HTBLA IMST

Aufbaulehrgang für Informatik

Lieferrex

Individualisierbare Bestellplattform für Restaurants mit Liefer-/Abholservice

Ersteller: Niklas Heim, Burak Eraslan, Michael Bogensberger, Liuming Xia, Julian Meilinger

Datum: 18.10.2021

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbst verfasst und keine anderen als die angeführten Behelfe verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Ich bin damit einverstanden, dass meine Arbeit öffentlich zugänglich gemacht wird.

_____
Ort, Datum

_____ _____
Niklas Heim Julian Meilinger

_____ _____
Michael Bogensberger Liuming Xia

Vorwort

[Ein großer Dank gilt allen, …] Text.

Kurzfassung

Die Corona-Pandemie, sowie Faktoren wie beispielsweise die Digitalisierung, machen es Restaurants immer schwerer ohne eine digitale Präsenz zu existieren. Immer mehr Menschen sehnen sich danach, bequem von der Couch aus Essen bestellen zu können. Das Problem ist, dass viele Besitzer von Restaurants oder Schnell-Imbissen meist keine Möglichkeit besitzen, Online-Bestellungen anzunehmen. Dies wird derzeit noch oft umständlich über Telefon abgewickelt. Durch unser entwickeltes System ist es einem Resteraunt oder beispielsweise einem Schnell-Imbiss möglich, sich rasch eine Internetpräsenz mit Bestell- sowie Abholfunktionen aufzubauen. Sie können sich schnell und ohne Vorkenntnisse eine eigens konfigurierte Webseite nach ihren eigenen Designvorstellungen erstellen. Des Weiteren lassen sich Produkte beziehungsweise Gerichte zur Abholung oder zur Lieferung über die eigene Webseite anbieten. Zudem wird das Hosting der Webseite automatisch und ohne viel Aufwand erledigt.

Zur Umsetzung jenes Systems wird ein bewährter Technologie-Stack verwendet. Das Backend wird mittels Spring Boot umgesetzt. Ein beliebtes Java Framework für Enterprise Applikationen. Für das Frontend wird das auf dem Material Design Konzept basierende Framework Materialize verwendet. Als Verbindungsstück jener Technologien wird Thymeleaf verwendet.

Abstract

[in englisch] Text.

Inhaltsverzeichnis

1 Einleitung

[Allgemeine Beschreibung wie die Diplomarbeit zu Stande gekommen ist]

….

2 Projektmanagement

[Einführende Worte zum Inhalt des Kapitels Projektmanagement] Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text.

2.1 Metainformationen

2.1.1 Projektteam

- [Auflistung der Team-Mitglieder und ihrer Verantwortungsbereiche] Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
- Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
- Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text

2.1.2 Projektbetreuer

- [Auflistung der Projekt-Betreuer] Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text
- Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text

2.1.3 Projektpartner

- [Firmenname, Adresse und Ansprechperson des Projektbetreuers] Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text

2.2 Vorerhebungen

[Einführende Worte zum Inhalt des Abschnitts Vorerhebungen] Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text.

…

2.2.1 Ist-Zustand

Seit der Corona Zeit gab es schon mehrere Lockdowns, Lokale wurden zugesperrt und konnten nicht vor Ort ihre Kunden bedienen. Die gesetzliche Lage verschärft sich für die Gastronomie in dieser Zeit, die Regelung besagt, dass die Lieferung von Waren erlaubt ist und auch empfohlen wird. Viele Besitzer von Restaurants oder einem Schnell-Imbissen besitzen meist keine Möglichkeit, Online-Bestellungen anzunehmen. Dies wird derzeit noch oft umständlich über Telefon abgewickelt. Durchs Telefonieren können häufig Missverständnisse entstehen und dadurch können die Bestellungen nicht eindeutig identifiziert oder wie gewünscht ausgeführt werden. Mithilfe des Internets und eines entwickelten Systems können Bestellung leicht und detailreich eingereicht werden.

2.2.2 Soll-Zustand

In der Zeit, in dem das Internet keine Neuheit ist und die meisten Dinge über das World Wide Web abgewickelt wird, werden auch die meisten Lieferdienste über das Internet Angeboten. Viele Klein-Restaurants können mithilfe des neu entwickelten Systems schnell und ohne Vorkenntnisse eine eigens konfigurierte Webseite nach ihren eigenen Designvorstellungen erstellen. Das Hosting der Webseite wird automatisch und ohne viel Aufwand erledigt.

2.2.3 Projektumfeldanalyse

Im Rahmen des Projektes hat das Projektteam mit vielen verschiedenen Personen zu tun. Diese können und werden das Projekt positiv oder gar negativ beeinflussen. Es wird ermittelt, wer genau diese Stakeholder sind, welchen Einfluss sie haben, wie nahe sie zum Projekt stehen und welche Maßnahmen getroffen werden, um diese abzuholen. In *Tabelle 1: Stakeholder* werden alle Stakeholder mit ihrer Einstellung, ihrem Einfluss und ihrer Nähe (im Bereich von 1 bis 10) zum Projekt aufgelistet.

Stakeholder	**Einstellung**	**Einfluss**	**Nähe**
Projektmitglieder	Positiv	8	9
Auftraggeber	Positiv	3	6
Zukünftige Nutzer (z. B. Koch, Restaurantbesitzer)	Positiv	2	3
Betreuer	Neutral	7	8
Restaurantkunde	Positiv	2	2
Mitbewerber	Negativ	1	1

Tabelle 1: Stakeholder

Im Folgenden werden die einzelnen Stakeholder beschrieben:

Ein Stakeholder ist das Projektteam, es ist hauptsätzlich für das Projekt verantwortlich und trifft viele Entscheidungen, wie die Planung, Umsetzung und Einsatz von verschiedenen Technologien. Es ist wichtig, eine gute Kommunikation und Planung im Team zu halten, damit jedes Mitglied immer auf dem gleichen Stand ist und so gemeinsam, zielstrebig am Projekt arbeiten können.

Einen hohen Einfluss auf das Projekt hat der Auftraggeber. Er bestimmt die grundliegenden Rahmenbedingungen und Anforderung des Projektes und wird anschließend durch das Ergebnis des Projektes vermarktet.

Der Projektbetreuer ist ein weiterer Stakeholder. Er betreut das Projekt und unterstützt das Team bei Entscheidungen und kontrolliert den Fortschritt. Der Betreuer wird regelmäßig durch Besprechungen über das Projekt informiert.

Zukünftige Nutzer werden leichten Einfluss auf das Projekt haben. Sie sind die Restaurantbesitzer und Köche, die das entwickelte System zur Vermarktung oder zum Planen ihrer Arbeitsabläufe verwenden werden.

Auch Restaurantkunden werden als Stakeholder definiert. Sie können das Ergebnis des Projektes verwenden, um bei den verschiedenen Restaurants, die das System verwenden, bestellen.

Das Projekt hat auch einen negativ gestimmten Stakeholder. Mitbewerber beziehungsweise Konkurrenten, die ähnliche Systeme anbieten und durch unser Projekt möglicherweise Kunden verlieren.

Hier in *Abbildung 1: Stakeholder grafisch* werden die Stakeholder grafisch dargestellt. Die dazugehörige Legende befindet sich unterhalb, in *Tabelle 2: Legende Stakeholder grafisch*.

Ihre Nähe zum Projekt wird als Abstand zum mittleren Kreis dargestellt. Deren Einstellung wird als Pfeil oder Waage interpretiert. Bei einer positiven Einstellung zeigt der Pfeil nach oben, bei einer negativen nach unten und bei einer Waage neutral. Der Einfluss auf das Projekt wird durch Farben veranschaulicht. Je höher der Einfluss ist, desto mehr ist der Stakeholder rot gefärbt. Es wird auch über Sprechblasen und Linien angezeigt, welche Stakeholder miteinander kommunizieren.

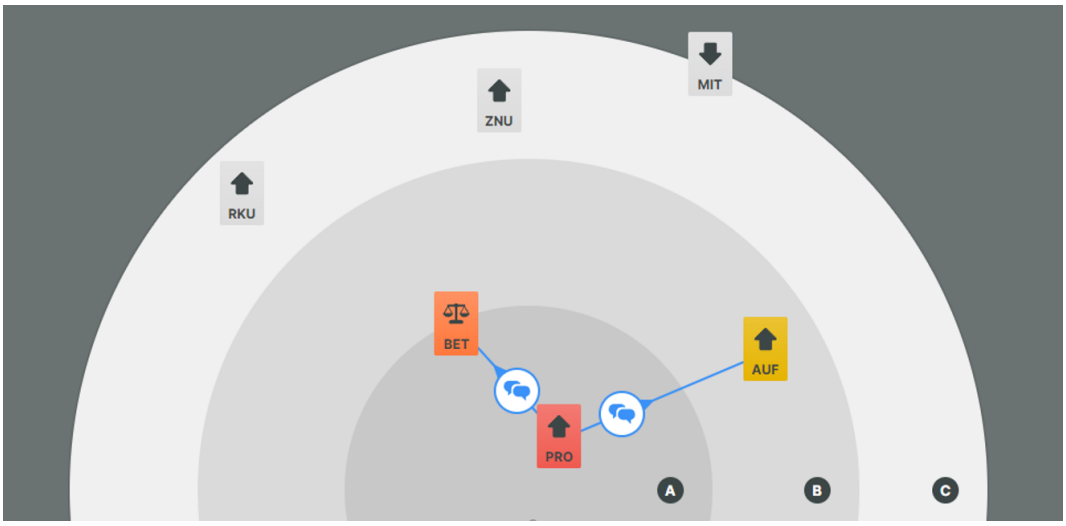

Abbildung 1: Stakeholder grafisch

Folgende Tabelle dient als Legende für die Stakeholderabbildung.

