschriftung
        Axis yAxis = barModel.getAxis(AxisType.Y);
        yAxis.setLabel("Anzahl"); //y-Achsen Beschriftung
        yAxis.setMin(0);
        yAxis.setMax(Collections.max(sortDates,
                    new Comparator<SortDates>(){
              @Override public int compare(SortDates p1, SortDates p2) {
                  return p1.getProducts()- p2.getProducts();
              }
        }).getProducts()+1); //Maximale Höhe des Diagramms
        return barModel;
}
```

3.3.2.3 Testen

Schon während der Entwicklung sind sämtliche Funktionalitäten getestet und verbessert worden. Zum Abschluss der Diplomarbeit wurden nochmals alle Anwendungsfälle durchgegangen und die Fehler dokumentiert. Diese Fehler wurde ausgebessert und das System erneut getestet. Auch die Performance wurde laufend verbessert.

Ein weiterer Punkt war die Verbesserung des Verschlüsselungsverfahrens für Passwörter. Da das vorher gewählte Verfahren MD5 als nicht mehr sicher angesehen wird, wurde das PBKDF2WithHmacSHA1 Verschlüsselungsverfahren eingesetzt.

4 Funktionsumfang

4.1 Funktionalität des Grundsystems

In der folgenden Grafik ist das Grundsystem und der Ablauf der Diplomarbeit mit allen Benutzergruppen vereinfacht dargestellt.

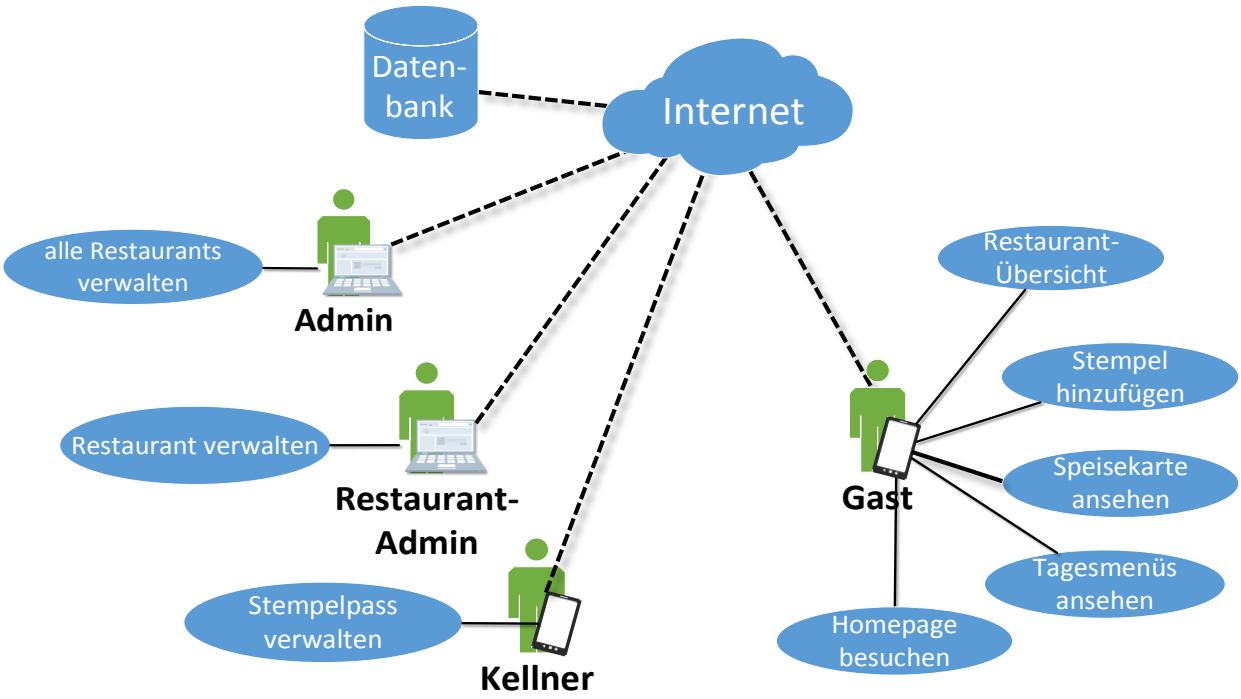


Abbildung 15: Use-Case-Diagramm

4.1.1 Funktionalität der mobilen Applikation

4.1.1.1 Übersicht aller Restaurants

Unmittelbar nach dem Start der mobilen Applikation wird eine Landkarte angezeigt auf der Restaurants abgebildet sind.

4.1.1.2 Einleitung einsehen

Sollte sich ein neuer Benutzer nicht mit der Handhabung der mobilen Anwendung auskennen, so kann sich dieser eine Einleitung ansehen. In dieser wird die Bedienung der App einfach erklärt und eine Übersicht aller Funktionen gegeben.

4.1.1.3 QR-Code einscannen

Jeder Benutzer hat die Möglichkeit einen QR-Code einzuscannen um einen Stempel zu dessen Stempelpass hinzuzufügen. Kellner können den QR-Code von den Stempelpässen der Kunden einlesen, auf Vollständigkeit prüfen und zurückzusetzen.

4.1.1.4 Speisekarte ansehen

Nachdem der Benutzer auf ein Restaurant geklickt hat, stehen im diverse Funktionalitäten zur Verfügung. Eine dieser Funktionalitäten ist es die Speisekarte des jeweiligen Restaurants einzusehen. Dabei werden alle Produkte mit dem entsprechenden Preis in einer übersichtlichen Liste angezeigt. Außerdem hat ein Benutzer die Möglichkeit durch einen Klick auf das jeweilige Produkt weitere Informationen, wie beispielsweise eine Beschreibung sowie die enthaltenen Allergene, einzusehen.

4.1.1.5 Tagesmenüs ansehen

Nachdem der Benutzer auf ein Restaurant geklickt hat, kann er außerdem dessen Tagesmenüs einsehen. Dabei wird eine Liste aller Tagesmenüs des jeweiligen Restaurants mit deren Preisen angezeigt. Falls sich ein Benutzer für die enthaltenen Produkte eines Tagesmenüs interessiert, so kann dieser einfach auf das entsprechende Tagesmenü klicken und erhält eine Auflistung der enthaltenen Speisen und Getränke.

4.1.1.6 Stempelpass ansehen

Eine weitere Funktionalität, welche dem Benutzer nach dem Klick auf ein Restaurant zur Verfügung steht, ist es seinen virtuellen Stempelpass einzusehen. Der Kunde hat auch die Möglichkeit, nach der Bestellung einer Speise, einen Stempel zu seinem Stempelpass hinzuzufügen. Sobald ein Kunde seinen Menüpass vollständig aufgefüllt hat kann dieser, wie bei einem realen Menüpass, eine kostenlose Speise bestellen. Im Anschluss wird der Stempelpass von einem Kellner zurückgesetzt.

4.1.1.7 Homepage besuchen

Ein Benutzer hat nach dem Klick auf ein Restaurant außerdem die Möglichkeit dessen Homepage zu besuchen.

4.1.1.8 Stempelpass verwalten

Ein Kellner kann den Stempelpass eines Kunden einlesen und überprüfen ob dieser vollständig gefüllt ist. Außerdem kann er den eingescannten Stempelpass zurücksetzen, falls der Kunde beispielsweise eine kostenlose Speise erhalten hat.

4.1.1.9 Tagesmenüs verwalten

Ein Restaurant-Administrator hat die Möglichkeit die Tagesmenüs seines Restaurants zu verwalten. Dabei wird eine Liste aller im Restaurant existierenden Tagesmenüs angezeigt. Sobald der Restaurant-Administrator auf eines dieser Tagesmenüs klickt, kann er den Preis, die beinhalteten Getränke, Speisen sowie den Wochentag des jeweiligen Tagesmenüs verwalten. Darüber hinaus kann er auch ein neues Tagesmenü erstellen.

Für Informationen über die Implementierung der einzelnen Funktionen siehe Abschnitt 3.3.1.3.

4.1.2 Funktionalität der Webapplikation

4.1.2.1 Funktionalität aus der Sicht eines Benutzers und eines Kellners

Ein Benutzer und ein Kellner haben keinen Zugriff auf die Webapplikation, da diese nur dem Administrator und den Restaurant-Administratoren für Verwaltungstätigkeiten zur Verfügung stehen.

4.1.2.2 Funktionalität aus der Sicht eines Restaurant-Administrators

Ein Restaurant-Administrator hat die Möglichkeit die ihm zugeteilten Restaurants, Produkte und Benutzer zu verwalten. Außerdem kann er für diese Restaurants Werbung erzeugen und Statistiken über verkaufte Produkte ansehen.

4.1.2.2.1 Restaurantverwaltung

Die Restaurantverwaltung dient dem Restaurant-Administrator, die ihm zugeteilten Restaurants zu administrieren. Dieser kann seine Restaurants ansehen und bearbeiten, jedoch keine neuen hinzufügen.

Es können sämtliche eingetragenen Informationen der Restaurants verändert werden, sowie beispielsweise der Restaurantname, das Logo und die geografische Position über Google Maps.

4.1.2.2.2 Benutzerverwaltung

Der Restaurant-Administrator hat die Möglichkeit sämtliche Benutzer zu verwalten, die einem seiner Restaurants zugeteilt sind.

Es besteht die Möglichkeit alle zugeteilten Kellner und Gäste, samt deren gespeicherten Informationen, anzusehen und die Verknüpfung zu dem jeweiligen Restaurant zu löschen. Außerdem können, mittels der Webapplikation, neue Kellner und Gäste hinzugefügt und einem Restaurant zugeteilt werden.

Ein Benutzer kann mehreren Benutzergruppen und Restaurants zugeteilt sein.

4.1.2.2.3 Produktverwaltung

Ein Restaurant-Administrator kann Produkte für seine Restaurants hinzufügen sowie Tagesmenüs erstellen und diesen Produkten zuweisen.

Für die Erstellung eines neuen Produktes gibt der Restaurant-Administrator sämtliche benötigten Daten an, beispielsweise den Namen, ein Produktbild und den Preis. Anschließend gibt er die enthalten Allergene an.

Tagesmenüs können verschiedenen Produkten, einem Wochentag sowie einem Restaurant zugeordnet werden.

4.1.2.2.4 Werbung erzeugen

Ein Restaurant-Administrator kann über die Weboberfläche benutzerspezifische Werbung erstellen. Die Werbung erscheint auf den entsprechenden Smartphones der Benutzer als Benachrichtigung.

Dazu gibt er an ob Werbung über ein Produkt oder Tagesmenü erstellt werden soll. Anschließend werden mittels Parametern diverse Informationen eingegeben. Zuletzt wird noch angegeben, welche Nachricht die Werbung enthalten soll und in welchem Zeitraum sie angezeigt wird.

4.1.2.2.5 Statistiken ansehen

Ein Restaurant-Administrator kann Statistiken über die Anzahl gekaufter Speisen und Getränke in einem angegebenen Zeitraum ansehen und auswerten. Dazu wird ein Diagramm erzeugt auf dem zu sehen ist, welche Produkte oder Tagesmenüs an welchen Tagen wie oft verkauft wurden (siehe Abbildung 2).

4.1.2.3 Funktionalität aus der Sicht eines Administrators

Ein Administrator hat über die Webapplikation die Möglichkeit sämtliche Restaurants und Benutzer ohne Beschränkungen zu verwalten.

4.1.2.3.1 Restaurants verwalten

Der Administrator kann über die Weboberfläche Restaurants hinzufügen, löschen sowie bearbeiten.

4.1.2.3.2 Benutzer verwalten

Der Administrator hat die Möglichkeit Benutzer jeglicher Art zu erstellen, zu bearbeiten und zu löschen.

Für Informationen über die Implementierung der einzelnen Funktionen siehe Abschnitt 3.3.2.2.

4.2 Funktionalität der grafischen Benutzeroberfläche

4.2.1 Mobile Applikation

4.2.1.1 Material Design

In diesen Abschnitt werden die neuen Designrichtlinien für Android Applikationen, dem sogenannten „Material Design“ erläutert.

4.2.1.1.1 Allgemeines

Mit der Android Version 5.0 Lollipop, welche am 3. November 2014 veröffentlicht wurde, kam es für Entwickler nicht nur zu einer Erneuerung der Programmierschnittstelle, sondern auch zu einer völligen Auffrischung der Designrichtlinien.

4.2.1.1.2 Wesentliche Designänderungen

Der allgemeine Hintergrund dieser neuen Designrichtlinien war es die Grenze zwischen den mobilen Applikationen und der realen Welt zu verringern.

In den folgenden Absätzen werden nun die wesentlichen Änderungen der Designrichtlinien veranschaulicht.

4.2.1.1.3 3D Design

Vor der Änderung der Designrichtlinien waren die Elemente einer Android Applikation auf derselben Ebene. Dies entspricht allerdings nicht der realen Welt, wenn man beispielsweise an einem Schreibtisch sitzt und darauf ein Block liegt, kann man klar erkennen, dass der Block höher liegt als der Tisch. Die neuen Designrichtlinien verlangen es hingegen ein 3D Design in Android Apps zu verwenden, da dieses Design wesentlich realistischer als das alte Flache Design wirkt.

Mit Hilfe der folgenden beiden Grafiken soll der Unterschied zwischen dem neuen und dem alten Design veranschaulicht werden.

Bis Android 5.0 Flaches Design	Ab Android 5.0 3D Design

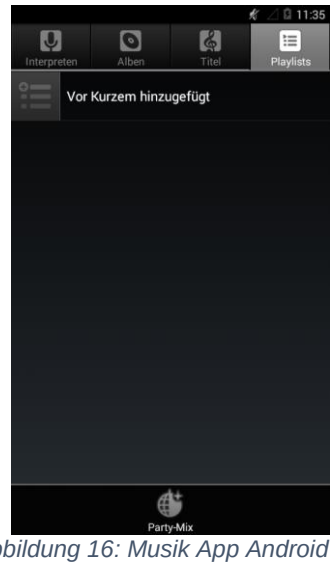
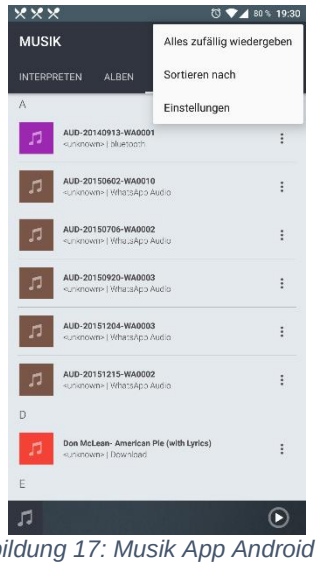

	

Abbildung 16: Musik App Android 4.4	*Abbildung 17: Musik App Android 5.0*

Vergleicht man die beiden Bilder, ist klar erkennbar, dass beim neuen 3D Design das Menü eine Ebene über den anderen Elementen der App liegt, während sich beim alten Flachen Design alles auf einer Ebene befindet.

Um einen solchen Ebenen-Effekt zu realisieren, wurden die Koordinaten aller Designelemente, neben den ursprünglichen Positionskoordinaten x und y, um eine z Koordinate, der Ebenen-Koordinate, erweitert.

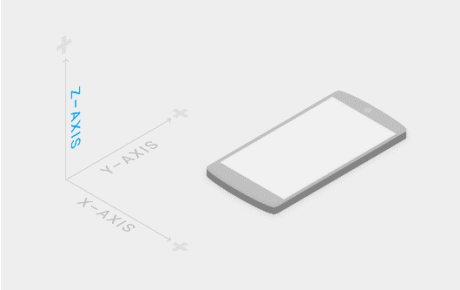
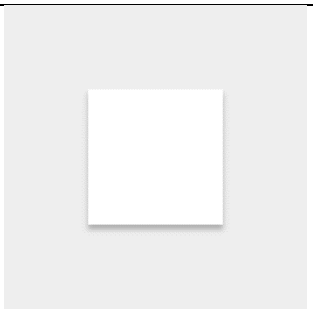


Abbildung 18: neue Z Koordinate

Um diesen Ebenen-Effekt zusätzlich variieren zu können, ist es möglich verschiedene Lichtquellen zu einer App hinzuzufügen. Durch die Veränderung einer Lichtquelle ändern sich auch die Positionierungen und die Größen der Schatten aller Elemente. Dies wird in den folgenden Grafiken veranschaulicht.

Auswirkung:	 *Abbildung 19: Hauptlicht*	*Abbildung 20: Umgebungslicht*	 *Abbildung 21: Hauptlicht und Umgebungslicht*
Lichtquelle(n):	Hauptlicht	Umgebungslicht	Hauptlicht und Umgebungslicht

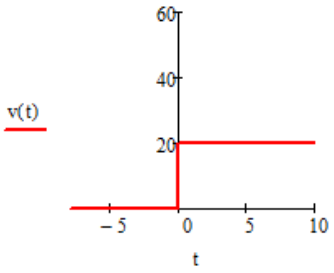
4.2.1.1.4 Bewegung von Elementen

Eine weitere wesentliche Änderung, welche das Material Design mit sich brachte, ist, dass Elemente eine bestimmte Masse haben. Aufgrund dessen kann nun eine natürliche Beschleunigung für Elemente berechnet werden, welche im Gegensatz zur alten Beschleunigung sehr realistisch ist.

In der nachfolgenden Abbildung wird mathematisch argumentiert, warum die neue Beschleunigung natürlicher als die alte Beschleunigung wirkt.

Künstliche Beschleunigung bis Android 4.4 bzw. 4.W

$$v(t) := \begin{cases} 20 & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$ //Geschwindigkeit



//Der Button wird zum Zeitpunkt t=0 gedrückt.

Hierbei hat ein Element immer eine konstante Beschleunigung.
Diese Funktion bildet beispielsweise das Bewegen eines Buttons um 20 Pixel pro Sekunde ab.

Im nächsten Schritt wird überprüft, ob solch ein Verhalten auch in der Natur auftreten kann. Dafür muss man zuerst die Geschwindigkeitsfunktion ableiten, um die Beschleunigungsfunktion zu erhalten.

$$\frac{d}{dt}v(t) \rightarrow$$

Dies ist allerdings nicht möglich, da die Ableitung der Geschwindigkeitsfunktion zum Zeitpunkt t=0 unendlich ist. Somit kann man sagen, dass solch eine Bewegung von Elementen in der Natur nicht möglich ist.

Natürliche Beschleunigung ab Android 5.0

Hier wird nicht wie bei der künstlichen Beschleunigung sofort eine Funktionsgleichung für die Geschwindigkeit aufgestellt. Es wird stattdessen die Beschleunigung eines Elements berechnet, und anhand dessen die Geschwindigkeitsfunktion ermittelt.

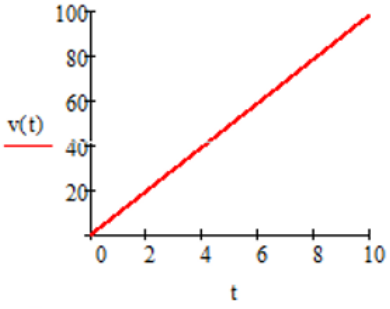
$m := 1kg$ //Masse

$a := g$ //Beschleunigung

$F := m \cdot a = 9.807\,N$ //Kraft

$a(t) := g$ //Beschleunigungsfunktion

$$v(t) := \int a(t)\,dt \rightarrow g \cdot t$$



Diese Geschwindigkeitsfunktion ist zwar auch in der Natur möglich, doch die Beschleunigung der Anziehungskraft kommt der Beschleunigung beim Bewegen eines Gegenstands durch menschlicher Krafteinwirkung nicht gleich. Dieser Krafteinwirkung kommen hingegen die sogenannten Ease-Funktionen sehr nahe. Diese werden sowohl in den neuen Android Versionen als auch in den neuen iOS Versionen für das Bewegen von Elementen verwendet.

Abbildung 22: Mathematische Argumentation für die neue Beschleunigungsfunktion

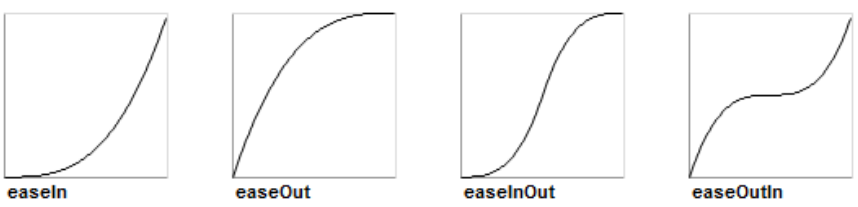


Abbildung 23: Verschiedene Ease-Funktionen

4.2.2 Webapplikation

Im Folgenden werden einzelne Designelemente der Webapplikation beschrieben.

4.2.2.1 Hintergrund

Für die Farbe des Headers, als auch für die Hintergrundfarbe, wurde ein sogenannter Gradient-Hintergrund ausgewählt, welcher einen Farbübergang hat. Um den CSS-Code für den Farbübergang zu generieren, wurde ein CSS Gradient-Generator verwendet.

Abbildung 24: Beispiel eines Gradient-Generators

4.2.2.1.1 Beispiel

In dem unteren Codebeispiel ist zu sehen, dass für jeden Browser ein eigener Code zur Darstellung von Gradient-Hintergründen generiert wird, da die meisten Internetbrowser unterschiedliche Standards zur Darstellung dieser haben.