Symbol	Bedeutung
PRO	Projektmitglied
BET	Betreuer
AUF	Auftraggeber
ZNU	Zukünftige Nutzer
RKU	Restaurantkunde
MIT	Mitbewerber
Pfeil nach oben	Positive Einstellung
Pfeil nach unten	Negative Einstellung
Wage	Neutrale Einstellung
A	Hohe Nähe
B	Mittlere Nähe
C	Geringe Nähe
Nachrichtensymbol	Kommunizieren miteinander

Tabelle 2: Legende Stakeholder grafisch

2.2.4 Maßnahmen

Hier in Tabelle 3: Stakeholder Maßnahmen werden die Interessen der verschiedenen Stakeholder und einige Maßnahmen angeführt, um die verschiedenen Stakeholder besser zu stimmen und sie abzuholen.

Stakeholder	Beschreibung der Interessen	Maßnahmen
Projektmitglieder	Entwicklung des Endproduktes mit gewünschter Qualität	Gutes Klima im Projektteam sowie ein guter Informationsaustausch
Auftraggeber	Endprodukt mit gewünschten Funktionen und gewünschter Qualität	Regelmäßiger Informationsaustausch
Zukünftige Nutzer (z. B. Koch, Restaurantbesitzer)	Vereinfachte Prozesse (z. B. Leichtere Vertragsabwicklung, Leichtere Aufnahme und Verarbeitung von Bestellprozessen)	Möglichst gute Qualität des Endproduktes, sowie einfache Bedienbarkeit.
Betreuer	Bestmögliches Projekt sowie einen nicht zu großen Unterstützungsaufwand	Guter Informationsaustausch der Projektmitglieder sowie einen gelegentlichen Austausch mit dem Projektbetreuer
Restaurantkunde	Möchte schneller Restaurants in der Nähe finden sowie leicht und flexibel Essen bestellen.	Das Endprodukt möglichst intuitiv bedienbar gestalten.
Mitbewerber	Ist negativ gestimmt. Möchte nicht, dass das Projekt erfolgreich ist.	Geheimhaltung des Projektes

Tabelle 3: Stakeholder Maßnahmen

2.2.5 Risikoanalyse

Im Folgenden werden alle Risiken ermittelt, die beim Projekt auftreten können. Jedes Risiko erhält einen Titel, Status und eine Kategorie. Des Weiteren wird ermittelt, wie wahrscheinlich das Risiko eintreten kann und welche Folgen es mit sich bringt. Anschließend werden Maßnahmen definiert, um die Risiken zu überwachen oder eliminieren. Anhand eines Risikoportfolios werden ermittelten Risiken tabellarisch dargestellt. In den Tabellen (*Tabelle 4: Risikoportfolio Teil 1* und *Tabelle 5 Risikoportfolio Teil 2*) werden die einzelnen Risiken aufgelistet. Die Risiken in den jeweiligen Tabellen sind durch die Nummerierung (Nr.) erkennbar.

Die Risiken werden von 1 bis 11 durchnummeriert und erhalten jeweils einen Status der entweder mit *eliminiert* oder *überwacht* benannt wird. Zudem gibt es jeweils die Kategorien „menschlich/kulturell“, „technische/produktbezogen“, „wirtschaftlich“ und „politisch“. Dazu ist ein kurzer Titel zum Risiko, die Folgen des jeweiligen Risikos (wenn nichts dagegen unternommen wird) und dazu passende Gegensteuerungsmaßnahmen zu finden. Zudem finden sich Eintrittswahrscheinlichkeit, Auswirkungsgrad und Risikopotential wieder. Das Risikopotential ist die Multiplikation von Eintrittswahrscheinlichkeit und Auswirkungsgrad. Sie beschreibt wie schwerwiegend ein Risiko als Ganzes ist.

Nr.	Risikotitel	Status	Kategorie	Folgen des Risikos
1	Datenverlust	Eliminiert	menschlich/ kulturell	Projektfortschritt wird zurückgeworfen
2	Endprodukt entspricht nicht den Erwartungen des Auftraggebers	Überwacht	menschlich/ kulturell	Endprodukt kann vom Auftraggeber nicht verwendet werden
3	Anwendung spärlich zu bedienen	Überwacht	technisch/ produktbezogen	Anwender sind unzufrieden
4	Projektmitglied fällt aus	Überwacht	menschlich/ kulturell	Geplante Projektumsetzung im gleichen Ausmaß nicht mehr möglich
5	Projektmitglied führt Arbeitsaufträge nur mangelhaft aus	Überwacht	menschlich/ kulturell	Geplante Projektumsetzung im gleichen Ausmaß nicht mehr möglich
6	Fehleinschätzung des Aufwands	Überwacht	menschlich/ kulturell	Projektziele können nicht erreicht werden
7	Ausfall des Hosting Anbieters	Eliminiert	technisch/ produktbezogen	Anwendung kann nicht verwendet werden
8	Arbeitspakete der Projektanten ergänzen sich nicht	Überwacht	menschlich/ kulturell	Projektergebnis entspricht nicht den Erwartungen
9	Schnittstelle nicht kompatibel	Überwacht	technisch/ produktbezogen	Geplante Funktionalität nicht umsetzbar
10	Ähnliche Anwendung erscheint während der Projektentwicklung auf dem Markt	Überwacht	wirtschaftlich	Produkt des Mitbewerbers wird vorgezogen
11	Rechtliche Anforderungen werden nicht erfüllt	Überwacht	politisch	Anwendung darf nicht veröffentlicht werden

Tabelle 4: Risikoportfolio Teil 1

Nr.	Risikotitel	Eintritts-wahrschein-lichkeit	Auswirkung	Risiko-potential	Gegensteuerungs-maßnahme
1	Datenverlust	2	3	6	Projekt wird in GitHub gespeichert
2	Endprodukt entspricht nicht den Erwartungen des Auftraggebers	1	3	3	Regelmäßiger Informationsaustausch
3	Anwendung spärlich zu bedienen	1	2	2	Auf Designstandards achten
4	Projektmitglied fällt aus	1	3	3	Gutes Teambuilding
5	Projektmitglied führt Arbeitsaufträge nur mangelhaft aus	2	2	4	Regelmäßige Statusberichte
6	Fehleinschätzung des Aufwands	2	2	4	Arbeitsaufwand kontrollieren
7	Ausfall des Hosting Anbieters	1	3	3	Vertrauenswürdigen Hosting Anbieter auswählen
8	Arbeitspakete der Projektanten ergänzen sich nicht	1	3	3	Regelmäßiger Informationsaustausch
9	Schnittstelle nicht kompatibel	1	2	2	Im Vorfeld darüber Informieren
10	Ähnliche Anwendung erscheint während der Projektentwicklung auf dem Markt	1	2	2	Marktentwicklung beobachten
11	Rechtliche Anforderungen werden nicht erfüllt	3	2	6	Über den BSI informieren

Tabelle 5 Risikoportfolio Teil 2

Folgende Tabelle dient als Legende für das Risikoportfolio.

Symbol	Bedeutung
Eintrittswahrscheinlichkeit	Gemessen von 1 bis 4, je höher desto öfter tritt das Risiko ein
Auswirkung	Gemessen von 1 bis 4, je höher, desto mehr Geld und Aufwand muss erbracht werden, um das Problem zu lösen
Risikopotential	Multiplikation der ermittelten Werte Eintrittswahrscheinlichkeit und Auswirkung, Werte 1 bis 16, entscheidet, wie schnell und sorgfältig mit dem Risiko umgegangen wird

Tabelle 6: Legende Risikoportfolio grafisch

In *Abbildung 2: Risikomatrix* werden die ermittelten Risiken in einer Matrix dargestellt. Die Risiken werden von der oberliegenden *Tabelle 4: Risikoportfolio* in die folgende Grafik überführt. Hier werden sie von R1 bis R11 durchnummeriert. Durch diese wird veranschaulicht, wo welche Risiken liegen und auf welche besonders geachtet werden sollten. Auf den beiden Achsen sind die Eintrittswahrscheinlichkeit und der Grad der Auswirkung aufgetragen. Risiken, die sich rechts oben in der Matrix befinden, sind besonders schwerwiegend. Dies wird auch durch einen immer roter werdenden Hintergrund hervorgehoben.

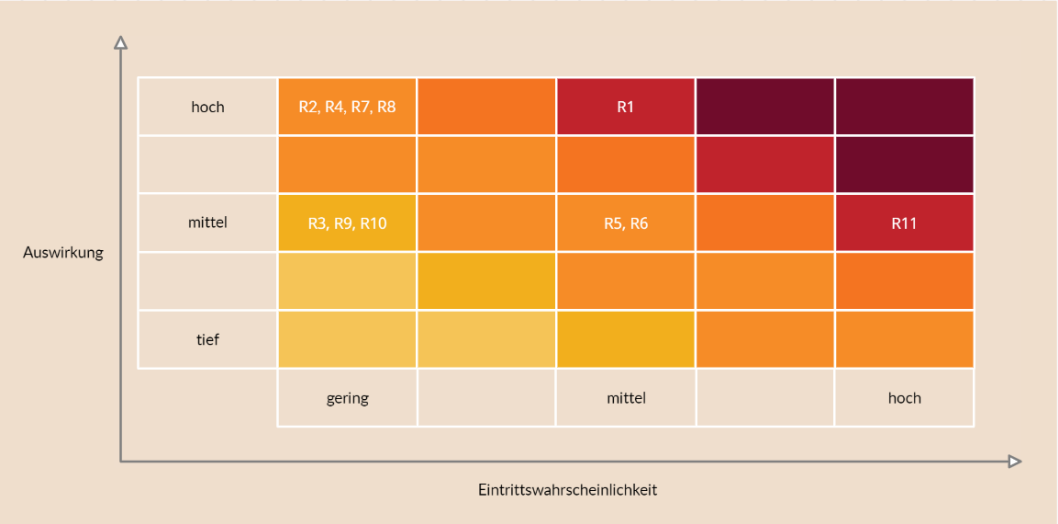

Abbildung 2: Risikomatrix

2.3 Pflichtenheft

Die folgenden Absätze konkretisieren die Pflichten des Projektteams. Dabei werden die Funktionalitäten, das Einsatzgebiet und die Ziele genau definiert.