```
background: -webkit-gradient(linear, left top, left bottom, color-
stop(0%, rgba(255,255,255,1)), color-stop(1%, rgba(255,255,255,1)),
color-stop(100%, rgba(237,237,237,1))); /* safari4+,chrome */
background: -o-linear-gradient(90deg, rgba(237,237,237,1) 0%,
rgba(255,255,255,1) 99%, rgba(255,255,255,1) 100%); /* opera 11.10+ */
background: -ms-linear-gradient(90deg, rgba(237,237,237,1) 0%,
rgba(255,255,255,1) 99%, rgba(255,255,255,1) 100%); /* ie10+ */
```

4.2.2.2 Facelets Templates

Für das einheitliche Design des Headers der Weboberfläche wurde ein sogenanntes Facelet Template verwendet. Facelets Templates ermöglichen in JSF das einfache Implementieren und Erweitern der Benutzeroberfläche.

Dazu wird ein Grunddesign, das sogenannte Template, in einer XHTML Datei festgelegt und im Programmcode eingebunden. Dies hat den Vorteil, dass Codeduplizierung vermieden wird und auf jeder Seite das gleiche Design und der gleiche Inhalt übernommen wird.

Im Falle dieser Diplomarbeit wurden Facelets Templates für das Design des Headers eingesetzt. Der Header ist auf jeder Seite der Webapplikation ersichtlich. Dieser bietet die Möglichkeit den Benutzernamen des aktuell eingeloggten Benutzers zu sehen und sich von jeder Stelle der Webapplikation aus abzumelden.

Abbildung 25: Header Webapplikation

4.2.2.2.1 Aufbau eines Facelet Templates

Ein Facelet Template besteht aus einer Template-Datei und mehreren Content-Dateien für die unterschiedlichen Abschnitte einer Website (bspw. Header, Content und Footer).

In der Template-Datei wird das grundlegende Layout festgelegt indem beispielsweise CSS-Dateien eingebunden werden. Mittels *ui:insert* wird ein Designbereich definiert und mit den *ui:include* Tags wird ein Standartinhalt der entsprechenden Content-Datei eingebunden. Der Inhalt eines Bereiches kann in den Dateien überschrieben werden, wo das Template eingesetzt wird.

Template-Datei

```
<h:head>
       <link rel="stylesheet" type="text/css"
       href="./resources/css/stylesheet.css" />
</h:head>
<h:body>
      <div id="page">
            <div id="header">
                  <ui:insert name="header">
                        <ui:include src="commonHeader.xhtml"/>
                  </ui:insert>
            </div>
            <div id="content">
                  <ui:insert name="content">
                        <ui:include src="commonContent.xhtml" />
                  </ui:insert>
            </div>
      </div>
</h:body>
```

Content-Datei (Header)

```
<h:head>
</h:head>
<h:body>
	<ui:composition>
		<div class="header-background">
		<p>
			<h:form>
				<b>Gastro Menü</b>
				<h:commandButton action="#{userBean.logout()}"
				value="Logout" class="leftHeaderElements"/>
				<h:outputLabel class="leftHeaderElements high
				lightUsername" value="#{userBean.getSession-
				Name()}"/>
				<b class="leftHeaderElements">Willkommen, </b>
			</h:form>
		</p>
		</div>
	</ui:composition>
</h:body>
```

Das Template wird mittels *ui:compostion* eingebunden. Alles was innerhalb dieser Tags ist, wird in dem entrechtenden Design des Templates formatiert. Falls der Standartinhalt der Content-Datei überschrieben werden soll, muss das *ui:define* Tag verwendet werden.

Datei in der das Template eingebunden wird

```
<ui:composition template="templates/commonLayout.xhtml">
	<ui:define name="content">
		<h1>Übersicht</h1>
		<h:form>
			<li><h:commandLink action="manageRestaurant?faces-redi-
			rect=true" value="Restaurants verwalten" /></li>
		</h:form>
	</ui:define>
</ui:composition>
```

4.2.2.3 CSS Template

Für das Design des Inhalts wurde ein CSS Template verwendet. Einzelne Elemente werden formatiert, indem entweder eine class definiert wird oder direkt das Design einzelner Tags verändert wird (z.B. alle <a>-Tags).

4.2.2.3.1 Ausschnitt aus einem CSS-Template

In dem untenstehenden Beispiel ist ein Ausschnitt aus einem CSS-Template zu sehen. Hierbei wird allen Elementen denen die class *content* zugeordnet ist oder allen *table* Elementen ein Design zugewiesen.

```
#content {
	padding: 30px; /*Abstand von 30 Pixel*/
	background: #fff; /*weißer Hintergrund*/
	margin: auto; /*automatischer Abstand nach außen*/}
table {
	margin-left: auto;
	margin-right: auto;
	border:1px dashed lightblue;
	table-layout:fixed;
}
```

4.2.2.4 Tabellen

Für die Darstellung einer HTML-Tabelle wurde in JSF eine datatable verwendet, die mit den entsprechenden Daten befüllt wurde.

Zur besseren Darstellung auf Bildschirmen mit einer niedrigeren Auflösung, wurde in CSS eine sich automatisch verkleinernde Tabelle programmiert.

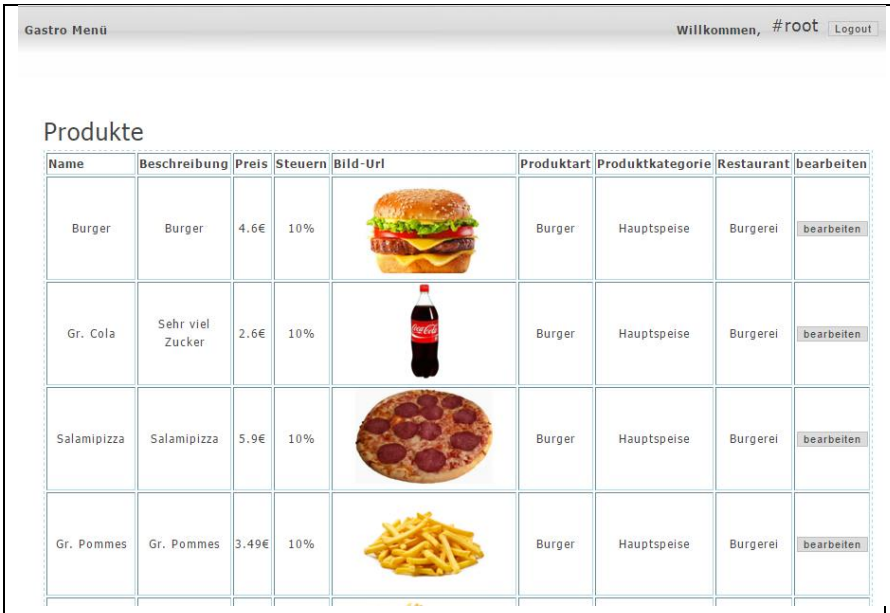


Abbildung 26: Tabellen auf Bildschirmen mit höhrere Auflösung

Abbildung 27: Tabelle auf Bildschrimen mit geringerer Auflösung

4.2.2.5 Anzeige von Bildern

Zur richtigen Anzeige der Bilder wurde die EX[9] verwendet.

```
<h:graphicImage value="#{product.productImageurl == '' ? 'https://up-
load.wikimedia.org/wikipedia/commons/thumb/a/ac/No_image_availa-
ble.svg/300px-No_image_available.svg.png' : fn:replace(product.produc-
tImageurl, '%@', 'localhost')}" height="100px"/>
```

Das Beispiel stellt dar wie ein Bild eingefügt wird. Wenn die Bild-URL leer ist, wird ein Standartbild angezeigt. Falls ein Link vorhanden ist, wird geprüft ob sich der Teilstring %@ im String befindet und dieser durch die richtige Bildadresse ersetzt.

[9] EX: Expression Language, kann innerhalb von JSF logische Funktionen verwirklichen.

4.3 Funktionalität des QR Code Scanners

In diesem Abschnitt wird die Funktionalität des QR-Code[10] Scanners der mobilen Applikation veranschaulicht.

4.3.1 Allgemeines zu QR-Codes

Ein QR-Code ist eine Weiterentwicklung des Barcodes[11]. Der Barcode wird beispielsweise für die Abbildung der EAN[12] auf Produkten verwendet. Dieser ist im Gegensatz zum QR-Code nicht eindimensional sondern zweidimensional (siehe Abbildung 28 und Abbildung 29). Die Hauptaufgabe von QR-Codes ist die Abbildung von größeren Informationsmengen.

Abbildung 28: Barcode

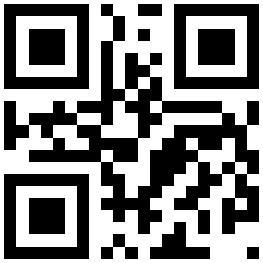


Abbildung 29: QR-Code

4.3.2 Allgemeines zur Problemstellung

Da die mobile Applikation einen virtuellen Stempelpass beinhaltet, musste überlegt werden, wie ein Kellner einen Stempel zu dem Stempelpass eines Kunden hinzufügt.

Die Lösung dieses Problems war, dass der Kunde nach dem Kauf einer Speise einen QR-Code von einem Kellner erhält. Der Kunde kann diesen QR-Code mit der mobilen Applikation einscannen und somit einen Stempel zu seinem virtuellen Stempelpass hinzufügen. Bereits einscannten QR-Codes können kein zweites Mal an einem Tag erfasst werden.

[10] QR-Code: Quick Response Code
[11] Barcode: Strichcode
[12] EAN: European Article Number ist eine Nummer welche alle Artikel innerhalb Europas eindeutig identifizierbar macht.

4.3.3 Einlesen eines QR-Codes

In dem folgenden Abschnitt wird anhand eines Beispiels veranschaulicht, wie QR-Codes gelesen werden.

Abbildung 30: Beispiel QR-Code

Generell enthält jeder QR-Code drei Position Patterns. Diese werden benötigt um einen QR-Code auch dann korrekt einlesen zu können, wenn dieser nicht richtig positioniert ist. Zwischen den einzelnen Position Patterns befinden sich die Timing Patterns. Diese bestehen aus umso mehr Pixel, je größer ein QR-Code ist. Daher werden sie auch verwendet, um die Größe eines QR-Codes zu ermitteln. Alignment Patterns werden nur bei größeren QR-Codes benötigt, um den Vorgang des Einscannens zu erleichtern. Aus wie vielen Alignment Patterns ein QR-Code besteht, ist wiederum von dessen Größe abhängig.

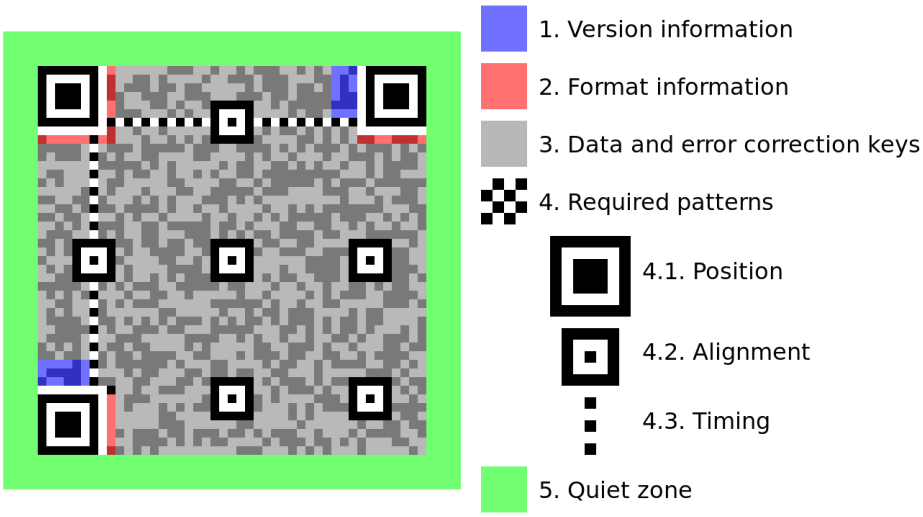


Abbildung 31: QR-Code Unterteilung

Zu Beginn müssen alle Position Patterns des QR-Codes gesucht werden, um zu ermitteln ob der QR-Code korrekt positioniert ist.

Abbildung 32: QR-Code, Ermittlung der Position Patterns

In der vorhergehenden Abbildung kann man erkennen, dass der QR-Code drei Position Patterns beinhaltet. Um feststellen zu können, dass ein QR-Code korrekt positioniert ist, muss sich ein Position Pattern in der linken oberen Ecke, ein weiteres in der rechten oberen Ecke sowie eines in der linken unteren Ecke befinden.

Da in diesem QR-Code kein Alignment Pattern existiert, muss der QR-Code auch nicht hinsichtlich dieses Patterns überprüft werden.

Abbildung 33: QR-Code, Ermittlung des Timing Pattern

In der oberen Abbildung kann man erkennen, dass das Timing Pattern in diesem QR-Code eine Länge von sieben Pixel aufweist. Da ein Position Pattern immer eine Länge von sieben Pixel aufweist, beträgt die Breite als auch die Länge dieses QR-Codes 21 Pixel.

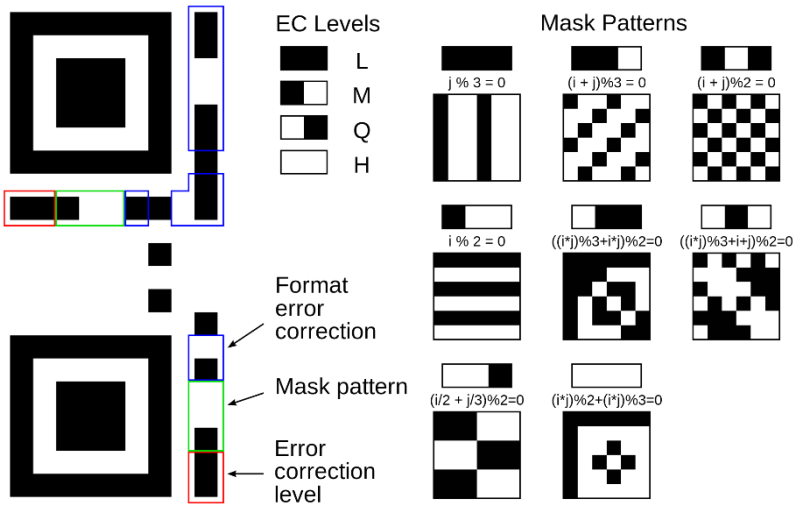


Abbildung 34: QR-Code, Format-Information sowie Mask Patterns

Neben den bereits erwähnten Patterns, enthält ein QR-Code auch eine Formatinformation. Dieses besteht aus einem Error correction level, einem Mask Pattern sowie der Format error correction. Das Error correction level sowie die Format error correction werden für die Behandlung von Lesefehlern verwendet.

Daher muss zunächst das Mask Pattern ermittelt werden.

Abbildung 35: QR-Code, Ermittlung des Mask Pattern

Generell ist die in QR-Codes beinhaltete Nachricht zu Beginn immer verschlüsselt. Um diese zu lesen, muss sie daher zuerst entschlüsselt werden. Das Mask Pattern gibt Auskunft darüber, mit welcher Vorgehensweise ein QR-Code entschlüsselt werden muss. In der Abbildung 34 sind alle existierenden Mask-Patterns sowie deren Entschlüsselungsvorgehensweisen dargestellt. Nach dem Ermitteln der Vorgehensweise, muss das daraus berechnete Entschlüsselungsmuster über den eigentlichen QR-Code gelegt werden.

Abbildung 36: QR-Code Entschlüsselungsmuster

In der vorhergehenden Abbildung ist zu sehen, wie das Entschlüsselungsmuster für diesen QR-Code aussieht. Es ist zu beachten, dass jeder Pixel, welcher sich auf dem Entschlüsselungsmuster befindet, umgekehrt werden muss, sodass ein schwarzer Pixel im Anschluss weiß ist und umgekehrt.

Abbildung 37: Entschlüsselter QR-Code

Nachdem alle notwendigen Pixel umgekehrt wurden, ergibt sich der entschlüsselte QR-Code.

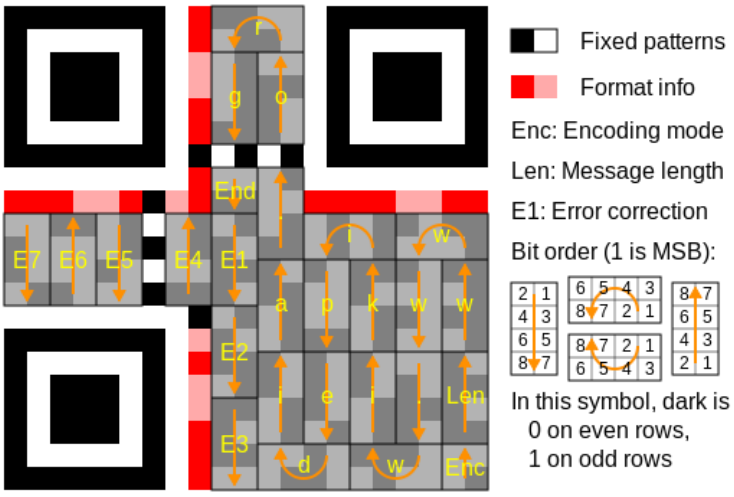


Abbildung 38: Lesevorgang bei entschlüsselten QR-Codes

Nachdem der QR-Code entschlüsselt ist, kann mit dem eigentlichen Lesevorgang begonnen werden. Man beginnt dabei von der unteren rechten Ecke. Dafür werden zunächst die ersten vier Pixel des QR-Codes und anschließend die nächsten acht Pixel in Pixelkästen gegeben (siehe Abbildung 39). Diese beiden Kästen stellen nun den Encoding Mode sowie die Message Length des QR-Codes dar. Der dargestellte QR-Code verwendet also den Encoding-Mode 4 und dessen Nachrichtenlänge beträgt drei Pixelkästen.

Abbildung 39: QR-Code Encoding Mode und Message Length

Encoding Mode	Bedeutung
0	Ende der Nachricht
1	Nummern (10 Bit für 3 Ziffern)
3	Nachricht ist auf mehreren QR-Codes verteilt
4	Byte-Codierung (8 Bit für ein Zeichen)
7	Alternative Codierung
8	Kanji Codierung (11 Bit für ein Zeichen)

Wie man der vorgehenden Tabelle entnehmen kann bedeutet Encoding Mode 4, dass die in diesem QR-Code enthaltene Nachricht eine Zeichenkette ist, welche in Byte Codierung gespeichert ist. Nun werden auch die nachfolgenden Pixel in Kästen zu je acht Bit eingeteilt, um die einzelnen Zeichen der Nachricht zu erhalten. Dabei muss beachtet werden, dass die Zeichenkette nur aus den nächsten drei Pixelkästen besteht, denn alle nachfolgenden Kästen sind für die Fehlerbehebung zuständig.

Abbildung 40: Einteilung der Daten in Pixelkästen

In der Tabelle sind die numerischen Werte der drei Pixelkästen dargestellt, welche die Nachricht des QR-Codes abbilden.

Pixelkasten	Numerischer Wert in Bit	Numerischer Wert
1	01100001	97
2	01110000	112
3	01110000	112

Diese numerischen Werte bilden jeweils ein ASCII[13]-Zeichen ab. Für die Umrechnung der Werte wird die nachfolgende ASCII-Tabelle verwendet.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Abbildung 41: ASCII Tabelle

Bei der Umwandlung der numerischen Werte auf ASCII-Zeichen wird der Wert 97 zu „a“ und die beiden Werte 112 werden zu „p“. Die in dem QR-Code beinhaltete Zeichenkette ist also „app“.

[13] ASCII: American Standard Code for Information Interchange, eine Zeichenkodierung bei welcher ein Zeichen auf 7 Bit abgebildet wird.

4.3.4 Realisierung in der mobilen Applikation

Wie anhand des vorhergehenden Beispiels zu erkennen ist, sind mehrere, voneinander abhängige Schritte notwendig, um einen QR-Code korrekt einzulesen. Alle diese Vorgänge selbst zu programmieren ist sehr zeitaufwändig. Außerdem wäre dies nicht wirklich sinnvoll, da bei der Entwicklung von Android sowie iOS Applikationen Bibliotheken zur Verfügung stehen bzw. direkt in die Programmiersprache integriert sind, welche das Einscannen von QR-Codes übernehmen.

Aufgrund dessen wurde entschieden in der iOS Anwendung die von der Programmiersprache mitgelieferte Bibliothek „AVFoundation“ zu verwenden, während in Android auf die Bibliothek „ZBar“ zurückgegriffen wurde.

In der nachfolgenden Abbildung wird veranschaulicht wie in den beiden mobilen Applikationen ein QR-Code eingelesen und verarbeitet wird.

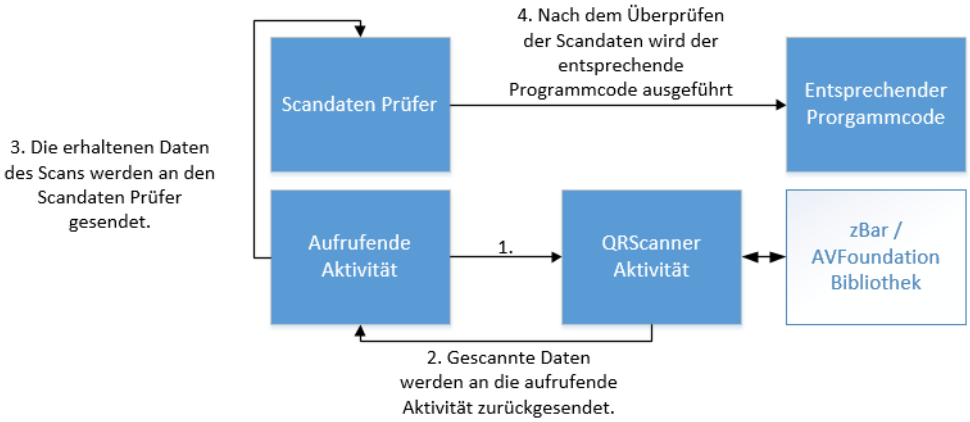


Abbildung 42: Ablauf des Einscannens eines QR-Codes

Wie man in der Abbildung erkennen kann, ruft ein Benutzer über eine Aktivität die QR-Scanner Aktivität auf, diese scannt mit Hilfe der ZBar bzw. AVFoundation Bibliothek die Daten eines QR-Codes ein. Im Anschluss werden diese Daten wieder an die aufrufende Aktivität zurückgesendet und dort an den Scandaten-Prüfer weitergeleitet, welcher die eingescannten Daten überprüft und den entsprechenden Programmcode ausführt.

Der Scandaten-Prüfer spielt daher in den mobilen Anwendungen eine wesentliche Rolle. In der nachfolgenden Tabelle ist aufgelistet, welche übergebenen Scandaten von dieser Komponente verarbeitet werden.

Scandaten	Beschreibung
add_stampcard;…	Der Benutzer erhält für das jeweilige Restaurant eine virtuelle Stempelkarte.
add_stamp;…	Ein Stempel wird zu einen bereits existierenden Stempelpass hinzugefügt. Falls der Kunde noch kein Stempelpass des jeweiligen Restaurants besitzt, wird dieser zusätzlich angelegt.
switch_to_waiter;…	Ein Kunde wird für das jeweilige Restaurant zu einen Kellner umgewandelt.
switch_to_admin;…	Ein Kunde wird für das jeweilige Restaurant zu einen Restaurantadministrator umgewandelt.

stampcard;…	Eine Stempelkarte wird eingelesen um sie im Anschluss zu verwalten. Dies funktioniert allerdings nur wenn der einscannende Benutzer ein Kellner des jeweiligen Restaurants ist.

4.3.5 Realisierung des QR-Code Generators in der Webapplikation

Zur Generierung des QR-Codes wurde in der Webapplikation die Google Chart API verwendet. Dazu ist es notwendig den Link https://chart.googleapis.com/chart? mit den entsprechenden Parametern aufzurufen. Dieser liefert einen QR-Code in Form einer PNG Datei zurück, welche in einer XHTML Datei mittels *<h:graphicImage/>* angezeigt werden kann.

Parameter:

Parameter	Erforderlich	Beschreibung
cht=qr	Ja	Es wird ein QR-Code erzeugt.
chs=<Breite>x<Höhe>	Ja	Die Größe des zu erzeugenden QR-Codes.
chl=<Text>	Ja	Der Text der in einen QR-Code umgewandelt werden soll.
choe=<Kodierung>	Nein	• UTF-8 (Standard) • Shift_JIS • ISO-8859-1
chld=<Fehlerkorrektur-stufe>| <Rand>	Nein	L - Ermöglicht die Wiederherstellung von bis zu 7% Datenverlust. (Standard) M - Ermöglicht die Wiederherstellung von bis zu 15% Datenverlust. Q - Ermöglicht die Wiederherstellung von bis zu 25% Datenverlust. H – Ermöglicht die Wiederherstellung von bis zu 30% Datenverlust. Die Breite des weißen Randes um den QR-Code.

4.3.5.1 Beispiel:

```
<h:graphicImage value='https://chart.google-
apis.com/chart?chs=300x300&amp;cht=qr&amp;chl=Hello%20world&amp;choe=UTF-
8'/>
```

In diesem Beispiel ist zu sehen, dass ein QR-Code mit dem Wert „Hello World“ erzeugt und angezeigt wird.

4.4 ORM Tool

4.4.1 Allgemeines

Da das Datenbankmodell dieser Diplomarbeit mittels eines model-first Ansatzes erstellt wurde (siehe 3.3.1.2.4.1), wird ein ORM Tool[14] benötigt, welches die Tabellen der Datenbank auf die Objekte der jeweils verwendeten Programmiersprachen abbildet.

Die Wahl des ORM Tools für die Webapplikation fiel auf die Schnittstelle JPA.

Bei der Wahl des ORM Tools für die Android Applikation sowie für die iOS Applikation war es von oberster Priorität, dass die Datenübertragung über XML oder JSON abläuft. Obwohl mehrere kostenlose ORM Tools für diese Plattformen existierten, unterstütze keine davon eine Datenübertragung mittels XML bzw. JSON. Aufgrund dessen wurde entschieden selbst ein ORM Tool für unsere mobilen Applikationen zu entwickeln.

4.4.2 Selbst entwickeltes ORM Tool

Bei der Entwicklung eines eigenen ORM Tools für die beiden mobilen Applikation wurde nach den empfohlenen Richtlinien der jeweiligen Plattformen vorgegangen. Beide Plattformen bestehen dabei darauf, dass die Datenübertragung entweder mittels XML oder JSON durchgeführt werden soll. Daher wurde als Programmiersprache für die Schnittstelle die Skriptsprache PHP verwendet. Die Aufgabe der Schnittstelle ist es, die in der Datenbank enthaltenen Daten in das Format XML zu bringen. Der Vorgang der Datenübertragung zwischen dem Webserver, welcher die PHP Schnittstelle bereitstellt und dem Mobiltelefon, welches darauf zugreift, ist in der nachfolgenden Abbildung dargestellt.

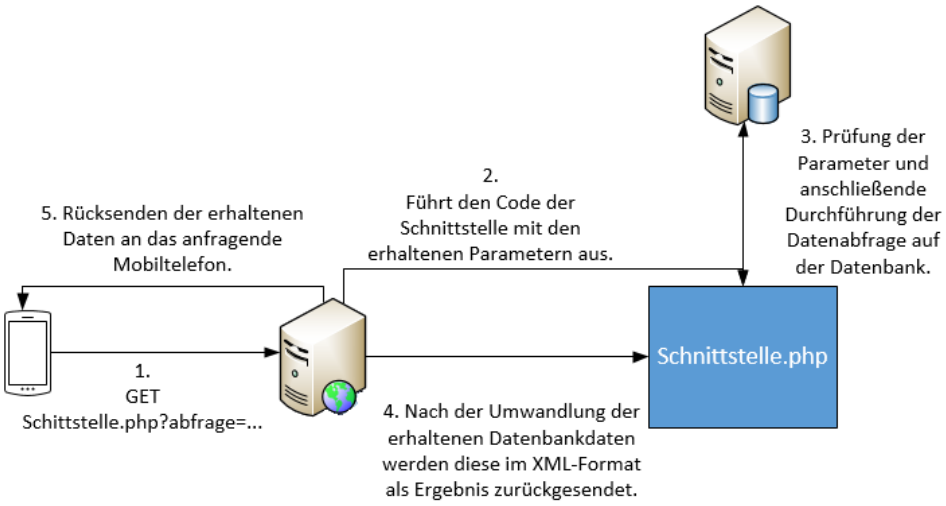


Abbildung 43: Datenanfrage eines Mobiltelefons

[14] ORM Tool: Ein Object Relational Mapping Tool realisiert die objektrelationale Abbildung von Datenbankobjekten in Objekten der jeweiligen Programmiersprache.

Quellcode der Schnittstelle:

```
<?php
	header('Content-Type: text/xml; charset=utf-8');
	$stmt = $_GET["statement"];
	$stmt = utf8_encode($stmt);  //otherwise umlauts are not possible
	$txt = $stmt;
	$logfile = file_put_contents('logs.txt', $txt.PHP_EOL ,
                                FILE_APPEND);
	executeStatement($stmt);

	function executeStatement($stmt) {
		$servername = "127.0.0.1";
		$username = "example";
		$password = "********";
		$schema = "mrasm";

		// Connect
		$db =  mysql_connect($servername, $username, $password) or
		       die("Keine Verbindung möglich: " . mysql_error());

		mysql_select_db($schema);

		if ((strpos(strtoupper($stmt), 'CALL')) !== false){
			$mysqli = new mysqli($servername, $username, $password,
                                $schema);
			$rs = $mysqli->query( $stmt );
			echo $mysqli->error;
			while($row = $rs->fetch_row())
			{
				$stmt = $row[0];
			}
		}
		if((strpos(strtoupper($stmt), 'INSERT')) !== false ||
		   (strpos(strtoupper($stmt), 'UPDATE')) !== false ||
		   (strpos(strtoupper($stmt), 'DELETE')) !== false){
			$result = mysql_query($stmt);
			echo "<query><row><col name=\"id\" value=\"" .
			     mysql_insert_id($db) . "\"/></row></query>";
		}
		else{
			$result = mysql_query($stmt);
			echo "<query>";
			if (gettype($result) == "resource"){
				while ($row = mysql_fetch_array($result,
				       MYSQL_NUM)) {
					$i=0;
					echo "<row>";
					while($i < count($row)){
						$meta = mysql_fetch_field($result,
                                                          $i);
						echo "<col name=\"";
						if ($i+1 < count($row));
						echo $meta->name . "\" value=\"";
						echo toXMLString($row[$i]);
						echo "\"/>";
						$i++;
					}
					echo "</row>";
				}
			}
			echo "</query>";
```

```
            }
        }

        function toXMLString( $str )
    {
            $str = str_replace("&", "@and", $str);
            $str = str_replace("=", "@equ", $str);
          return $str;
    }
?>
```

Nachdem die Kommunikation zwischen dem Smartphone und der Datenbank fehlerfrei funktionierte, wurde mit der Entwicklung einer eigenen Datenbankverbindung begonnen. Diese Datenbankverbindung erhält dabei die Abfragen, welche auf der Datenbank ausgeführt werden sollen. Im Anschluss werden die Daten der Datenbank in einfach anzusteuernden Objekten der jeweiligen Programmiersprache zurückgegeben. Dieser Ablauf wird in der nachfolgenden Grafik näher beschrieben.

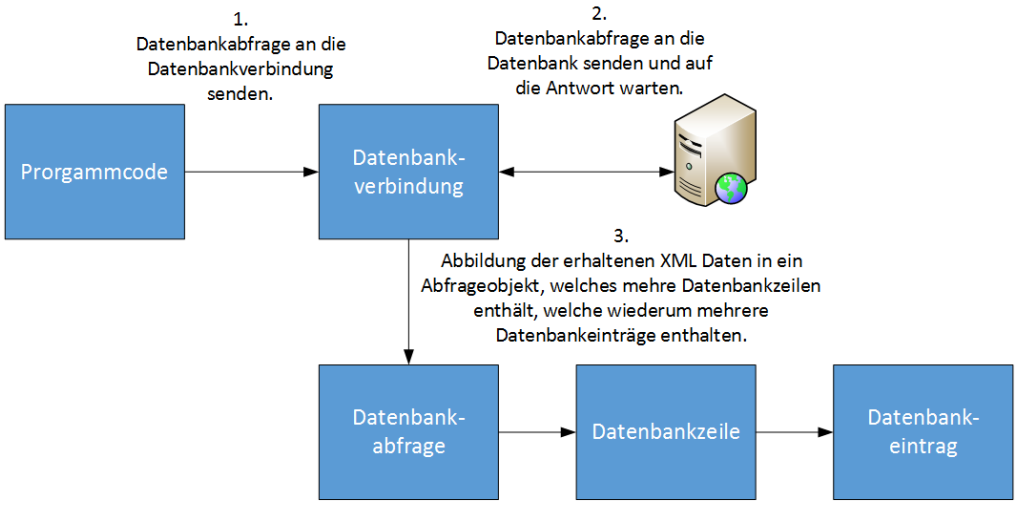


Abbildung 44: Abbildung der erhaltenen XML Daten in Objekte

Als die Entwicklung dieser Komponenten abgeschlossen war, wurde mit der Implementierung der typischen Funktion eines ORM Tools begonnen. Das Ziel dabei war eine Überklasse zu schreiben, welche auf die Variablen seiner Unterklassen zugreift und diese im Anschluss mit der in der Datenbank befindlichen Daten befüllt. Außerdem soll diese nicht nur die Read-Methode, sondern direkt alle CRUD[15]-Methoden realisieren, sodass diese nicht für jede Unterklasse einzeln implementiert werden müssen.

Bevor die Realisierung dieser Überklasse durchgeführt werden konnte, musste zuerst überleget werden wie man auf einzelne Variablen von erbenden Klassen zugreifen kann. Dabei erwies sich das Reflektion-Konzept[16] als besonders hilfreich, welches sowohl von Android als auch von iOS Applikation unterstützt wird.

[15] CRUD: Create, Read, Update und Delete Methoden der Datenbank
[16] Reflektion-Konzept: Ein Konzept welches es ermöglicht, dass ein Programm während seiner Laufzeit Änderungen an seinen Variablen, Funktionen oder Strukturen durchführt.

Auszug der Read-Funktion der Enitity Klasse:

```
public Entity (Class entityClass, String tableName, MySQLRow row){
    this(entityClass, tableName);
    Field[] fields = this.entityClass.getDeclaredFields();  //get all
                                                fields of the passed class
    for (int i = 0; i < fields.length; i++){
        if (fields[i].getType().equals(String.class)){  //check if
                                                    the field is a string
            String value = row.findEntryByName(fields[i].geName())
                                                .getValue();
            try{
                fields[i].setAccessible(true);
                fields[i].set(this, encodeSpecialChars(value));
            }
            catch (IllegalAccessException e){
                e.printStackTrace();
            }
        }

        if (fields[i].getType().equals(Integer.class) ||
            fields[i].getType().equals(int.class)){  //check if the
                                                  field is an integer
            int value = Integer.parseInt(row
                                        .findEntryByName(
                                         fields[i].getName())
                                        .getValue());
            try{
                fields[i].setAccessible(true);
                fields[i].set(this, value);
            }
            catch (Exception e){
                e.printStackTrace();
            }
        }

        if (fields[i].getType().equals(double.class)){  //check if
                                                the field is a double
            double value = Double.parseDouble(row
                                        .findEntryByName(
                                         fields[i]
                                        .getName()).getValue());
            try{
                fields[i].setAccessible(true);
                fields[i].set(this, value);
            }
            catch (Exception e){
                e.printStackTrace();
            }
        }

        if (fields[i].getType().equals(Calendar.class)){  //check if
                                                the field is a Calendar
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
                                                 HH:mm:ss", Locale.GERMAN);
            Calendar value = Calendar.getInstance();
            value.setTimeZone(TimeZone.getDefault());
            try{
                value.setTime(sdf.parse(row
                                        .findEntryByName(
                                         fields[i].getName())
                                        .getValue()));
```

```
                }
                catch (java.text.ParseException e){
                    e.printStackTrace();
                }
                try{
                    fields[i].setAccessible(true);
                    fields[i].set(this, value);
                }
                catch (Exception e){
                    e.printStackTrace();
                }
            }
        }
    }
```

Nach dem Fertigstellen und Testen der Entity Klasse wurde mit der Entwicklung der DAOs[17] begonnen. Diese DAOs ermöglichen einen einfachen Zugriff auf die Daten der Datenbank ohne dabei direkt Datenbankbefehle zu verwenden. Diese werden dabei in den Funktionen der DAOs abgewickelt. Im Programmcode werden diese Funktionen mit den entsprechenden Parametern aufgerufen.

Zur übersichtlichen Entwicklung der DAOs wurde eine abstrakte DAO Überklasse entwickelt, welche die Datenbankzugriffe abwickelt.

Quellcode der DAO Überklasse:

```
public abstract class Dao {  //The super class for all Daos
    //Variables
    protected MySQLConnection mySQLConnection = null;  //The mysql
                    connection which is needed to connect to the database

    //Constructor
    public Dao(MySQLConnection mySQLConnection) {
        this.mySQLConnection = mySQLConnection;  //Set the mysql
                    connection with the value of the passed mysqlconnection
    }

    //Setter and Getter

    //Operative Methods
    public ArrayList<MySQLRow> runMySQLQuery (String query) {
      //Sends the passed mysqlquery to the database and returns the
      response of the database
        return this.mySQLConnection.executeQuery(query);
    }
}
```

[17] DAOs: Data Access Objects bzw. Datenzugriffsobjekte, werden verwendet um in einer Applikation auf eine Datenbank zuzugreifen.

In der nachfolgenden Abbildung wird verdeutlicht wie all diese Komponenten zusammenarbeiten um eine Datenbankabfrage durchzuführen.

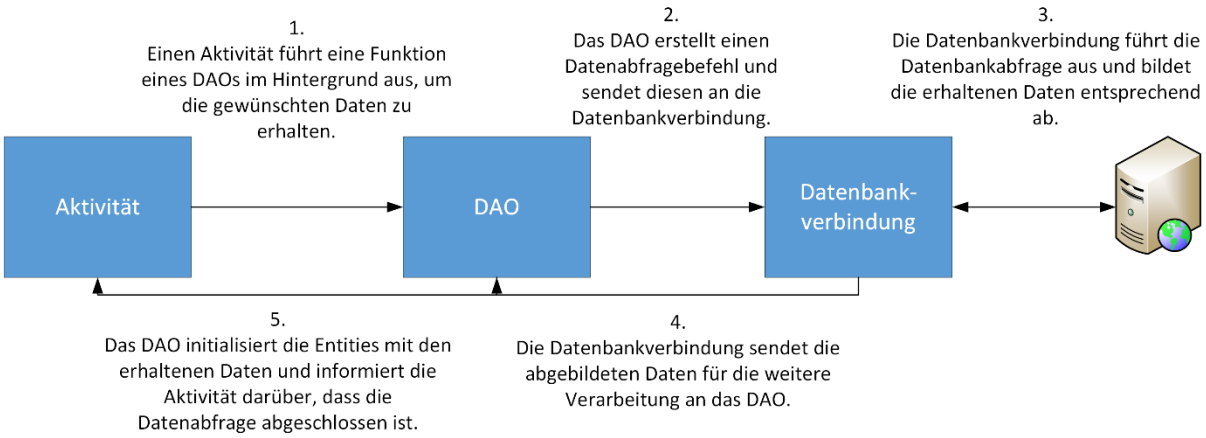


Abbildung 45: Komponenten der Datenbankabfrage

4.5 Benutzerspezifische Werbung

4.5.1 Allgemeines

Ein wesentlicher Nutzen der Diplomarbeit ist, dass Unternehmen möglichst einfach benutzerspezifische Werbung erstellen können. Diese Werbung wird auf den Mobiltelefonen der Benutzer als Push Benachrichtigung dargestellt.

Generell wird unter benutzerspezifischer Werbung verstanden, dass nur Konsumenten einer oder ähnlicher Speisen bzw. Tagesmenüs dieselbe Werbung erhalten. Außerdem kann diese Werbung auf Vergünstigungen für diverse Produkte hinweisen.

4.5.2 Werbungsempfang in der mobilen Applikation

Bereits im Pflichtenheft wurde festgelegt, dass der Werbungsempfang in den beiden mobilen Applikationen mittels Push Benachrichtigungen zu realisieren ist. Daher konnte sofort mit den Vorbereitungen für die Entwicklungsphase begonnen werden. Dabei war es sehr wichtig sich zuvor in die jeweiligen Dokumentationen einzulesen. Erst nachdem das Konzept hinter der Entwicklung von Push Benachrichtigungen verstanden war, wurde mit der Entwicklung der Werbungsanzeige für die mobilen Applikationen begonnen.

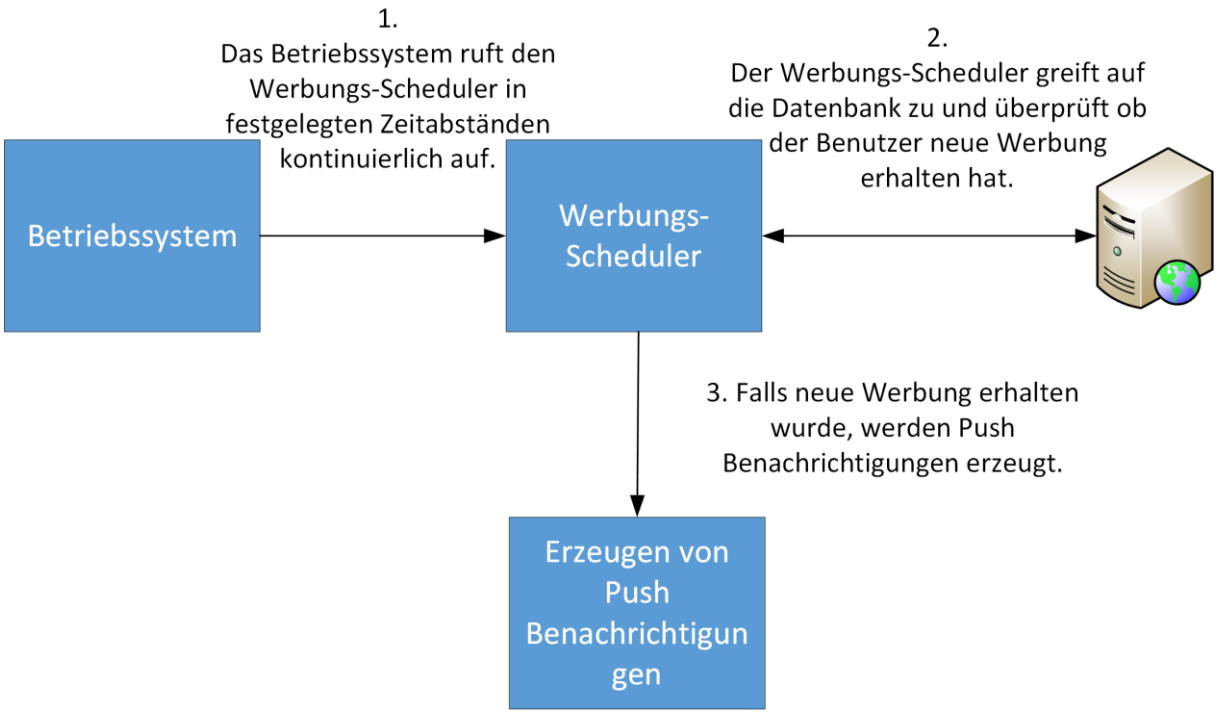


Abbildung 46: Erzeugung von Werbung

In der oberen Abbildung ist der grober Ablauf der Erzeugung von Push Benachrichtigungen für Werbezwecken in den mobilen Applikationen abgebildet. Der genaue Ablauf wird in den nächsten Abschnitten erklärt.

4.5.2.1 Aufruf des Werbungs-Scheduler

Um auch dann Werbung anzeigen zu können, wenn der Benutzer eine Anwendung gerade nicht gestartet hat, muss ein sogenannter Scheduler registriert werden. Dieser wird nach seiner Registrierung in bestimmten Zeitabständen aufgerufen. Dabei kann der Entwickler selbst entscheiden ob fixe Zeitabstände oder ungefähre Zeitabstände verwendet werden sollen. Generell wird empfohlen ungefähre Zeitabstände zu verwenden, wenn nicht unbedingt fixe Zeitabstände wie beispielsweise bei einem Wecker benötigt werden. Der Grund dafür ist, dass mehrere Scheduler, welche innerhalb einer ähnlichen Zeitspanne gestartet werden sollen, vom Betriebssystem gesammelt und somit nach einer bestimmten Zeit hintereinander aufgerufen werden können. Somit wird das Mobiltelefon seltener aus dessen Standby-Modus geholt, wodurch weniger Akku verbraucht wird. Daher wurde auch bei den beiden mobilen Applikationen dieser Diplomarbeit eine ungefähre Zeitangabe mit einer Zeitspanne von 30 Minuten verwendet.

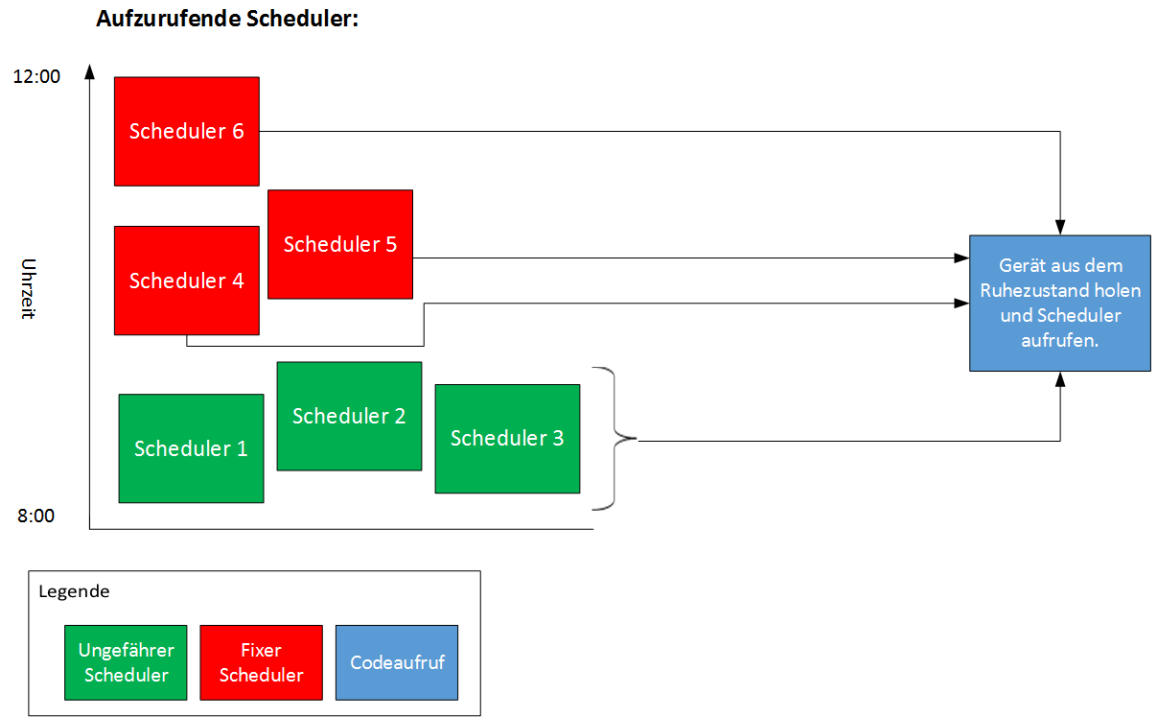


Abbildung 47: Scheduler mit ungefährer Zeitangabe vs. Scheduler mit fixer Zeitangabe

4.5.2.2 Zugriff auf die Datenbank

Beim Zugriff auf die Datenbank ist es besonders wichtig, vor dem Aufbau der Internetverbindung, diese zu überprüfen. Falls diese nicht geprüft wird, wird zwar versucht eine Internetverbindung herzustellen allerdings wird diese nach einem bestimmten Timeout fehlschlagen. Somit wird nur die Akkulaufzeit des Benutzers verschwendet. Falls allerdings eine Internetverbindung vorhanden ist, kann sofort mit dem Laden der Daten aus der Datenbank begonnen werden. Die Prüfung ob die jeweilige Werbung für den Kunden gedacht ist, erfolgt in dieser Diplomarbeit nicht auf dem Smartphone, sondern mittels Prozeduren auf dem Datenbankserver. Dies liegt daran, dass ein Smartphone Daten viel langsamer vergleichen kann als eine Datenbank. Außerdem werden somit weniger Daten aus dem Internet heruntergeladen.

Quellcode der Datenbankprozedur, welche die verfügbare Werbung für die jeweiligen Kunden ermittelt:

```
DROP PROCEDURE IF EXISTS AVAILABLEADVERTISEMENTSBYPRODUCT;
delimiter //
PROCEDURE `AVAILABLEADVERTISEMENTS`(IN advertisementkind char(11), IN
                                    iduser INT)
proc: begin
      DECLARE advertisement_id INT(11);
      DECLARE advertisement_query char(16383);
      DECLARE idrestaurant INT(11);
      DECLARE exit_loop BOOLEAN;
      DECLARE advertisement_ids text;
      DECLARE selectcount INT(11);
      DECLARE cur1 CURSOR FOR select advertisment_idadvertisment,
                  advertisment_query, restaurant_idrestaurant from
                  product_advertisment;
      DECLARE cur2 CURSOR FOR select advertisment_idadvertisment,
                  advertisment_query, restaurant_idrestaurant from
                  day_menu_advertisment;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
      OPEN cur1;
      OPEN cur2;
      SET advertisement_ids = "";
      advertisement_loop: LOOP
            IF advertisementkind="product" THEN
                  FETCH cur1 INTO advertisement_id, advertisement_query,
                        idrestaurant;
            ELSE
                  FETCH cur2 INTO advertisement_id, advertisement_query,
                        idrestaurant;
            END IF;
            IF exit_loop THEN
                  LEAVE advertisement_loop;
            END IF;
            SET advertisement_query = REPLACE(REPLACE(
               advertisement_query, "/idusers/", iduser),
               "/idrestaurant/", idrestaurant);
            CALL dynamic_sql(CONCAT(CONCAT("INSERT INTO dynamic_query
                 (result) (", advertisement_query), ")"));
            SELECT result INTO selectcount from dynamic_query order by
                   iddynamic_query desc limit 1;
            IF selectcount > 0 THEN
                  SET advertisement_ids = CONCAT(CONCAT(advertisement_ids,
                      advertisement_id), ",");
        END IF;
      END LOOP;
      SET advertisement_ids = SUBSTRING(advertisement_ids, 1, LENGTH(
                              advertisement_ids) - 1);

     SET @temp1 = CONCAT("select ", CONCAT(CONCAT(CONCAT("'select *
                  from ", advertisementkind), "_advertisment where
                  advertisment_idadvertisment in (", advertisement_ids),
                  ")' from dual;"));
      PREPARE stmt1 FROM @temp1;
    EXECUTE stmt1;
    DEALLOCATE PREPARE stmt1;
END //
```

4.5.2.3 Push Benachrichtigungen anzeigen

Nachdem die Daten erhalten wurden, werden diese sofort aus den entsprechenden DAOs geladen. Im Anschluss werden diese zu Push Benachrichtigungen umgewandelt. Bei dieser Umwandlung wird außerdem eine Verlinkung auf das jeweilige Restaurant hinzugefügt, sodass bei einem Klick auf eine Benachrichtigung die Aktivität des jeweiligen Restaurants angezeigt wird. Nachdem diese Umwandlung und Verlinkung abgeschlossen ist, werden die Benachrichtigungen in der Statusleise des Smartphones angezeigt. Falls nun ein Kunde auf eine Werbungsbenachrichtigung klickt, wird gespeichert, dass diese Werbung bereits angesehen wurde und somit nicht mehr angezeigt wird.

Quellcode der Funktion, die Werbungen zu Push Benachrichtigungen umwandelt und diese anzeigt:

```
private void sendNotification(Restaurant restaurant, Class
             advertismentClass, int advertismentId, String
             advertismentMessage) {
        Intent intent = new Intent(this, MapsActivity.class);  //If the
        passed class is wrong, just open the startpage (MapActivity)
        if (advertismentClass.toString().equals(
            ProductAdvertisment.class.toString())){
            intent = new Intent(this, MenuActivity.class);
        } else if (advertismentClass.toString().equals(
                    DayMenuAdvertisment.class.toString())){
            intent = new Intent(this, DayMenuActivity.class);
        }
        intent.putExtra("restaurant", restaurant.getIdrestaurant());
        intent.putExtra("advertisment", advertismentId);
        PendingIntent contentIntent = PendingIntent.getActivity(this,
                GlobalVariables.NotificationIDOffset + advertismentId,
                intent, PendingIntent.FLAG_UPDATE_CURRENT);  //GlobalVar
                            iables.NotificationIDOffset + ProductAdver
                            tisment.getIdProductAdvertisment() and
                            FLAG_UPDATE_CURRENT has to be done, because
                            otherwise getExtra returns the last passed
                            value and not the passed value
        NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(this)
                        .setSmallIcon(R.drawable.restaurant)
                        .setContentTitle(restaurant.getRestaurant_name()
                                        + " - Aktion")
                        .setStyle(new NotificationCompat.BigTextStyle()
                                .bigText(advertismentMessage))
                        .setContentText(advertismentMessage)
                        .setColor(restaurant.getRestaurant_logo_color());

        mBuilder.setContentIntent(contentIntent);

        mNotificationManager.notify(GlobalVariables.NotificationIDOffset
                                    + advertismentId, mBuilder.build());
    }
```

Abbildung 48: Beispiel für eine Werbungsbenachrichtigung

4.5.3 Werbungserstellung auf der Webapplikation

Um Werbung auf den Smartphones anzeigen zu können, muss der Restaurant-Administrator diese erst in der Webapplikation erstellen.

Dazu wird zuerst ausgewählt ob Werbung für ein Produkt oder für ein Tagesmenü erstellt werden soll. Nachdem diese Auswahl getätigt wurde, kann der Restaurant-Administrator die Nachricht eingeben, welche die Benutzer der Smartphone Applikationen im Anschluss erhalten.

Dabei müssen folgende Parameter angegeben werden:

- der Zeitraum in der das Angebot gültig ist
- das Produkt oder Tagesmenü das vergünstigt wird
- der Preis um wie viel das Produkt oder Tagesmenü vergünstigt wird
- das Restaurant wo das Angebot gültig ist
- der Text der Benachrichtigung an sich

Der Restaurant-Administrator hat nun die Möglichkeit diese Werbung für alle Kunden zu erstellen, die beispielsweise eine gewisse Anzahl eines bestimmten Produktes in einem angegeben Zeitraum gekauft haben.

Aus diesen Parametern wird eine Query erzeugt in welcher die entsprechenden Benutzer auswählt werden, auf die diese Kriterien zutreffen.

Beispiel einer Query:

```
select count(*) from users, users_to_restaurant where
                    iduser = user_iduser and
                    user_iduser=/idusers/ and
                    restaurant_idrestaurant=/idrestaurant/;
```

Mit dieser Query werden alle Benutzer selektiert, welche bereits ein entsprechendes Restaurant besucht haben.

Ist alles bestätigt worden, werden die Daten in den entsprechenden Datenbanktabellen gespeichert.

5 Ressourcenplanung

5.1 Hardware

5.1.1 Smartphone bzw. Tablet (Android)

Um die mobile Android Applikation entwickeln und testen zu können, wurde ein Android Gerät mit einer Mindestversion von Android 4.2 benötigt. Für die Anwendungstests wurde das Smartphone OnePlus One verwendet, welches während des Entwicklungsprozesses von der Android Version 4.4.4 laufend zu der Version 6.0 aktualisiert worden ist. Bei größeren Meilensteinen ist die App auch auf anderen Smartphones getestet worden.

Abbildung 49: OnePlus One

5.1.2 Tablet (iPad)

Für die Entwicklung der iOS Applikation, wurde ein Gerät ab der Version iOS 7 zum Testen benötigt. Während des Entwicklungsprozesses ist ein iPad 2 mit iOS 9.0 zur Verfügung gestanden. Des Weiteren wurde die Applikation auf diversen Emulatoren getestet.

Abbildung 50: iPad2

5.1.3 iPod Touch

Der iPod Touch läuft unter dem Betriebssystem iOS und kann daher ab iOS Version 7 diese Applikation verwenden. Daher ist dieser kurzfristig für die Testung der mobilen iOS Applikation verwendet worden. Die Hauptaufgabe eines iPods liegt darin Musik zu speichern und wiederzugeben. Deswegen ist dessen Performance etwas schlechter als die eines iPhones bzw. iPads.

Abbildung 51:IPod Touch

5.1.4 Windows Laptop

Für die Entwicklung der mobilen Android Applikation und der Webapplikation haben wir unsere Schullaptops verwendet. Diese hatten beide das Betriebssystem Windows 8 installiert.

Abbildung 52: Laptop

5.1.5 MacBook Air

Um Applikationen für das Betriebssystem iOS zu entwickeln, wird die Entwicklungsumgebung XCode benötigt. Da diese Entwicklungsumgebung derzeit nur für das Betriebssystem Mac OS verfügbar ist, wurde ein Mac Book für die Entwicklung der mobilen iOS Applikation verwendet. Dieses Gerät wurde von der HTL Perg bereitgestellt, wodurch hohe Kosten für dessen Anschaffung vermieden werden konnten.

Abbildung 53: MacBook Air

5.1.6 Server im Internet

Sämtliche Daten der Applikationen werden in einer Datenbank gespeichert und müssen vom Internet aus erreichbar sein. Daher wurde ein Server im Internet benötigt, welcher ebenfalls von der HTL Perg zur Verfügung gestellt wurde. Mit diesem Server konnten die mobilen Applikationen, sowie die Webapplikation unter realen Bedingungen getestet werden.

Auf dem Server wurde das Betriebssystem Windows Server 2012 R2 Standard verwendet.

Abbildung 54: IBM Server

5.2 Software

5.2.1 Windows Server 2012 R2

Windows Server 2012 R2 ist ein Betriebssystem, welches von Microsoft speziell für den Einsatz in Servern entwickelt wurde. Es ist in den Varianten Foundation, Essentials, Standard sowie Datacenter unterteilt.

Für dieser Diplomarbeit wurde die Version Standard verwendet, da diese alle benötigten Funktionen enthält. Generell fiel die Wahl auf dieses Betriebssystem, da es Windows 8 sehr ähnlich ist.

Abbildung 55. Windows Server 2012 Logo

5.2.2 Android Studio

Android Studio ist eine kostenlose Entwicklungsumgebung welche von Google entwickelt wurde und seit ihrer Veröffentlichung im Jahr 2013, die offizielle Entwicklungsumgebung für die Entwicklung von Android Applikationen ist. Android Studio basiert auf IntelliJ IDEA.

Android Studio wurde verwendet, da diese, im Gegensatz zu andern Entwicklungsumgebungen, einige Vorteile bietet.

Abbildung 56: Android Studio Logo

5.2.3 XCode

XCode ist eine kostenlose Entwicklungsumgebung welche von Apple veröffentlicht wurde. Diese ist die offizielle Entwicklungsumgebung für die Entwicklung von iOS Applikationen und Mac OS Anwendungen. XCode unterstützt die Programmiersprachen C, Objective-C, C++ sowie Swift.

Für die Entwicklung der iOS Applikation unter der Programmiersprache Swift, wurde XCode verwendet.

Abbildung 57: XCode Logo

5.2.4 Eclipse

Eclipse ist eine kostenlose Entwicklungsumgebung, entwickelt von der Eclipse Foundation. Ihr ursprünglicher Zweck war es eine integrierte Entwicklungsumgebung für die Programmiersprache Java bereitzustellen. Mittlerweile ist Eclipse, wegen dessen Möglichkeit der Erweiterbarkeit, auch für viele andere Entwicklungsaufgaben im Einsatz.

Eclipse wurde für die Entwicklung der Webapplikation verwendet, dabei wurde auf die aktuellste Version „Mars“ zurückgegriffen.

Abbildung 58: Eclipse Logo

5.2.5 MySQL Workbench

MySQL Workbench ist ein Datenbank-Modellierungswerkzeug, welches von der Oracle Corporation entwickelt wurde. Es bietet eine benutzerfreundliche Oberfläche zur Erstellung von Datenbankmodellen und zur Verwaltung von MySQL Datenbanken.

Zur Modellierung des Datenbankmodells wurde die MySQL Workbench verwendet. Zudem wurde der Großteil der Wartung des Datenbankmodells mit Hilfe dieser Software durchgeführt.

Abbildung 59: MySQL Workbench Logo

5.2.6 Google Drive

Der von Google entwickelte Filehosting-Dienst Google Drive ermöglicht das Speichern und Herunterladen von Dateien in einer Cloud. Die hochgeladenen Dateien können, je nach Freigabe, von unterschiedlichen Benutzern angesehen oder bearbeitet werden.

Google Drive diente zum Verteilen und Austauschen sämtlicher schriftlicher Dokumente.

Abbildung 60: Google Drive Logo

5.2.7 VMware vSphere Client

VMware vSphere Client ist eine Software, welche Zugang zu virtuellen Maschinen auf einem vSphere Server ermöglicht.

Mit Hilfe des VMware vSphere Client wurde der Fernzugriff auf den Windows Server ermöglicht und somit die Entwicklung der Webapplikation gestattet. Außerdem wurde so ein direkter Zugriff auf die MySQL Datenbank ermöglicht.

Abbildung 61: VMWare vSphere Client

6 Implementierung

6.1 Schnittstellen

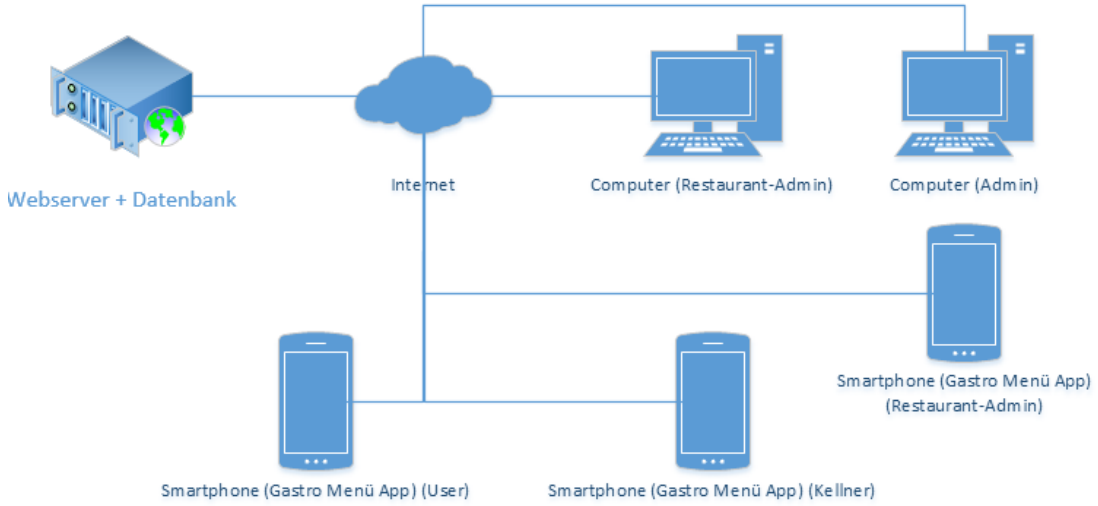


Abbildung 62: Darstellung der Schnittstellen

In der vorhergehenden Abbildung sind die Schnittstellen dieser Diplomarbeit dargestellt. Grundlegend greifen alle Schnittstellen über das Internet auf eine gemeinsame Datenbank zu. Kunden sowie Kellner können nur mit einem Smartphone, auf welchem unsere App installiert ist, auf die Datenbank zugreifen. Restaurant-Administratoren können, sowohl mit einem Smartphone, als auch mittels eines Computers über die Webapplikation, auf die Datenbank zugreifen. Administratoren haben nur die Möglichkeit mit der Webapplikation auf die Datenbank zuzugreifen.

6.2 Technische Begriffe

6.2.1 Mobile Applikationen und Webapplikation

6.2.1.1 MySQL

MySQL ist ein seit 1994 bestehendes relationales Open-Source Datenbankverwaltungssystem, welches 2008 von Sun Microsystems übernommen wurde und seinerseits von Oracle gekauft wurde. MySQL ist, neben Oracle und dem Microsoft SQL Server, eines der weitverbreitetsten Datenbankverwaltungssysteme seiner Art, welches besonders bei Datenspeicherungen für Webservices eingesetzt wird.

Sämtliche Daten sind in einer gemeinsam, von den mobilen Applikationen und der Weboberfläche, genutzten MySQL Datenbank gespeichert.

Abbildung 63: MySQL Logo

6.2.1.2 XAMPP

XAMPP ist eine Sammlung freier Software welche es ermöglicht einen Webserver mit einer Datenbank aufzusetzen. Generell umfasst XAMPP folgende Technologien: Apache HTTP Server, MySQL, PHP und Perl. Von diesen Technologien leitet sich auch der letzte Teil der Abkürzung XAMPP ab. Der erste Buchstabe X steht für cross-platform, da XAMPP auf allen gängigen Betriebssystemen installiert werden kann.

Abbildung 64: XAMPP Logo

Außerdem wurden folgende Programme zur Entwicklung der mobilen Applikationen und der Weboberfläche verwendet:

- MySQL Workbench
- Windows Server
- Google Drive
- VMware vSphere Client

Die nähere Beschreibung dieser Programme befindet sich im Abschnitt 5.2.

6.2.2 Mobile Applikationen

6.2.2.1.1 Android SDK

Das Android Software Development Kit bzw. Android SDK wird für die Entwicklung nativer Android Applikation verwendet. Es erweitert die Bibliotheken des Java SDKs um einige weitere Funktionalitäten, welche die Entwicklung mobiler Applikationen vereinfacht.

Da das Android SDK auf dem Java SDK basiert, können Android Applikationen auch mit allen gängigen Java-Entwicklungsumgebungen erstellt werden.

Abbildung 65: Android Logo

In der nachfolgenden Abbildung ist der LifeCycle von Android-Aktivitäten dargestellt. Jede Android-Aktivität erbt von der Überklasse Activity und funktioniert daher nach diesem LifeCycle.

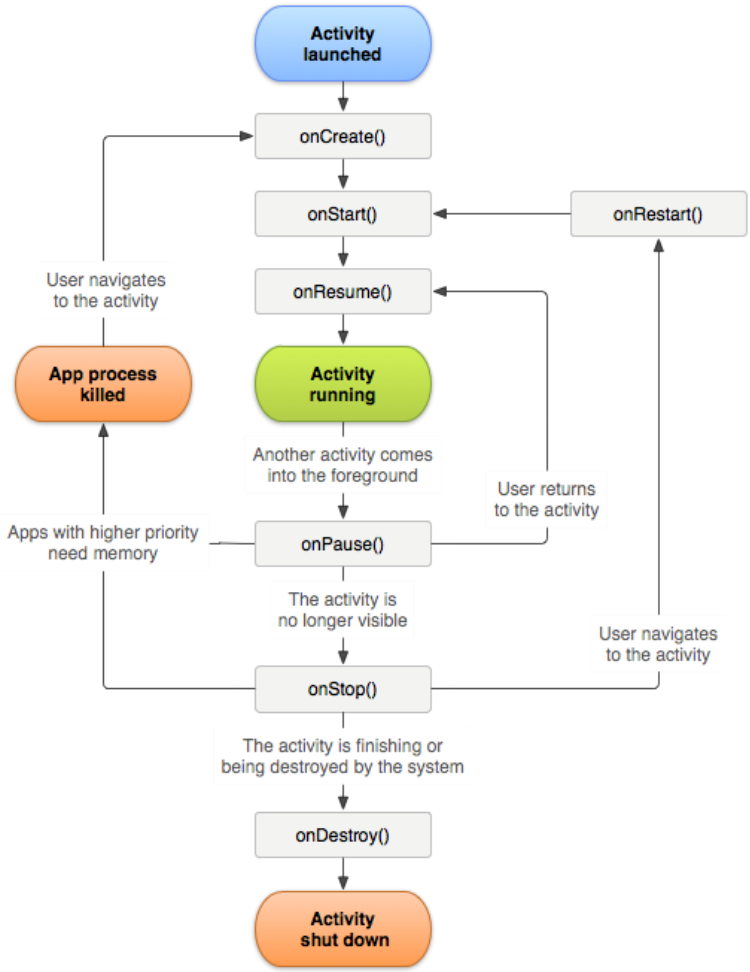


Abbildung 66: Android Activity-LifeCycle

6.2.2.2 Swift

Die Programmiersprache Swift wurde von dem Unternehmen Apple entwickelt, um die Programmiersprache Objective-C abzulösen. Seit ihrer Veröffentlichung im Jahr 2014 wurden immer mehr Anwendungen mit Swift entwickelt. Mit dieser Programmiersprache können sowohl mobile iOS Applikationen als auch Mac OS Desktopanwendungen programmiert werden. Die offizielle Entwicklungsumgebung für diese Programmiersprache ist XCode, welche ebenfalls von Apple entwickelt wurde.

Abbildung 67: Swift Logo

In der folgenden Abbildung ist der LifeCycle von iOS Aktivitäten abgebildet nach dem alle iOS-Aktivität funktionieren.

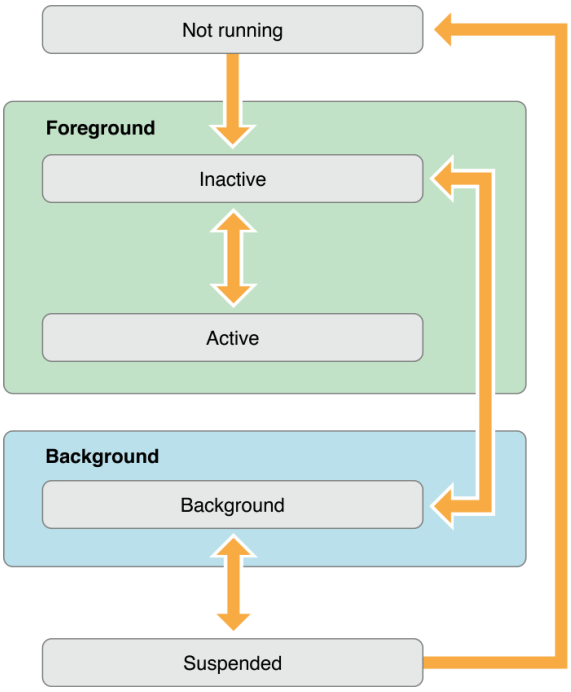


Abbildung 68: iOS Activity-LifeCycle

6.2.2.3 XML

XML ist eine Datenbeschreibungssprache und steht für Extensible Markup Language. Der Unterschied von XML zu anderen Datenbeschreibungssprachen ist, dass XML-Daten sowohl von Menschen, als auch von Rechnern einfach gelesen werden können. Dies wird ermöglicht indem XML Daten hierarchisch strukturiert abspeichert werden. Hauptsächlich wird XML zur Datenübertragung im Internet verwendet.

Abbildung 69: Bild, welches XML-Dateien repräsentiert.

6.2.2.4 VMware Player

Das Programm VMware Player ermöglicht innerhalb eines Rechners weitere virtuelle Rechner zu erstellen. Somit können Anschaffungskosten für zusätzliche Computer oder Server eingespart werden. Außerdem ist VMware Player im Vergleich zu ähnlichen Programmen für den privaten Gebrauch kostenlos. Das Programm selbst wurde von dem gleichnamigen Unternehmen VMware entwickelt.

Abbildung 70: VMware Player Logo

6.2.2.5 PHP

Die serverseitige Skriptsprache PHP wurde erstmals 1995 veröffentlicht und ist heutzutage eine der gängigsten Programmiersprachen für die Entwicklung einfacher Webanwendungen. Ursprünglich bedeutete PHP Personal Homepage Tools. Die aktuellste Version von PHP (zurzeit 7.0.4) enthält bereits viele Funktionen. Somit kann diese Technologie, auch ohne weitere Bibliotheken, auf Datenbanken zugreifen.

Abbildung 71: PHP logo

Außerdem wurden folgende Programme zur Entwicklung der mobilen Applikationen verwendet:

- Android Studio
- XCode

Die nähere Beschreibung dieser Programme befindet sich im Abschnitt 5.2.

6.2.3 Webapplikation

6.2.3.1 JavaServer Faces

JavaServer Faces ist eine Technologie zum Entwickeln von dynamischen Webanwendungen und gehört zur Java Plattform, Enterprise Edition.

Das Framework dient zur Entwicklung von Webapplikationen. Dabei werden diverse Komponenten in Webseiten eingebunden.

Abbildung 72: JSF Logo

6.2.3.1.1 Allgemeines

1997 wurde die Servlet-Technologie[18] in Java eingeführt. Damit war es möglich HTML Seiten dynamisch am Server zu generieren. Doch diese Technologie wurde wieder verworfen, da in Java die gesamte Seite mit *println*-Anweisungen erstellt wurde und dadurch schnell unübersichtlich wurde.

Daher wurde später die Web-Programmiersprache JavaServer Pages[19] entwickelt. Mit JSP war es nun möglich Java-Methoden in HTML einzubinden, was zur Erleichterung der Erstellung von komplexeren Seiten führte.

6.2.3.1.2 Model-View-Controller

Webframeworks wurden eingeführt um die Layout- und Funktionsbeschreibung möglichst zu trennen. Das heißt in der Seitendeklarationssprache (bspw. JSP oder XHTML) sollte so wenig Anwendungslogik wie möglich sein. Um diese Trennung konsequent durchzuführen, wird in der Praxis ein Entwicklungsmuster verwendet. Das Model-View-Controller-Muster[20] strukturiert in einer JSP/JSF-Anwendung Modell (Bean), Ansicht (JSP/XHTML) und Steuerungslogin (Servlet).

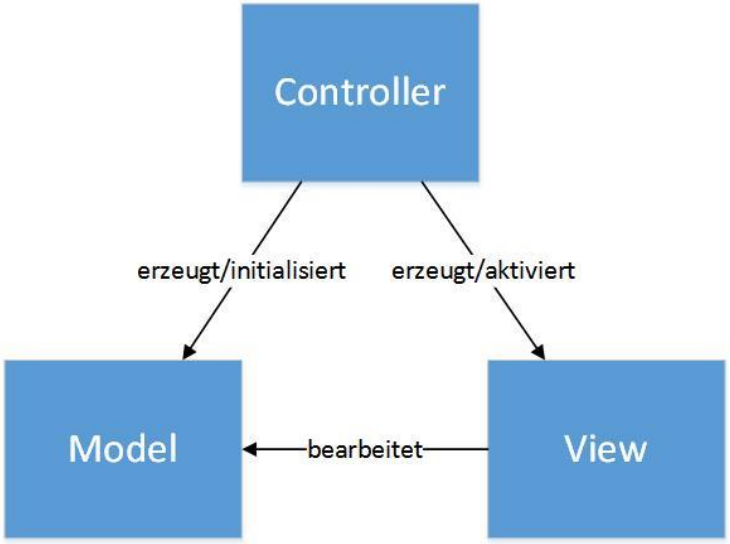


Abbildung 73: MVC

[18] Servlets: nehmen Server Anfragen von Clients und beantworten diese.
[19] JSP: eine Web-Programmiersprache zur dynamischen Erzeugung von Webseiten.
[20] MVC: ein Entwicklungsmuster welches in drei Einheiten aufgeteilt ist (Model, View und Controller) und spätere Veränderungen oder Erweiterungen erleichtern soll.

6.2.3.1.3 Aufbau

Das Framework JavaServer Faces löste 2004 JSP ab, mit dem Zweck die Entwicklung von Webanwendungen zu standardisieren. Seitdem hat sich JSF stetig weiterentwickelt und es wurden zahlreiche Erweiterungen dazu entwickelt.

6.2.3.1.4 Implementierung

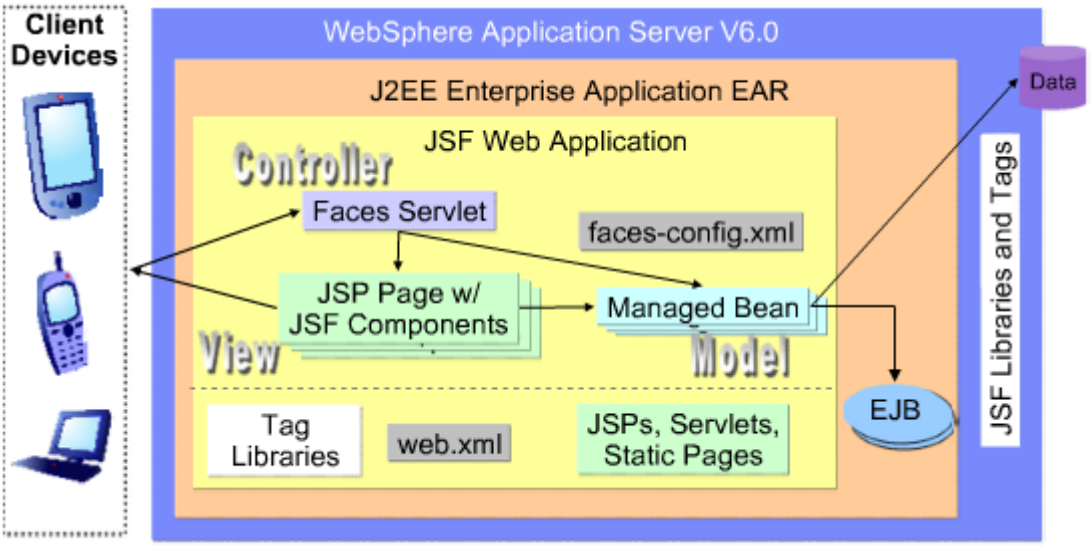


Abbildung 74: MVC Prinzip in JSF

Für die Entwicklung einer JSF-Applikation muss zuallererst ein aktuelles JDK[21] installiert werden. Weiters wird ein Anwendungsserver,[22] wie WildFly[23] oder ein Servlet-Container[24] als Laufzeitumgebung benötigt.

6.2.3.1.5 web.xml

Die web.xml Datei ist eine Webkonfigurationsdatei die auch Deployment Deskriptor genannt wird. In dieser Konfigurationsdatei kann beispielsweise das *welcome-file*, zur Bestimmung der Startseite oder das *servlet-mapping* eingestellt werden. Letztere Einstellung dient dazu, dass Dateien mit einer bestimmten Endung (z.B. „.jsf“) von dem JSF-Servlet bearbeitet werden.