2.3.1 Zielsetzung

Das Projektteam setzt sich die Entwicklung einer individualisierbaren Bestellplattform für die Gastronomie als Ziel. Damit sollen Restaurants die Möglichkeit haben, schnell und einfach einen Liefer-/Abholservice einzurichten. Das größte Hindernis ist dabei die Programmierung des individuell gestaltbaren Baukastensystems. Die Bestellplattform wird als Responsive Web-App veröffentlicht und folgt dem Material-Design als Formgebung. Als Schnittstelle zwischen dem Client und Server wird eine REST-API implementiert. Wunschziel ist es, dass unser Projektpartner und weitere mögliche Kandidaten die Bestellplattform verwenden und im täglichen Geschäft gebrauchen können. Die Absolvierung des Projekts hat für jeden Projektbeteiligten eine positive Auswirkung. Die Gastronomie bekommt eine weitere Option, ihre Lebensmittel Online zu vermarkten und können personalisierte Bestellplattformen erstellen, die ihrer Unternehmensphilosophie entsprechen. Die Gesellschaft hat eine weitere Möglichkeit Essen im Internet zu kaufen. Die Projektanten haben die Möglichkeit bei einem sehr erfolgreichen Abschluss des Projekts einen kleinen Nebenverdienst zu generieren und haben sich Wissen zu neuartigen Technologien in der Webentwicklung angeeignet. Das Erreichen der Mindestanforderungen ist realistisch. Das Projektteam hat die Kompetenz innerhalb des geforderten Zeitraums die Aufgaben zu bewältigen und das Projekt vorzustellen. Jeder Projektant ist motiviert, seine Aufgabenstellungen zu absolvieren und so gut wie möglich zu bearbeiten, um ein reibungsloses Zusammenspiel der Teilaufgaben zu ermöglichen. Die Diplomarbeit wird im Juni 2022 abgegeben. Ein erster Prototyp soll, bis Jänner 2022 fertiggestellt werden.

2.3.2 Produkteinsatz und Umgebung

Unsere Software soll die Arbeit in Gastronomiebetrieben vereinfachen. Deshalb versucht das Projektteam gezielt die Mensch-Computer-Interaktion so intuitiv wie möglich zu machen. Somit soll es den Mitarbeitern der Restaurants möglich sein, Onlinebestellungen schnell und einfach anzunehmen. Einen weiteren Einsatz findet unser Projekt bei allen hungrigen Personen, die sich bequem Essen bestellen möchten. Die Onlineplattform stellt dabei die besten Voraussetzungen, um eine Reibungslose Kommunikation zwischen Restaurant und Kunde zu ermöglichen.

2.3.3 Funktionalitäten

Es ist besonders wichtig, die Funktionalitäten des Projektes klar zu definieren. Diese müssen eingeteilt werden in Muss- und Kann-Funktionalitäten. Muss- Funktionalitäten müssen im Rahmen des Projektes erfüllt werden. Kann-Kriterien sind zusätzliche Funktionen, die erfüllt werden können, aber nicht müssen. Des Weiteren werden sie für eine klare Übersicht und Struktur in funktionale und nicht funktionale Anforderungen gegliedert. Funktionale Anforderungen beschreiben gewünschte Funktionalitäten. Sie beschreiben, was das entwickelte System kann. Nicht funktionale Anforderungen erhöhen die Qualität des Projektes. Die Anforderungen werden nach dem FURPS[1] System geplant. Sie beschreiben die Funktionalität, Benutzbarkeit, Zuverlässigkeit, Zuverlässigkeit und Wartbarkeit des Projektes.

Im folgenden Teil werden die Muss-Anforderungen aufgelistet. Sie beschreiben Anforderungen, die im Rahmen des Projektes erfüllt werden müssen, da sie wichtige Kernelemente dieses bilden. Sie werden in funktionale und nicht funktionale Anforderungen aufgeteilt.

- Funktional
  - Anmeldung für Benutzer der Webseite.
  - Dashboard für Restaurants für Verwaltung.
  - Erstellen von Bestellungen.
  - Anlegen neuer Gerichte.
  - Gestalten von Webseiten mit dem Baukastensystem.
  - Responsive Webseite auf allen Endgeräten.
- Nicht funktional
  - Das Design der Webseite ist ansprechend.
  - Die Webseite ist gut erweiterbar.
  - Das System ist gut dokumentiert.
  - Baukastensystem leicht bedienbar.

Im nächsten Teil werden die Kann-Funktionalitäten des Projektes aufgelistet. Sie können zusätzlich zu den Muss-Funktionalitäten erfüllt werden, müssen aber nicht. Auch die Kann-Funktionalitäten werden in funktionale und nicht funktionale Anforderungen aufgeteilt.

- Funktional
  - Weitere Anpassungsmöglichkeiten mit dem Baukastensystem.
  - Restaurant werden über Sub-Domains aufgerufen.
  - Online-Bezahlung über die Webseite.
- Nicht funktional
  - Die Daten sollen gut mit Backups gesichert werden.
  - Fehler der Webseite sollen leicht behoben werden können.
  - Die Webseite muss barrierefrei gestaltet sein.

2.3.4 Liefervereinbarung

…

[1] Functionality, Usability, Reliability Perfomance, Supportability

2.4 Planung

Die folgenden Absätze schildern detailliert die Planungsansätze für unser Projekt. Der Projektstrukturplan beinhaltet die wichtigsten Arbeitspakete und kategorisiert sie in dessen Phasen ein. Der Projektablaufplan übernimmt diese Arbeitspakete und ordnet sie zeitlich ein.

2.4.1 Projektstrukturplan

In der folgenden Grafik (*Abbildung 3*: Projektstrukturplan) finden Sie den zum Projekt passenden Projektstrukturplan. Wir haben uns für einen Phasen-orientierten Projektstrukturplan entschieden, da sich dadurch die Ablaufplanung leichter gestalten lässt.

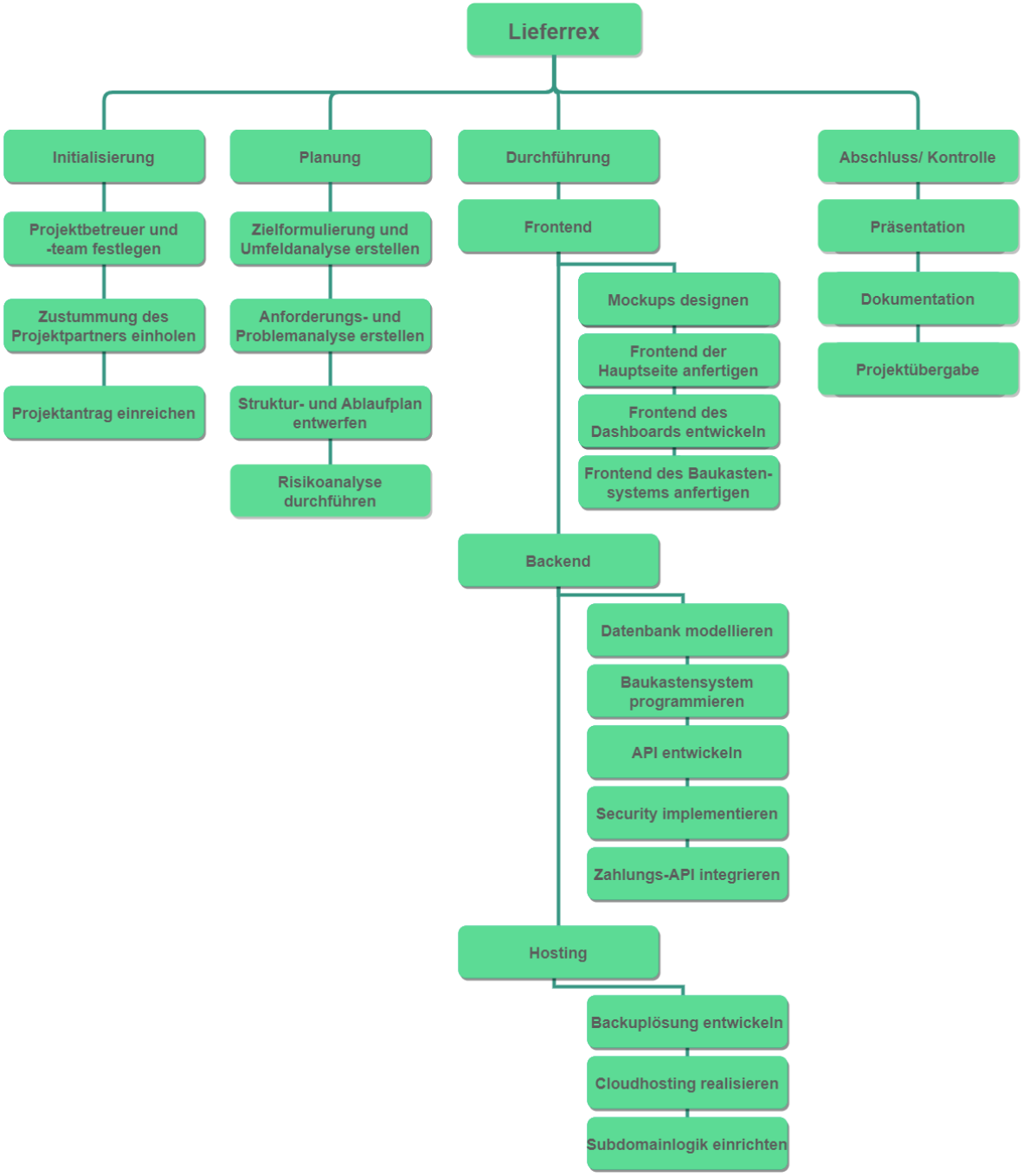

Abbildung 3: Projektstrukturplan

2.4.2 Projektablaufplan

Aus dem Projektstrukturplan wurde im nächsten Schritt ein Projektablaufplan *(Abbildung 4:* Projektablaufplan erstellt. Dieser listet alle Teilaufgaben mit Datum, benötigter Arbeitszeit und verwendeter Ressourcen auf. Auch zu sehen sind Meilensteine, die zeigen, wann wichtige Kernelemente des Projekts abgeschlossen werden.