```
<welcome-file-list>
  <welcome-file>welcome.jsf</welcome-file>
</welcome-file-list>

<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

[21] JDK: das Java Development Kit wird für das Entwickeln von Java Anwendungen benötigt.
[22] Anwendungsserver: ein Server der Anwendungsprogramme ausführt.
[23] WildFly Server: ein Anwendungsserver nach JavaEE Standard.
[24] Servlet-Container: unterstützten die Servlet API.

6.2.3.1.6 FacesConfig.xml

In der FacesConfig.xml Datei werden die Navigationsregeln der JSF-Applikation definiert.

```
<navigation-rule>
  <from-view-id>/login.xhtml</from-view-id>
  <navigation-case>
   <from-outcome>welcome</from-outcome>
   <to-view-id>/welcome.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

6.2.3.1.7 XHTML

XHTML wird in JSF als Seitendeklarationssprache verwendet. Dort ist definiert was auf der Website angezeigt wird. Daten werden in XHTML von den Bean Klassen in der Form *#{managedBean.eigenschaft}* aufgerufen und angezeigt.

```
<h:commandButton   value="Löschen"   action="#{userBean.delete(userBean.u-
ser.iduser)}"/>
```

In diesem Beispiel wird ein Button zum Löschen eines Benutzers in XHTML definiert. Der Button greift, wenn er geklickt wird, auf eine Managed Bean Klassen Methode zu.

6.2.3.1.8 Managed Bean

Managed Bean Java Klassen liefern die Werte zum Befüllen der JSF Komponenten in den XHTML Dateien. Sie bilden die Verbindung zum Modell und der Geschäftslogik.

```
public String delete(int id) {
      userDao.delete(id);
      return "manageUser?faces-redirect=true";
}
```

Hier ist zu sehen wie die Managed Bean Methode zum Löschen eines Benutzers wiederum auf eine Dao Methode zugreift.

6.2.3.1.9 Dao

Um Daten von einer Datenbank abzurufen oder in die Datenbank zu schreiben werden sogenannten Dao[25] Klassen benötigt. Diese Klassen greifen beispielsweise mittels der Hibernate-Language[26] auf Model Klassen zu, wo die Tabellen der Datenbank abgebildet sind.