Vorgangsname	Dauer	Anfang	Ende
Voranalysen	**15 Tage**	**Mit 22.09.21**	**Die 12.10.21**
Projektzielplan	2 Tage	Mit 22.09.21	Don 23.09.21
Projektumfeldanalyse	2 Tage	Die 28.09.21	Mit 29.09.21
Risikoanalyse	3 Tage	Die 05.10.21	Don 07.10.21
Abschluss Analyse	0 Tage	Die 12.10.21	Die 12.10.21
Projektplanung	**6 Tage**	**Don 21.10.21**	**Don 28.10.21**
Projektstrukturplan	2 Tage	Don 21.10.21	Fre 22.10.21
Anforderungsanalyse	2 Tage	Die 26.10.21	Mit 27.10.21
Ablaufplan	1 Tag	Don 28.10.21	Don 28.10.21
Abschluss Projektplanung	0 Tage	Don 28.10.21	Don 28.10.21
Durchfuehrung	**74 Tage**	**Mit 10.11.21**	**Mon 21.02.22**
Frontend	**33 Tage**	**Mit 10.11.21**	**Fre 24.12.21**
Mockups erstellen	7 Tage	Mit 10.11.21	Don 18.11.21
Frontend der Hauptseite anfertigen	8 Tage	Don 25.11.21	Mon 06.12.21
Frontend des Dashboard angertigen	6 Tage	Don 09.12.21	Don 16.12.21
Frontend des Baukastensystems anfertigen	6 Tage	Fre 17.12.21	Fre 24.12.21
Abschluss Frontend	0 Tage	Fre 24.12.21	Fre 24.12.21
Backend	**47 Tage**	**Mit 10.11.21**	**Don 13.01.22**
Datenbank modellieren	10 Tage	Mit 10.11.21	Die 23.11.21
Baukastensystem programmieren	16 Tage	Mit 08.12.21	Mit 29.12.21
API entwickeln	14 Tage	Don 09.12.21	Die 28.12.21
Security implementieren	7 Tage	Mit 05.01.22	Don 13.01.22
Zahlungs-API integrieren	4 Tage	Don 06.01.22	Die 11.01.22
Abschluss Backend	0 Tage	Don 13.01.22	Don 13.01.22
Hosting	**22 Tage**	**Fre 21.01.22**	**Mon 21.02.22**
Backuploesung entwickeln	5 Tage	Fre 21.01.22	Don 27.01.22
Cloudhosting realisieren	4 Tage	Fre 04.02.22	Mit 09.02.22
Subdomainlogik einrichten	6 Tage	Mon 14.02.22	Mon 21.02.22
Abschluss Hosting	0 Tage	Mon 21.02.22	Mon 21.02.22
Abschluss Durchfuehrung	0 Tage	Mon 21.02.22	Mon 21.02.22
Kontrolle des Projekts	3 Tage	Die 22.02.22	Don 24.02.22
Praesentation	0 Tage	Mit 06.07.22	Mit 06.07.22
Dokumentation	200,38 Tage	Mit 22.09.21	Fre 01.07.22
Projektuebergabe	0 Tage	Fre 01.07.22	Fre 01.07.22

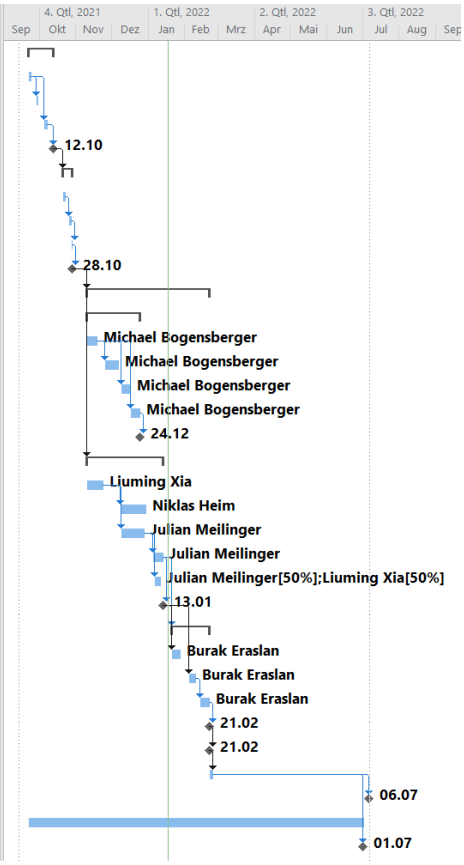

Abbildung 4: Projektablaufplan

2.4.3 Abnahmekriterien

…

2.4.4 Evaluationsplan

3 Vorstellung des Produkts

Die Frontendstruktur unseres Projektes ist grundsätzlich in drei Teile (Kundenseite, Dashboard, Baukasten) unterteilt. Auf der Kundenseite kann sich ein Kunde registrieren und dort von Restaurants Essen bestellen und sich dieses liefern lassen. Auf jener Startseite kann man nun nach Restaurants suchen. Tut man dies, gelangt man auf eine Übersicht mit den gefundenen Restaurants. Von dort aus kann man nun einen Bestellprozess abschließen und über die Startseite seine Bestellungen überwachen.

Das Dashboard ist für das Resteraunt gedacht. Dort kann man Bestellungen überwachen, Statistiken einsehen sowie Einstellungen ändern. Betritt man das Dashboard, so gelangt man zu einer Übersicht über das Resteraunt. Dort sind einfache Statistiken, die letzten Bestellungen sowie eine Hilfe zu sehen. Von dort aus kann man nun auf die jeweiligen Seiten navigieren.

Auf der Baukastenseite kann das Resteraunt sich nun eine eigene Webseite zusammenbauen. Auf jene Seite gelangt man über das Dashboard.

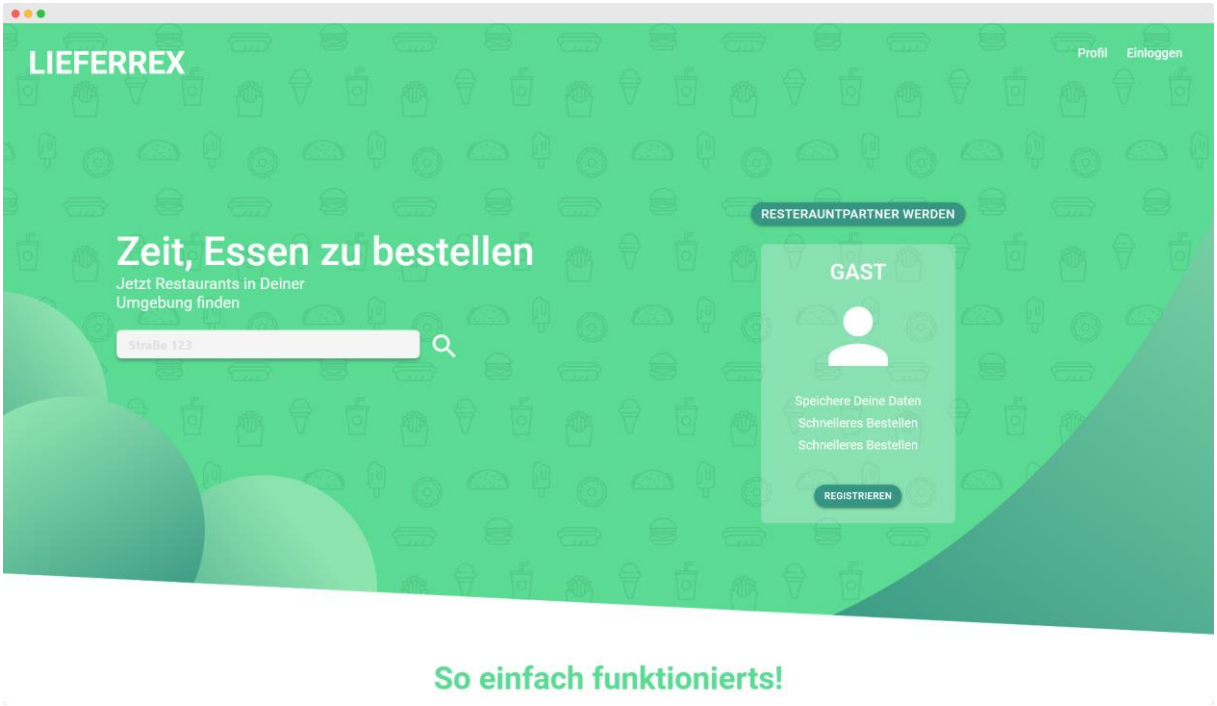

Abbildung 5 Mockup der Hompage

Im Baukasten angelangt können vom Mandanten allgemeine Einstellungen wie Akzentfarbe und weiteres getroffen werden. Als nächstes muss ein Layout aus vielen vordefinierten gewählt werden. Das Layout entscheidet, in welcher Form und Anordnung die kommenden Module dargestellt werden. Wurde ein Layout gewählt, wird dieses angezeigt. Über die einzelnen Kacheln mit Plussymbolen werden nun Module hinzugefügt. Es öffnet sich ein Pop-Up das alle verfügbaren Module auflistet. Verfügbare Module beinhalten beispielsweise: Text, Überschrift, Bild, Speisekarte, Karte, Öffnungszeiten und vieles mehr. Wurde ein Modul gewählt müssen entsprechende Daten eingegeben werden. Es können auch weitere Seiten wie Kontakt, Bildergalerie und mehr in der Navigationsleiste hinzugefügt werden. Des Weiteren kann ein Mandant zusätzliche Seiten wie „Über uns“, Bildergalerie und weiteres haben.

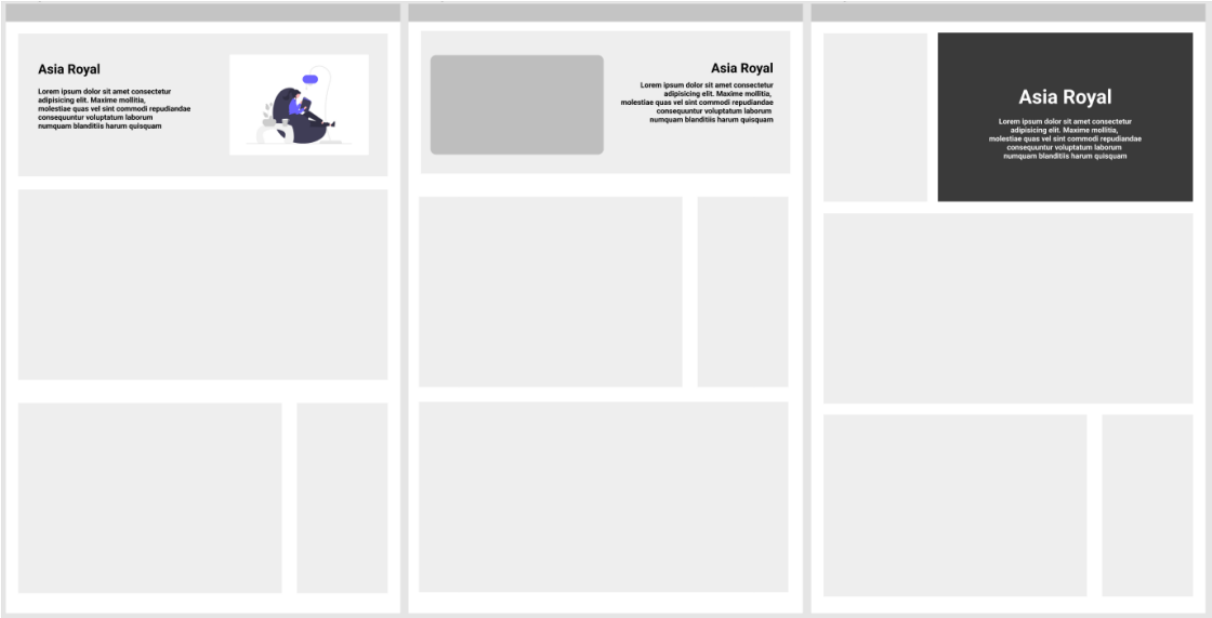

Abbildung 6 Mockup vom Baukastensystem

4 Eingesetzte Technologien

Im Zuge der Planung sowie der Entwicklung unseres Projektes kommen selbstverständlich auch Werkzeuge für zum Beispiel die Erstellung von Grafiken und Modell oder der Versionierung von Quellcode zum Einsatz. In folgendem Abschnitt sind die von uns verwendeten Werkzeuge aufgelistet.

4.1 Materialize

Materialize ist ein Frontend-Framework basierend auf der von Google entwickelten Designsprache Material Design. Es ist ein Framework, dass sich sehr gut für responsives Webdesign eignet. Das Projektteam hat sich für Materialize entschieden, da man relativ unkompliziert ansprechende User Interfaces nach dem Material Design Prinzip erstellen kann. Zudem haben uns die Komponenten für die mobile Darstellung überzeugt. Materialize wurde in unserem Projekt verwendet, um das Frontend der Hauptseite sowie des Baukastensystems zu realisieren. Eine sehr ähnliche Alternative wäre zum Beispiel das von Google entworfene Frontend-Framework Material Design. Dies richtet sich jedoch eher an JavaScript Frameworks wie Angular. (Materialize documentation, 2022)

4.2 Materialize Stepper

Der Materialize Stepper ist ein Plugin für das von uns verwendete Framework Materialize. Das Plugin erweitert das Framework um einen Stepper. Stepper sind Komponenten die einen verbundenen Prozess in kleine Schritte beziehungsweise in kleine User Interfaces aufteilen. Dies macht einen längeren Prozess (zb. Die Resteraunt-Registrierung) übersichtlicher. Jener Stepper wurde unter anderem bei der Registrierung der Kunden sowie der Resteraunt-Partner verwendet. Als Alternative könnte man natürlich selbst ein User Interface für jenen Zweck entwerfen. Da ein Stepper jedoch nicht so einfach zu implementieren ist und es bereits für Materialize eine Stepper Erweiterung gibt hat sich das Projektteam für jene Erweiterung entschieden. (Materialize Stepper documentation, 2022)

4.3 Halfmoon

Halfmoon ist ein Frontend-Framework das sich an Bootstrap orientiert. Jedoch hat Halfmoon Features die sich exzellent für unser Projekt eignen. So hat Halfmoon unter anderem einen eingebauten Dark Mode der sich sehr leicht anpassen lässt und voll anpassbare CSS-Variablen. Halfmoon eignet sich also sehr für das Erstellen von Dashboards. Der Aufbau der Komponenten in Halfmoon ist ebenfalls sehr für die Entwicklung von Dashboards ausgerichtet. Jedoch hat Halfmoon nicht so viele Komponenten

wie beispielweise das verwandte Bootstrap. Dennoch hat sich das Projektteam aus den oben genannten Gründen für Halfmoon entschieden. Mithilfe von Halfmoon wurde das Frontend des Dashboards realisiert. (Halfmoon dokumentation, 2022)

4.4 JQuery

JQuery ist eine beliebte JavaScript-Bibliothek, die unter anderem Funktionen zur DOM-Manipulation bereitstellt. Durch JQuery lässt sich unter anderem eine Menge Code sparen. Zudem besitzt JQuery ein breitgefächertes Repertoire an standardkonformen Funktionen zur Manipulation des „Document Object Models“. Da im Projekt hin und wieder für zum Beispiel die Validierung der Öffnungszeiten oder der asynchronen Änderung von Daten der DOM bearbeitet werden muss, kommt einem JQuery sehr gelegen. Zudem haben alle Mitglieder des Projektteams bereits Erfahrungen mit JQuery gesammelt. Deshalb fiel die Wahl für uns auf JQuery. Eine eher unpraktikable Alternative wäre es, jene Funktionalitäten in Vanilla-JavaScript zu lösen. Man könnte aber auch JavaScript Frameworks wie zum Beispiel VueJs oder Angular verwenden. Jedoch würden diese Frameworks die Anforderungen übertreffen und einen zu hohen Zeitaufwand verursachen. (JQuery documentation, 2022)

4.5 Spring Boot

Spring Boot ist ein Java-basiertes Open-Source-Framework, dass die Java Entwicklung von Business-Applikationen stark vereinfacht. Es basiert auf dem Spring Framework und beinhaltet schon vorab einige Konfigurationen, dass die Entwicklung zusätzlich benutzerfreundlicher und einfacher macht. Es wird die Version 2.6.0 von Spring Boot verwendet, da es zum Zeitpunkt der Erstellung des Projekts die beste und aktuelle Version war. (Baeldung, 2022)

Durch die vereinfachte Maven-Konfiguration der „Starter“-POMs (Project Object Models) ist es ganz einfach Drittanbieter-Bibliotheken – auch Dependencys genannt – hinzuzufügen. Dadurch ist es möglich die einzelne Microservices – die Notwendig für eine Business-Applikation sind – ohne größeren Aufwand zu implementieren.

Mithilfe der Starter-Web Bibliothek ist es möglich einen Apache Tomcat Webserver aufzusetzen. Dieser verwendet den Port 8080 und läuft sobald man die Applikation startet. Desweiterem inkludiert es die Möglichkeit REST-Schnittstellen zu erstellen. Durch diese Schnittstelle kann die Applikation nach außen kommunizieren.

Da im Frontend das Thymleaf verwendet wird auch die „spring-boot-starter-thymeleaf“ Dependency in die pom.xml verwendet. Somit kann man Thymleaf-Models über dem Webinterface mit senden.

Die “mysql-connector-java” und die „spring-boot-starter-data-jpa“ Bibliothek ermöglichen es in einem vereinfachten Format mit einer MySql Datenbank zu kommunizieren.

Mit der Lombock-Bibliothek kann man die Länge des Quellcodes erheblich verkürzen und die Lesbarkeit erhöhen. Ziel ist somit die Vermeidung von repetetivem Boilerplate-Code. Mit Annotationen kann man gewisse Parameter mitgeben, die dann zur Laufzeit in validen Java-Code umgewandelt wird.

Die “spring-boot-starter-security” Bibliothek ermöglicht es Authentifizierungs-Verfahren ganz einfach in Spring Boot zu implementieren. Man kann somit gewisse Ressourcen auf dem Webserver schützen, und nur mit autorisiertem Zugang freigeben.