```
public void delete(int idUser){
      User userManaged = load(idUser);
      entityManager.remove(userManaged);
}
```

Im obigen Beispiel wird in der Dao Klasse ein Benutzer aus der Datenbank entfernt. Dazu wird das zu löschende Benutzer Objekt geladen und mit der EntityManager[27] Methode *remove* aus der Datenbank entfernt.

[25] Dao: ein Data Access Object ist eine Schnittstelle für den Zugriff von Daten einer Datenbank.
[26] HQL: die Hibernate Query Language ermöglicht Objekte in eine relationale Datenbank zu speichern und umgekehrt aus Datensätzen Objekte zu erzeugen.
[27] EntityManager: ein Interface zur Interaktion mit dem persistence context, welches Entity Instanzen

6.2.3.1.10 JSF LifeCycle

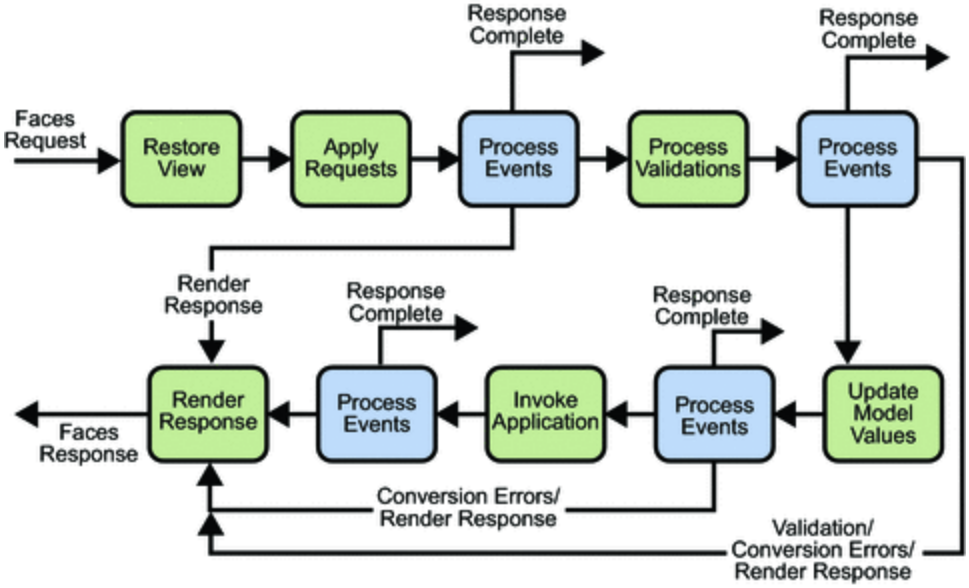


Abbildung 75:JSF-LifeCycle

Der JSF-Applikation-LifeCycle kann in sechs Phasen unterteilt werden:

1. Ansicht wiederherstellen
    - Aufbau des Komponentenbaumes nach Anfrage eines Clients
2. Request-Parameter anwenden
    - Komponentenbaum wird bearbeitet und Werte, die vom Benutzer eingetragen wurden, werden den einzelnen Komponenten zugewiesen
3. Validierung durchführen
    - der ausgelesene Wert einer Komponente wird konvertiert und validiert
4. Modell aktualisieren
    - Werte der Komponenten werden auf den Server gespeichert
5. Applikation ausführen
    - ausführen von Aktionen
6. Antwort rendern
    - Komponentenbaum wird gerendert und eine Antwort zum Client geschickt

verwaltet.

6.2.3.2 HTML und XHTML

HTML (Hypertext Markup Language) ist eine Auszeichnungssprache und Grundlage für das World Wide Web. Sie wird von Web-Browsern interpretiert und angezeigt.

Abbildung 76: HTML Logo

HTML beschreibt die logischen Bestandteile eines digitalen Dokuments wie z.B. Überschriften, Aufzählungen, Hyperlinks und Bilder, formatiert den Text aber nicht. Die Sprache dient um einen Text zu strukturieren, die visuelle Darstellung ist jedoch nicht Teil der HTML-Spezifikation sondern wird durch den Webbrowser sowie der Stylesheet Sprache CSS bestimmt. Mit sogenannten Tags wird die Struktur einer Website definiert welche der Browser liest und anzeigt.

Zusätzlich können in einem HTML-Dokument Metainformationen, beispielsweise über die verwendete Sprache oder über den Autor, enthalten sein welche wichtig für die Indexierung von Suchmaschinen wichtig ist.

Abbildung 77: XHTML Logo

XHTML (Extensible Hypertext Markup Language) ist eine striktere Neuformulierung von HTML in XML als Sprachgrundlage.

Da XML eine immer größer werdende Bedeutung für viele Dateiformate fand, wurde auch HTML mit XML definiert. Damit keine Inkompatibilität entstand, entschied man sich für die neue Sprache XHTML.

Verwendung

XHTML wurde bei der Webapplikation als Seitendeklarationssprache verwendet. XHTML Dateien, sind jene Dateien die sich am Server befinden und mit einem Webbrowser, beispielsweise mit der Dateiendung „.jsf“, aufgerufen werden. Bei dem Aufruf wird das FacesServlet aufgerufen, die „.xhtml“ Datei gesucht und die enthaltenen JSF Komponenten geparst.

Für nähere Information siehe Abschnitt 6.2.3.1.

6.2.3.3 CSS

Die Stylesheet-Sprache CSS (Cascading Style Sheets) wird oft zusammen mit HTML im World-Wide-Web eingesetzt.

Abbildung 78: CSS Logo

Mit Hilfe von CSS werden Gestaltungsanweisungen erzeugt die von Auszeichnungssprachen wie HTML und XML eingesetzt werden. CSS definiert somit das Layout von Dokumenten im World Wide Web. Der Sinn dahinter ist das Layout möglichst von Inhalten zu trennen. Daher wird in HTML bei Inhaltsabschnitten oft auf eine separate CSS Datei verwiesen um eine konsequente Trennung durchzuführen. Dies bringt auch den Vorteil mit sich, dass eine CSS Datei nur einmal, beispielsweise von einem Webbrowser, geladen werden muss und danach vielen Dokumenten zur Verfügung steht.

Verwendung

CSS wurde für die Gestaltung der Weboberfläche verwendet. Dazu wurden CSS Dateien als Template geschrieben, die von einem JSF-Template aufgerufen wurden.

Für nähere Information siehe Abschnitt 4.2.2.

6.2.3.4 JavaScript

Abbildung 79: JavaScript

JavaScript ist eine klassenlose Skriptsprache die vor allem von Webbrowsern unterstützt wird und die neben dem funktionalen Stil auch die objektorientiere Programmierung beherrscht. Sie wurde entwickelt um auf der Client Seite auf Benutzerinteraktionen, vor allem in HTML, dynamisch zu reagieren indem zum Beispiel Inhalte verändert, nachgeladen oder generiert werden.

Heute wird JavaScript vielseitiger eingesetzt und ist nicht nur in Webbrowsern in Anwendung.

Anwendung

Die Technologie JavaScript wurde für Einbindung von Google Maps verwendet.

Für nähere Information siehe Abschnitt 3.3.2.2.3.2.

6.2.3.5 Java

Java ist eine seit 1995 bestehende, weitverbreitete objektorientierte Programmiersprache, entwickelt von Sun Microsystems, welche aus dem Entwicklungswerkzeug JDK (Java Development Kit) und der Laufzeitumgebung JRE (Java Runtime Environment) besteht.

Durch das JDK wird der Java-Quellcode kompiliert und somit in einen maschinenverständlichen Bytecode zur Ausführung übersetzt.

Abbildung 80: Java Logo

Die Java Virtual Machine (JVM) ist in der JRE enthalten und dient zur Ausführung des durch das JDK erzeugten Maschinencodes. Somit kann ein Java-Programm plattformenunabhängig auf jeder Rechnerarchitektur laufen, wenn die Java Laufzeitumgebung installiert ist, da der Code nicht direkt durch die Hardware ausgeführt wird, sondern in einer virtuellen Maschine läuft.

Anwendung

Die Funktionalität der Weboberfläche wurde in Java realisiert.

Für nähere Information siehe Abschnitt 6.2.3.1.

6.2.3.6 JEE

Die Java Platform, Enterprise Edition (JEE) ist eine anwendungsorientierte Middleware vergleichbar mit der .NET-Plattform von Microsoft. Java EE spezifiziert eine Softwarearchitektur für Java und Web-Anwendungen und definiert Softwarekomponenten und Dienste. Dabei stellt die Plattform APIs und eine Laufzeitumgebung zur Verfügung um einen einheitlichen Rahmen für die Entwicklung von mehrschichten Anwendung zu bieten. Dadurch wird eine Kompatibilität von unterschiedlichen Softwarekomponenten gewährleistet.

Abbildung 81: JavaEE Logo

Anwendung

Die Weboberfläche wurde mit JSF, einer Webtechnologie der JEE, entwickelt.

Für nähere Information siehe Abschnitt 6.2.3.1.

6.2.3.7 Wildfly

WildFly, früher bekannt unter dem Namen JBoss, ist neben Apache TomEE oder Oracle GlassFish ein plattformenunabhängiger Open-Source Anwendungsserver nach dem Java EE Standard.

WildFly enthält einen vollwertigen Java EE Server. Außerdem verfügt es über CDI für die Bean Verwaltung und Dependency Injection.

Abbildung 82: WildFly Logo

Anwendung

Die Webanwendung läuft auf dem WildFly-Server.

Für nähere Information siehe Abschnitt 6.2.3.1.

6.2.3.8 jQuery

jQuery ist eine Bibliothek zur Erweiterung von JavaScript um Funktionalitäten wie z.B. der DOM-Navigation und Manipulation oder dem Erstellen von Animationen und der Entwicklung von Ajax Applikationen.

jQuery ist die am häufigsten genutzte JavaScript Library weltweit. Mehr als die Hälfte der populärsten Webseiten nutzen diese Bibliothek.

Abbildung 83: jQuery Logo

Anwendung

jQuery wurde für Einbindung von Google Maps verwendet.

Für nähere Information siehe Abschnitt 3.3.2.2.3.2.

6.2.3.9 Ajax

Ajax (Asynchronous JavaScript and XML) ist eine Technologie zum asynchronen Datenaustausch zwischen dem Browser und einem Server. Dadurch wird es ermöglicht Teile einer Website zu aktualisieren, ohne die Seite gänzlich neu zu laden.

Anwendung

Ajax wurde in der Webapplikation in Zusammenhang mit JSF verwendet, um bei einer gewissen Vorauswahl andere Auswahlen nicht mehr zur Verfügung zu stellen.

Abbildung 84: Ajax Logo

6.2.3.10 PrimeFaces

PrimeFaces ist eine Open Source Erweiterung für JSF von der Firma PrimeTek. Die Bibliothek erweitert JSF um über 100 Userinterface Komponenten und beinhaltet mehrere Frameworks.

Abbildung 85: PrimeFaces Logo

Anwendung

PrimeFaces wurde für sämtliche nicht standartgemäße JSF Komponenten verwendet wie z.B.: für das Dropdown-Menü zum Selektieren mehrere Einträge oder das Balken Diagramm.

Für nähere Information siehe Abschnitt 3.3.2.2.4 und 3.3.2.2.7.

6.2.3.11 PBKDF2WithHmacSHA1 Verschlüsselung

Die PBKDF2WithHmacSHA1 (Password Based Key Derivative Function With Keyed Hash Message Authentication Code - Secure Hash Algorithm 1) Verschlüsselung ist eine Hash-Verschlüsselung die das Key Stretching Verfahren nutzt, um eine Brute-Force Attacke durch eine hohe RAM Auslastung möglichst zu verhindern.

Anwendung

Das PBKDF2WithHmacSHA1-Verfahren wird bei der Verschlüsslung und Abgleichung des Passworts in der Webapplikation verwendet.

Außerdem wurden folgende Programme zur Entwicklung der Webapplikationen verwendet:

- Eclipse
- MySQL Workbench
- XAMPP

Die nähere Beschreibung dieser Programme befindet sich im Abschnitt 5.2 und 6.2.1.

7 Datenbankmodell

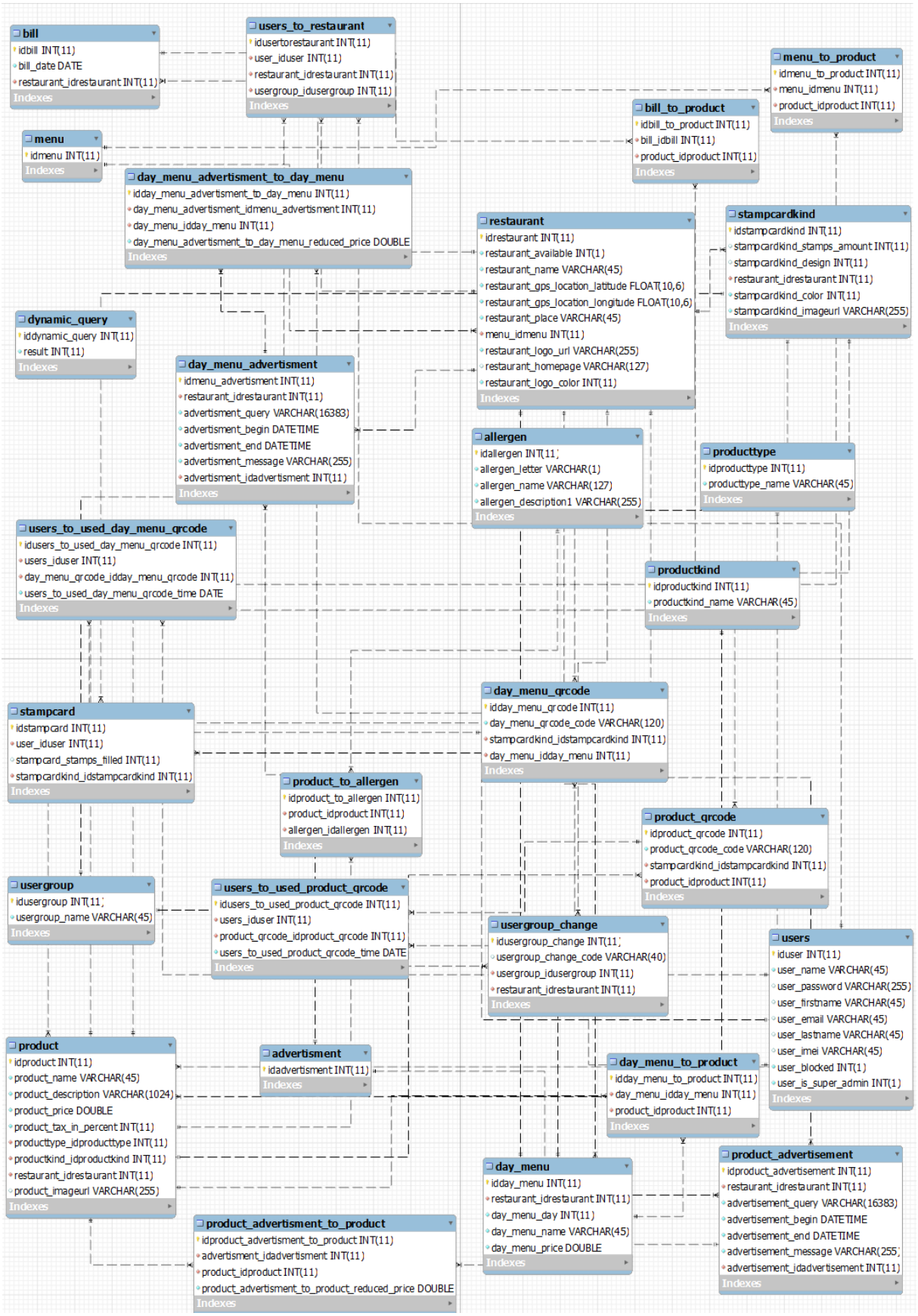
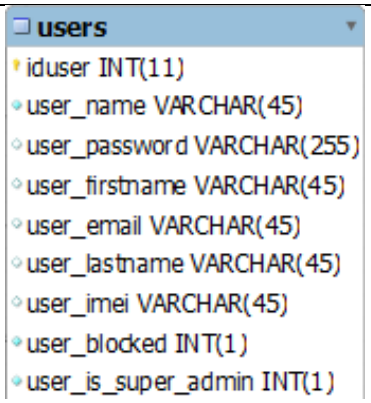
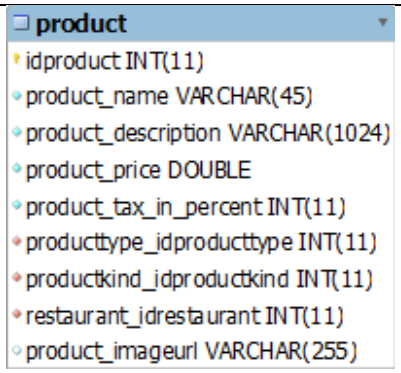
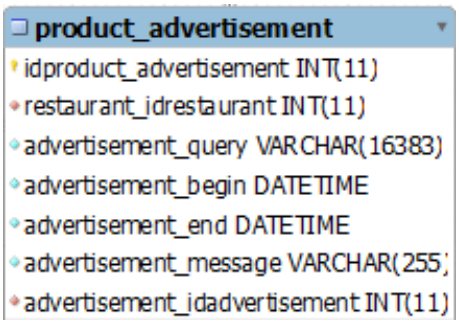


Abbildung 86: Datenbankmodell

Wie in der vorhergehenden Abbildung bereits zu erkennen ist, ist das für unsere Diplomarbeit entwickelte Datenbankmodell sehr umfangreich. Aufgrund dessen wurde das im nachfolgenden Abschnitt dargestellte Datenlexikon auf die wesentlichen Tabellen reduziert.

7.1 Datenlexikon & Beschreibung der wichtigsten Tabellen

Tabelle	Beschreibung
	Ein Restaurant hat eine Identifikationsnummer, eine Namen, eine Homepage, eine Ortsangabe als Text sowie zwei Positionskoordinaten. Außerdem ist zu jedem Restaurant auch ein Logo abgespeichert, welches als URL hinterlegt wird und einer bestimmten Farbe zugewiesen ist. Zudem muss noch eine Speisekarte angeben werden und es muss festgelegt werden, ob ein Restaurant verfügbar ist oder nicht.
	Ein Benutzer hat eine Identifikationsnummer, einen Benutzernamen, ein Passwort, einen Vornamen, einen Nachnamen sowie eine E-Mail Adresse. Zu einem Benutzer wird zudem noch die IMEI seines Mobiltelefons gespeichert. Es kann auch angegeben werden, ob ein Benutzer ein Administrator ist. Außerdem können Benutzer gesperrt werden.
	Ein Produkt hat eine Identifikationsnummer, einen Namen, eine Beschreibung, einen Preis sowie eine Steuersatzangabe. Außerdem beinhaltet es auch ein Bild, welches als URL gespeichert wird. Darüber hinaus gehört ein Produkt immer zu einem Restaurant und es hat einen Produkttyp sowie eine Produktart.
	Eine Produktwerbung hat eine Identifikationsnummer, eine Nachricht, eine Anfangszeit, eine Endzeit sowie eine Abfrage um festzustellen, welche Benutzer Werbung erhalten sollen. Außerdem wird sie immer einem Restaurant zugeordnet und gehört einer generellen Werbung an. Die Tagesmenüwerbung-Tabelle ist ähnlich wie die Produktwerbung-Tabelle aufgebaut.

8 Evaluierung

8.1 Ergebnis und Planung

Zu Beginn dieser Diplomarbeit wurde ein Pflichtenheft erstellt, in dem diverse Muss- als auch Abgrenzungskriterien definiert wurden. Im Laufe der Arbeit haben sich einige dieser Kriterien verändert. Zum Beispiel ist die ursprüngliche Funktion des Druckens einer Rechnung mit einem QR-Code, wo alle Speisen erfasst sind, aufgrund rechtlicher Gründen nicht realisierbar gewesen. Somit war auch die Erfassung der einzelnen Tische nicht mehr nötig.

Ein QR-Code wird nun nicht mehr mit der Rechnung abgedruckt, sondern von einem Kellner zum Scannen an den Kunden gegeben. Dieser QR-Code steht nun nicht mehr für alle Produkte auf der Rechnung, sondern für ein Produkt bzw. Tagesmenü.

8.2 Resümee

Durch die Diplomarbeit haben wir viele neue Erfahrungen gesammelt. Sowohl in den Bereichen des Projektmanagements und in der Programmierung.

Dabei lernten wir selbständig zu arbeiten und uns in neue Technologien einzuarbeiten. Außerdem konnten wir theoretische Konzepte, welche in unserer Ausbildung vermittelt wurden, in der Praxis umsetzen.

Aufgrund des großen Umfangs der Diplomarbeit war ein gutes Zeitmanagement wichtig. Daher haben wir unsere Fähigkeiten der Zeitplanung im Laufe der Arbeit wesentlich verbessert.

Durch die Arbeit lernten wir aufgetreten Problemen systematisch zu analysieren und zu lösen, da nicht immer sofort Hilfe zur Seite gestanden ist.

Alles in allem war diese Diplomarbeit eine wichtige Erfahrung für uns. Durch sie haben wir wesentliche Erkenntnisse für das später Berufsleben sammeln können.

9 Verfassungsnachweis

9.1 Andreas Egger

- Danksagung
- Einleitung
- Beteiligte
- Allgemeine Zielsetzung
- Vorgehensplan - Mobile Applikation
- Grundsystem - Funktionalität der mobilen Applikation
- Funktionalität der grafischen Benutzeroberfläche - Mobile Applikation
- Funktionalität des QR Code Scanners
- ORM Tool
- Benutzerspezifische Werbung- Werbungsempfang in der mobilen Applikation
- Ressourcenplanung - Hardware
- Implementierung - Schnittstellen
- Implementierung - Mobile Applikationen und Web
- Implementierung - Mobile Applikationen
- Datenbankmodell
- Evaluierung

9.2 Erik Puchinger

- Danksagung
- Einleitung
- Beteiligte
- Thema und Aufgabenstellung
- Vorgehensplan - Webapplikation
- Grundsystem - Funktionalität der Webapplikation
- Funktionalität der grafischen Benutzeroberfläche - Webapplikation
- Funktionalität des QR Code Scanners - Realisierung des QR-Code Generators in der Webapplikation
- Benutzerspezifische Werbung - Werbungserstellung auf der Webapplikation
- Ressourcenplanung - Software
- Implementierung - Mobile Applikationen und Web
- Implementierung - Webapplikation
- Evaluierung

10 Quellenverzeichnis

10.1 Textquellen

- http://jsfatwork.irian.at/book_de/introduction.html#!idx:/introduction.html:1
- http://stackoverflow.com/questions/3008395/jsf-facelets-sometimes-i-see-the-url-is-jsf-and-sometimes-xhtml-why
- http://balusc.omnifaces.org/2014/10/jsf-22-tutorial-with-eclipse-and-wildfly.html
- https://developers.google.com/chart/infographics/docs/qr_codes#syntax
- http://stackoverflow.com/questions/5161465/how-to-create-custom-easing-function-with-core-animation/5161588
- http://www.aberger.at
- http://wirtschaftslexikon.gabler.de/Definition/cascading-style-sheets.html?referenceKeywordName=CSS
- https://wiki.selfhtml.org/wiki/CSS/Einbindung
- http://wirtschaftslexikon.gabler.de/Definition/html.html
- http://www.gruenderszene.de/lexikon/begriffe/html
- https://wiki.selfhtml.org/wiki/HTML/Unterschiede
- http://www.itwissen.info/definition/lexikon/JavaScript-JavaScript.html
- http://developer.android.com/reference/android/app/Activity.html
- http://developer.android.com/training/scheduling/alarms.html
- http://www-03.ibm.com/systems/infrastructure/de/de/
- https://github.com/tadija/AEXML
- https://www.wko.at/Content.Node/branchen/oe/Gastronomie/Lobbying---Branchenthemen/Weiterfuehrende_Infos_Allergene.html
- http://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/

10.1.1 Wikipedia

- https://de.wikipedia.org/wiki/JavaServer_Faces
- https://de.wikipedia.org/wiki/Servlet
- https://de.wikipedia.org/wiki/JavaServer_Pages
- https://de.wikipedia.org/wiki/Model_View_Controller
- https://de.wikipedia.org/wiki/Java_Development_Kit
- https://de.wikipedia.org/wiki/WildFlyAnwendungsserver
- https://de.wikipedia.org/wiki/Google_Drive
- https://de.wikipedia.org/wiki/VMware_vSphere
- https://de.wikipedia.org/wiki/Java_(Programmiersprache)
- https://de.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- https://de.wikipedia.org/wiki/JQuery
- https://de.wikipedia.org/wiki/Ajax_(Programmierung)
- https://en.wikipedia.org/wiki/PrimeFaces
- https://de.wikipedia.org/wiki/Hibernate_(Framework)
- https://de.wikipedia.org/wiki/Data_Access_Objects

- https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange
- https://de.wikipedia.org/wiki/Android_(Betriebssystem)
- https://de.wikipedia.org/wiki/Android_Studio
- https://de.wikipedia.org/wiki/CRUD
- https://de.wikipedia.org/wiki/Data_Access_Object
- https://de.wikipedia.org/wiki/Eclipse_(IDE)
- https://de.wikipedia.org/wiki/European_Article_Number
- https://de.wikipedia.org/wiki/Extensible_Markup_Language
- https://de.wikipedia.org/wiki/JavaScript_Object_Notation
- https://de.wikipedia.org/wiki/Java_(Programmiersprache)
- https://de.wikipedia.org/wiki/Liste_von_Android-Versionen
- https://de.wikipedia.org/wiki/Microsoft_Windows_Server_2012#Windows_Server_2012_R2
- https://de.wikipedia.org/wiki/MySQL
- https://de.wikipedia.org/wiki/MySQL_Workbench
- https://de.wikipedia.org/wiki/Objektrelationale_Abbildung
- https://de.wikipedia.org/wiki/PHP
- https://de.wikipedia.org/wiki/Reflexion_(Programmierung)
- https://de.wikipedia.org/wiki/Strichcode
- https://de.wikipedia.org/wiki/Swift_(Programmiersprache)
- https://de.wikipedia.org/wiki/UTF-8
- https://de.wikipedia.org/wiki/VMware
- https://de.wikipedia.org/wiki/XAMPP
- https://de.wikipedia.org/wiki/Xcode
- https://en.wikipedia.org/wiki/IOS
- https://en.wikipedia.org/wiki/QR_code
- https://en.wikipedia.org/wiki/ZBar
- https://simple.wikipedia.org/wiki/ASCII
- https://de.wikipedia.org/wiki/Cascading_Style_Sheets
- https://de.wikipedia.org/wiki/JavaScript
- https://de.wikipedia.org/wiki/Hypertext_Markup_Language

10.2 externe Bilder

- http://cssmatic.com
- http://i.stack.imgur.com/Ck46r.png
- https://docs.oracle.com/javaee/5/tutorial/doc/bnaqq.html
- http://goqr.me/de/
- https://i-msdn.sec.s-msft.com
- http://www.apple.com/ipad/
- http://www.apple.com/ipod-touch/
- http://www.apple.com/macbook
- https://www.google.com/design/spec/material-design/introduction.html
- https://www.google.com/design/spec/what-is-material/environment.html#

- https://www.google.com/design/spec/what-is-material/environment.html#environment-3d-world
- https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html
- http://core0.staticworld.net/images/article/2012/10/m5-581t20win820left20facin-100008106-orig.jpg

10.3 Codequellen

- https://github.com/selwakowski/pracainz/blob/master/PracaInz_v01/src/com/howtodoinjava/hashing/password/demo/advanced/ReallyStrongSecuredPassword.java
- http://www.journaldev.com/7252/jsf-authentication-login-logout-database-example
- http://www.wimagguc.com/2013/06/jquery-latitude-and-longitude-picker-gmaps/
- https://css-tricks.com/responsive-data-tables/
- http://www.mkyong.com/jsf2/jsf-2-templating-with-facelets-example/
- https://developers.google.com/chart/infographics/docs/qr_codes
- https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSNotificationCenter_Class/index.html
- https://developer.apple.com/library/mac/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html

11 Abbildungsverzeichnis