4.6 MySQL

MySQL ist ein relationales Datenbanksystem. Das Datenbanksystem ist für die Datenspeicherung von Webservices zuständig. Mithilfe der Structured Query Language können Daten einfach abgefragt werden. In der SQL-Datenbank werden die Schemen für die Daten Tabellarisch angelegt, das heißt es

gibt in dieser Tabelle Spalten sowie Zeilen. Die Spalten werden mithilfe eines Namens und eines Typen erstellt. Eine Spalte beinhaltet den Primären Schlüssel, dieser Primär Schlüssel ist eindeutig und kann nicht für dieselbe Tabelle verwendet werden. Der Primär Schlüssel wird dafür verwendet um Datensätze eindeutig zu identifizieren. Das Datenbanksystem MYSQL wurde im Unterricht behandelt, dadurch wurde das Aneignen aus verschiedenen Quellen beschleunigt. Es gibt noch andere Datenbanksysteme, wie Key-Value oder Dokumentenbasierende Datenbanksysteme, die wurden jedoch zum Zeitpunkt des Projektstartes noch nicht behandelt. Die Entscheidung viel dadurch sehr schnell auf MySQL.

4.7 Thymeleaf

Thymeleaf ist eine Java XML/XHTML/HTML5 template engine. Das Hauptziel von Thymeleaf ist, dass HTML im Browser korrekt angezeigt werden kann und auch als statische Prototypen funktionieren. Im Template Mode können folgende Prozesse verwendet werden:

- HTML
- XML
- TEXT
- JAVASCRIPT
- CSS
- RAW

Das kleinste Detail kann mit Thymeleaf angepasst und realisiert werden. In der Kernbibliothek von Thymeleaf ist standardmäßig ein Dialekt namens Standard Dialect integriert, dieser Dialect sollte für die meisten Benutzer ausreichend sein. Thymeleaf ist für die Darstellung auf der Webseite zuständig, z.B. sind die Anzahl der Gerichte sowie den Namen des Gerichtes dynamisch dargestellt. Mithilfe von Maven kann die Library schnell aus dem Internet geholt werden. Spring Boot bietet schon von Anfang ein Webkit an, im Webkit ist Thymleaf beinhaltet. Ein typisches Merkmal für Themeleaf ist das im Quelltext th:action, th:field oder th:text oft vorkommen. Diese Merkmale sind Syntaxen, dieser Syntax ist speziell bei Thymeleaf dadurch können Texte oder Felder übergeben werden. (Thymeleaf documentation, 2022)

4.8 Spring Security

Spring Security Authentifizierungs- und Zugriffskontroll-Framework mit man den Zugriff auf Ressourcen des Webservers konfigurieren kann. (Baeldung, 2022)

Der Authentifizierungsprozess basiert auf verschiedenen Filtern und deren Interaktion untereinander. Wenn ein User seine Authentifizierung über einen Browser Request sendet, folgt entweder eine erfolgreiche Anmeldung oder in einem HTTP-403-Error. Falls die Autorisierung erfolgreich war, wird am Webserver eine Session mit einem bestimmten Token erstellt. Dieser Token wird als Response dem Client zurückgesendet und als Cookie gespeichert. Immer wenn der Client nun eine http-Anfrage macht, muss dieser Token mitgeschickt werden, um den User authentifizieren.

4.9 Visual Paradigm

Bei Visual Paradigm handelt es sich um ein sehr weit verbreitetes und bekanntes Werkzeug zum Erstellen von UML Grafiken. Visual Paradigm bietet eine frei verwendbare Version (Community Edition) an. Da öfters UML-Grafiken für zum Beispiel das ER-Modell, Use-Cases oder Klassendiagramme angefertigt werden müssen, ist man auf so ein Tool angewiesen. Visual Paradigm lässt sich intuitiv steuern und bietet zudem sehr viele verschiedene Vorlagen an. Deshalb viel für uns die Wahl eindeutig auf Visual Paradigm. Alternativen wären hier unter anderem Lucidchard, Draw.io oder Dia. Jedoch bieten die genannten Tools keine so intuitive Handhabung an. Zudem fehlt es einigen Alternativen an Features beziehungsweise Vorlagen.

4.10 PayPal API

Damit Kunden ihre Bestellungen bezahlen können verwenden wir die PayPal API. Somit können wir den Geldverkehr zwischen Kunde und Restaurant ohne großes Risiko realisieren.

Währenden der Entwicklungsphase wird der Sandbox-Modus von PayPal verwendet, damit man unbeschwert Geldbeträge versenden kann. Dabei muss in die entsprechende Dependency eingebunden werden. Über die REST API können die Geldbeträge gesendet werden.

PayPal wird verwendet, da es die einfachste Möglichkeit eine standardisierte Zahlungsmethode zu implementieren ist. (PayPal API documentation, 2022)

4.11 Google Maps API

Restaurant sollen ihren Standort auf einer Karte darstellen können. Um dies zu ermöglichen, wird die Google Maps API verwendet. Des Weiteren soll der Benutzer Restaurant in unmittelbarer Nähe finden können. Hierbei müssen die verschiedenen Distanzen berechnet werden.

Mithilfe der API sollen Karten dargestellt, Adressen in Geodaten übersetzt und Distanzen berechnet werden. (Google Maps Platform, 2022)

4.12 GIT

GIT (Global Information Tracker) ist eine Software, die das Protokollieren von Projekten-Versionen ermöglicht. Mit GIT werden stets alle Änderungen am Projekt lokal gespeichert. Hiermit ist es möglich, in ältere Versionen einzusehen oder auf diese zurückzuspringen. Über GitHub kann das Projekt auch in der Cloud gespeichert werden. Dies ermöglicht eine gemeinsame Arbeit des Projektteams. GIT stellt hier die Änderungen der verschiedenen Mitglieder dar, um alle auf dem neuesten Stand zu halten. Heutzutage werden Projekte fast ausschließlich mit GIT verwaltet.

4.13 IntelliJ

IntelliJ IDEA ist eine IDE (Integrated Development Environment) von JetBrains für Java. Diese Software wurde bereits in Unterricht für verschiedenste Projekte verwendet. Es wird hierbei die Ultimate Edition benutzt, da diese eine höhere Funktionalität bietet. Diese kann über eine Schullizenz erworben werden. Im Rahmen der Diplomarbeit wird diese hautsächlich verwendet, um das Backend zu implementieren.

Diese IDE bietet viele Funktionen, die das Schreiben von Code erleichtern. Das Verwalten von Maven-Projekten ist in IntelliJ IDEA integriert. Maven-Projekte können einfach erstellt und verwaltet werden. Eine automatische Code-Vervollständigung, nimmt dem Programmierer Arbeit beim Schreiben von Variablen oder großen Strukturen ab. IntelliJ IDEA bietet auch etwaige Keyboard-Shortcuts, mit denen das Schreiben von Code optimiert werden kann. Auch ist GIT in die IDE miteingebaut. Mithilfe von GIT kann in IntelliJ IDEA das Projekt schnell verwaltet werden. Änderungen können mithilfe weniger Knopfdrücke gespeichert werden.

Als Alternative zu IntelliJ IDEA gibt es Visual Studio Code von Microsoft. Visual Studio Code bietet vergleichbare Funktionen, muss aber vom Benutzer vorerst konfiguriert werden. Hier müssen beispielsweise Erweiterungen installiert werden, um die Unterstützung von Java, Spring Boot und weiterem zu ermöglichen.

5 Problemanaylse

5.1 Use-Case-Analyse

Ein Use Case ist eine Beschreibung, wie Benutzer mögliche Szenarien in einem System ausführen. Sie zeigen explizit Aktionen, aus der Sicht des Anwenders und erklären wie das System darauf reagiert.

Der Akteur des Systems ist meist eine Person, eine Rolle, eine Organisation oder ein anderes System. Dieser Akteur interagiert mit einem System, um ein bestimmtes Ziel in einer definierten Folge von Aktionen zu erreichen.

In diesem Abschnitt sind die für unser Projekt definierten User Stories als Aufzählung beschrieben. Diese sind zunächst in Kunde, Administrator, Mitarbeiter und Restaurant aufgeteilt. Der Kunde ist der normale Besucher unserer Webseite. Mit Restaurant ist der jeweilige Besitzer des Restaurants gemeint. Der Mitarbeiter ist ein Angestellter des jeweiligen Restaurants. Administratoren sind die Verwalter des gesamten Systems.

An dieser Stelle sind die User Stories für den Kunden zu sehen:

- Als Kunde möchte ich Restaurants in meiner Nähe finden, um bei diesen Essen zu bestellen.
- Als Kunde möchte ich Restaurants in meiner Nähe finden, um bei diesen Essen abzuholen.
- Als Kunde möchte ich mein Profil auf der Webseite speichern können, um Aufwand beim Bestellen zu sparen.
- Als Kunde möchte ich über die Option eines Dark-Mode verfügen, um es meiner Präferenz anzupassen.
- Als Kunde möchte ich, dass die Webseite auf sämtlichen Endgeräten verfügbar ist, um auch von unterwegs aus Essen bestellen zu können.

An dieser Stelle sind die User Stories für den Administrator zu sehen:

- Als Administrator möchte ich Restaurant und auch Kunden verwalten können, um Probleme zu beheben.
- Als Administrator möchte ich Informationen (Anzahl Kunden, Verkäufe, Restaurants, usw.) von Restaurants und Kunden erhalten können, um zu sehen, wie rentabel unser Geschäft ist.
- Als Administrator möchte ich Server verwalten können, um Probleme zu beheben oder Funktionen anzupassen.

An dieser Stelle sind die User Stories für das Restaurant zu sehen:

- Als Restaurant möchte ich Statistiken von meiner Webseite bekommen (Umsatz, Anzahl Bestellungen, Lieblingsprodukt, usw.), um mein Angebot immer besser optimieren zu können.
- Als Restaurant möchte ich eine individuelle Webseite gestalten können, um diese für Kunden attraktiver zu machen.
- Als Restaurant möchte ich Informationen über aktuelle Bestellungen erhalten, um diese zu erfüllen.
- Als Restaurant möchte ich einstellen können, welche Produkte ich anbiete, um die Bedürfnisse meiner Kunden erfüllen zu können.
- Als Restaurant möchte ich Zugriffsrechte für mein Personal anpassen, um nicht alle Informationen meinem Personal zur Verfügung stellen zu müssen.

An dieser Stelle sind die User Stories für den Mitarbeiter zu sehen:

- Als Mitarbeiter möchte ich Informationen über aktuelle Bestellungen erhalten, um diese zu erfüllen.

Im Folgenden werden die drei wichtigsten Use-Cases genauer in Form von Tabellen beschrieben. Sie beinhalten den Namen des Use-Cases, eine Beschreibung, Vorbedingungen, Essenzielle Schritte und weiteres.

In *Tabelle 7*: *Use-Case "Essen bestellen*" wird der Bestellprozess eines Kunden dargestellt. Es wird der Ablauf einer Bestellung erklärt, unter welchen Bedingungen diese erfolgen kann und welche Ausnahmefälle vorhanden sind.

NR	**UC-2021-001**	
Name	Essen bestellen	
Akteur	Kunde	
Trigger	Essen bestellen	
Kurzbeschreibung	Der Kunde kann über die Webseite Essen bestellen. Er kann seine Wahl zwischen vielen verschiedenen Restaurants treffen. Bei der Bestellung müssen Kundendaten eingegeben werden und eine erfolgreiche Zahlung vorhanden sein.	
Vorbedingung	Kunde ist angemeldet	
Essenzielle Schritte	**Intention des Benutzers**	**Reaktion des Systems**
	Kunde besucht die Webseite	Webseite zeigt Übersicht
	Kunde meldet sich an	Webseite meldet den Kunden an und erstellt eine Session
	Kunde sucht gewünschtes Restaurant	Webseite liefert eine Übersicht mit Restaurant
	Kunde wählt Gerichte	Webseite liefert eine Übersicht mit Gerichten
	Kunde gibt Lieferadresse und Zahlung ein	Webseite erstellt Bestellung
Ausnahmefälle	Webseite nicht verfügbar	
Zeitverhalten	Notwendige Schritte werden vom System in wenigen Milli-Sekunden abgeschlossen.	
Verfügbarkeit	Sofern nicht vom System deaktiviert.	
Fragen / Kommentare		

Tabelle 7: Use-Case "Essen bestellen"

In *Tabelle 8: Use-Case "Baukastensystem*" wird die Verwendung des Baukastensystems dargestellt. Das Baukastensystem wird von den Restaurants verwendet, um eigene Webseiten zu gestalten.

NR	**UC-2021-002**	
Name	Gestalten der Webseite mit dem Baukastensystem	
Akteur	Restaurant	
Trigger	Baukastensystem	
Kurzbeschreibung	Restaurants können über ein Baukastensystem eine eigene und individuelle Webseite anlegen. So ist es ihnen möglich, einen ansprechenden Auftritt im Netz hinzulegen und mehr Aufträge zu erzielen.	
Vorbedingung	Restaurant meldet sich an.	
Essenzielle Schritte	**Intention des Restaurants**	**Reaktion des Systems**
	Restaurant besucht die Webseite	Webseite zeigt Übersicht
	Restaurant meldet sich an	Webseite meldet das Restaurant an und erstellt eine Session
	Restaurant öffnet das Baukastensystem	Webseite liefert eine Übersicht des Baukastensystems mit verschiedenen Einstellungen
	Restaurant bearbeitet die Webseite	Webseite zeigt das Ergebnis mit den Änderungen
	Restaurant speichert das gewünschte Ergebnis	Webseite speichert das Resultat und verwendet dieses anschließend für das Restaurant
Ausnahmefälle	Webseite nicht verfügbar	
Zeitverhalten	Notwendige Schritte werden vom System in wenigen Milli-Sekunden abgeschlossen.	
Verfügbarkeit	Sofern nicht vom System deaktiviert.	
Fragen / Kommentare		

Tabelle 8: Use-Case "Baukastensystem"

In *Tabelle 9: Use-Case "Anmelden*" wird die Anmeldung eines Kunden genau beschrieben. Ein Kunde muss sich auf der Webseite anmelden, um Bestellungen aufgeben und personenbezogene Daten für weitere Bestellungen speichern zu können.

NR	**UC-2021-003**	
Name	Kunde meldet sich an	
Akteur	Kunde	
Trigger	Anmelden	
Kurzbeschreibung	Der Kunde kann sich auf der Webseite anmelden, um verschiedene Daten, wie Zahlungsinformationen, Lieferadresse und Bestellverlauf, zu speichern.	
Vorbedingung	Kunde ist registriert	
Essenzielle Schritte	**Intention des Benutzers**	**Reaktion des Systems**
	Kunde besucht die Webseite	Webseite zeigt Übersicht
	Kunde wählt Login	Webseite liefert eine Maske zum Anmelden
	Kunde gibt Benutzerdaten ein	Webseite validiert und meldet den Kunden an
Ausnahmefälle	Webseite nicht verfügbar, Kunde nicht registriert	
Zeitverhalten	Notwendige Schritte werden vom System in wenigen Milli-Sekunden abgeschlossen.	
Verfügbarkeit	Sofern nicht vom System deaktiviert.	
Fragen / Kommentare		

Tabelle 9: Use-Case "Anmelden"

In folgendem Abschnitt sind die Use-Case-Diagramme zum Restaurant-Besitzer sowie zum Kunden dargestellt. Zuerst ist das Use-Case-Diagramm zum Restaurant-Besitzer zu sehen. Darunter sieht man das Use-Case-Diagramm zum Kunden.

Der Akteur in der (*Abbildung 7)* ist der Restaurant Besitzer. Im ersten Schritt muss sich der Kunde einloggen um auf das Baukastensystem zugreifen zu können. Danach kann der Besitzer die Webseite bearbeiten und zum Schluss muss der Besitzer das gewünschte Design abspeichern

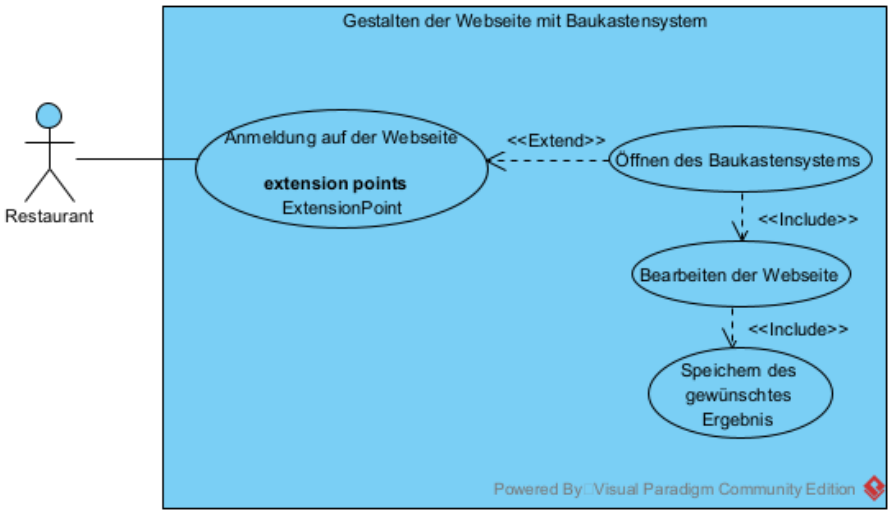

Abbildung 7: Use-Case-Diagramm Restaurant

In der *Abbildung 8* ist der Akteur unser Kunde und der Kunde möchte sich Essen bestellen. Bis er zur Essensbestellung kommt muss der Kunde einzelne Schritte durchlaufen. Als erstes muss sich der Kunde Anmelden und sich das gewünschte Restaurant auswählen. Im nächsten Schritt kann sich er die Gerichte auswählen und in den Warenkorb legen, danach schickt der Kunde die Bestellung ab. Zum Schluss muss der Kunde noch die Lieferadresse und seine Zahlungsmethode angeben.

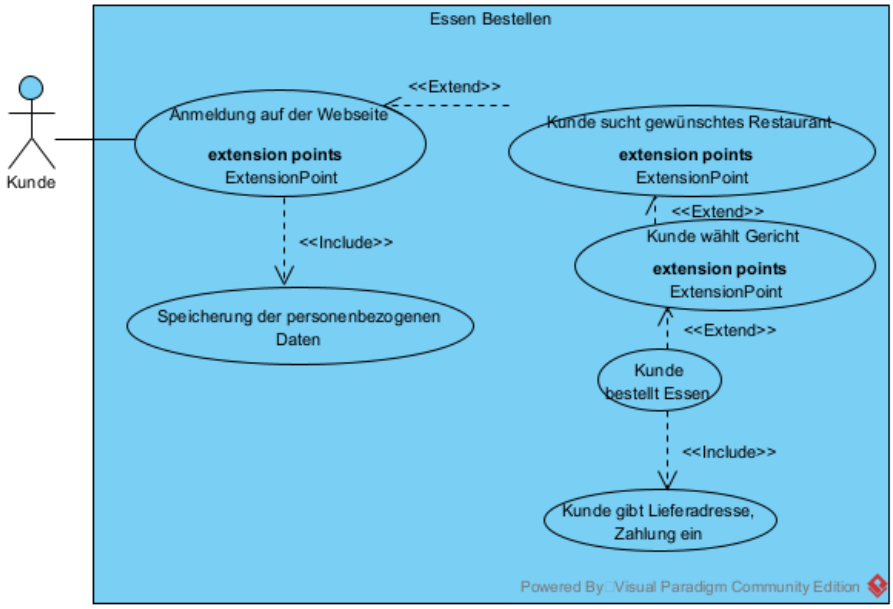

Abbildung 8: Use-Case-Diagramm Kunde

5.2 Domain-Class-Modelling

5.3 User-Interface-Design

Die in folgendem Abschnitt dargestellten Mockups wurden mit wem Web-Tool Figma erstellt. Figma ist ein webbasierter Vektorgrafik-Editor der sich perfekt für die Erstellung von Mockups für Anwendung eignet. Das folgende Mockup stellt die Startseite unseres Projekts dar. Auf jene Seite gelangt man, sobald man unsere Webadresse aufruft. Dies ist der Startpunkt für jeden Benutzer. Von hier aus soll man infolgedessen sich einloggen können, nach Resteraunt suchen können und allgemeine Informationen bekommen. Links ist die Desktopdarstellung zu sehen. Rechts die für Smartphones. Die folgenden zwei Mockups (*Abbildung 9, Abbildung 10*) sind im hellen Farbmodus dargestellt. Zudem gibt es bei machen Mockups einen dunklen Farbmodus. Für den hellen Farbmodus wird das Synonym „Light Mode“ verwendet. Für den dunklen Farbmodus wird das Synonym „Dark Mode“ verwendet.

Abbildung 10: Mockup - Starseite für Desktops

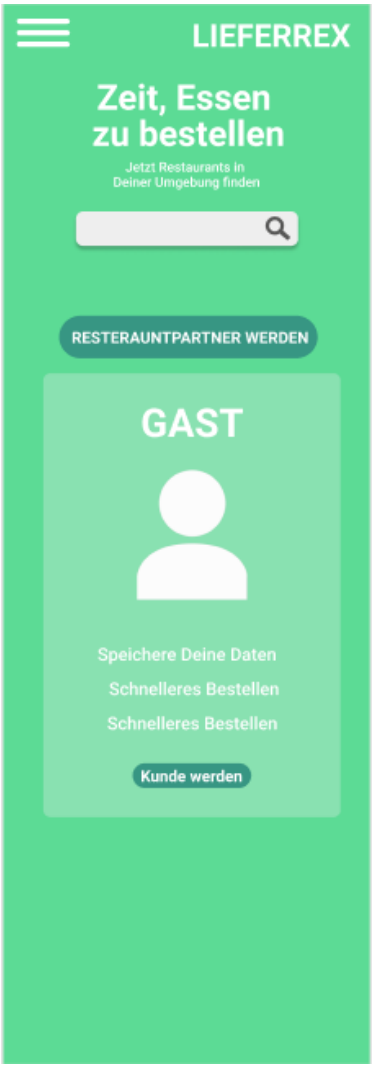

Abbildung 9: Mockup - Starseite für Mobilgeräte

Es soll ebenfalls die Möglichkeit bestehen einen Dark-Mode zu aktivieren. Deshalb ist in folgendem Mockup (*Abbildung 11*) der geplante Dark-Mode in der Desktopdarstellung zu sehen.

Abbildung 11: Mockup - Startseite für Desktops – Dark Mode

Das folgenden Wireframes (*Abbildung 12, Abbildung 13*) zeigen die Seite, die man erhält, wenn man von der Startseite aus nach einem Restaurant in der Nähe sucht. Links ist der normale Farbmodus zu sehen. Rechts ist der Dark-Mode zu sehen. Außerdem ist dies die Desktopdarstellung.

Abbildung 12: Mockup - Ergebnisse für Desktops – Light Mode

Abbildung 13: Mockup - Ergebnisse für Desktops – Dark Mode

Da das Restaurant natürlich ein Tool zur Verwaltung braucht, haben wir die folgenden zwei Mockups *Abbildung 14, Abbildung 15* entworfen. Die folgende Darstellung ist nur dem Restaurantbesitzer vorbehalten. Links ist der normale Farbmodus zu sehen. Rechts ist der Dark-Mode zu sehen.

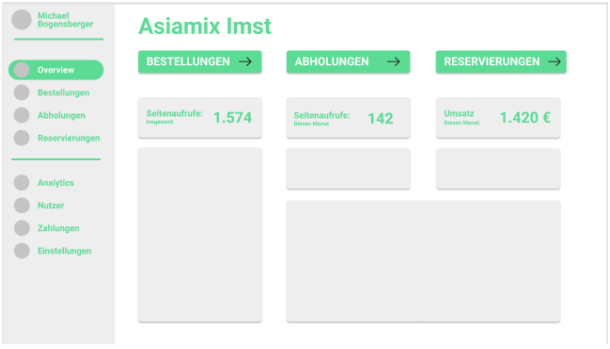

Abbildung 15: Mockup - Dashboard für Desktops – Light Mode

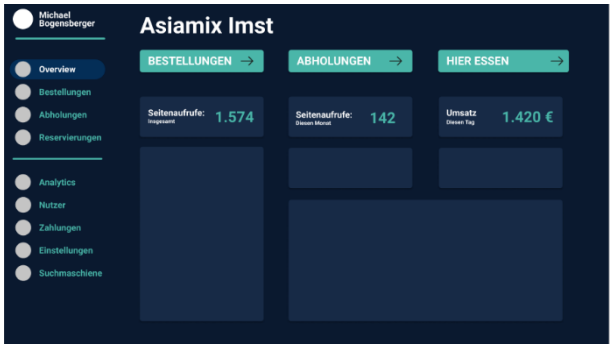

Abbildung 14: Mockup - Dashboard für Desktops – Dark Mode

Der Koch im Restaurant erhält folgende Darstellung. Hierfür gibt es das Mockup zurzeit nur im Dark-Mode. Links ist die Desktopdarstellung (*Abbildung 16*) zu sehen. Rechts (*Abbildung 17*) die für Smartphones.

Abbildung 16: Mockup - Dashboard für Desktops - Mitarbeiter - Dark Mode

Abbildung 17: Mockup - Dashboard für Mobilgeräte - Mitarbeiter - Dark Mode

Um sich registrieren zu können, gibt es ebenfalls ein Mockup (*Abbildung 18*). Die folgend dargestellte Registrierung ist den Kunden vorbehalten. Die Darstellung ist ebenfalls für Desktops und im hellen Farbmodus.

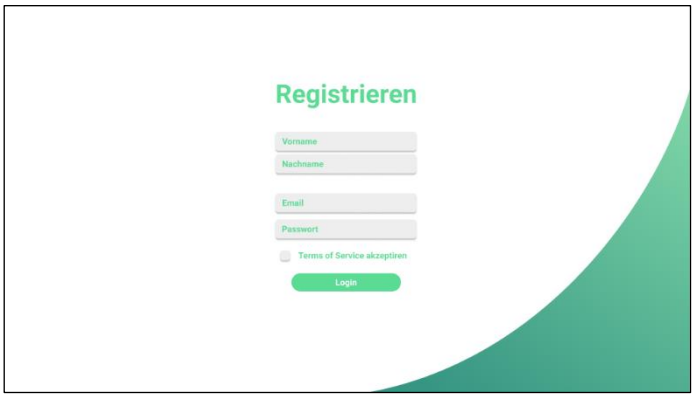

Abbildung 18: Mockup - Registrierung für Desktops - Light Mode

6 Systementwurf

Das entwickelte System besteht hauptsächlich aus dem Frontend und Backend. Um dies mit allen verschiedenen Komponenten möglichst übersichtlich zu strukturieren, wurde das MVC-Pattern angewendet. Dieses sogenannte Model-View-Controller-Pattern sorgt für eine Trennung von Darstellung und Verarbeitung von Daten. Objekte werden in Form von Models gespeichert. Diese Klassen werden mit allen Eigenschaften und Methoden definiert. Für die Ausgabe werden Views verwendet. Views, im Projekt HTML-Dateien, sorgen für eine saubere Präsentation der bereitgestellten Daten. Diese sind im Projekt unter den Ressourcen abgelegt. Ein Controller kümmert sich schließlich um Eingaben vom Benutzer (Aufruf der Webseite, Zugriff über REST-Schnittstelle) und liefert eine View mit entsprechenden Daten wieder.

Desweitern gibt es im Projekt Repositories und Services. Repositories ermöglichen die Kommunikation mit der Datenbank und stellen verschiedenste CRUD-Operationen bereit, um mit Daten aus dieser zu arbeiten. Die Services bauen auf die Repositories auf und erweitern diese um weitere Methoden.

Mandanten, oder auch Restaurant, können sich auf der Webseite über einen Baukasten ihre eigenen Webseiten zusammenbauen. Diese Webseiten bestehen aus verschiedensten Layouts und Modulen, die man nach Belieben wählen und bearbeiten kann. Ein User kann über die Webseite Restaurants durchsuchen und über die individuellen Seiten bestellen. Jedes Restaurant verfügt auch über ein Dashboard, das viele Inforationen und Einstellungen beinhaltet. Uber das Dashboard werden etwaige Informationen über den Mandaten eingegeben, so unter anderem die Speisekarte oder Öffnungszeiten.

Im Folgenden wird das Projekt in Frontend und Backend eingeteilt.

6.1 Architektur

6.1.1 Design der Komponenten

6.1.2 Benutzerschnittstellen

6.1.3 Datenhaltungskonzept

Im Projekt wird MySQL benutzt um die Daten, die vom dem User kommen, in die Datenbank zu speichern. Durch das Speichern dieser Daten auf einen auf Server, können jederzeit die Daten abgerufen werden. Ein Anwendungsfall ist z.B. die Speicherung von Gerichten sowie die Erstellung der Angestellten-Accounts. Die MYSQL Konfiguration kann unter resources und application-dev.proerties entnommen werden. In dieser Konfiguration ist die URL des MySQL Server sowie Username und Passwort enthalten, auch ist der Treiber für die MYSQL Verbindung enthalten. Für die Darstellung einer Tabelle werden oftmals ein ER-Diagramm vorgefertigt. Für das Projekt ist das ER-Diagramm in der Abbildung 19: Vollständiges ER-Diagramm verwendet worden.

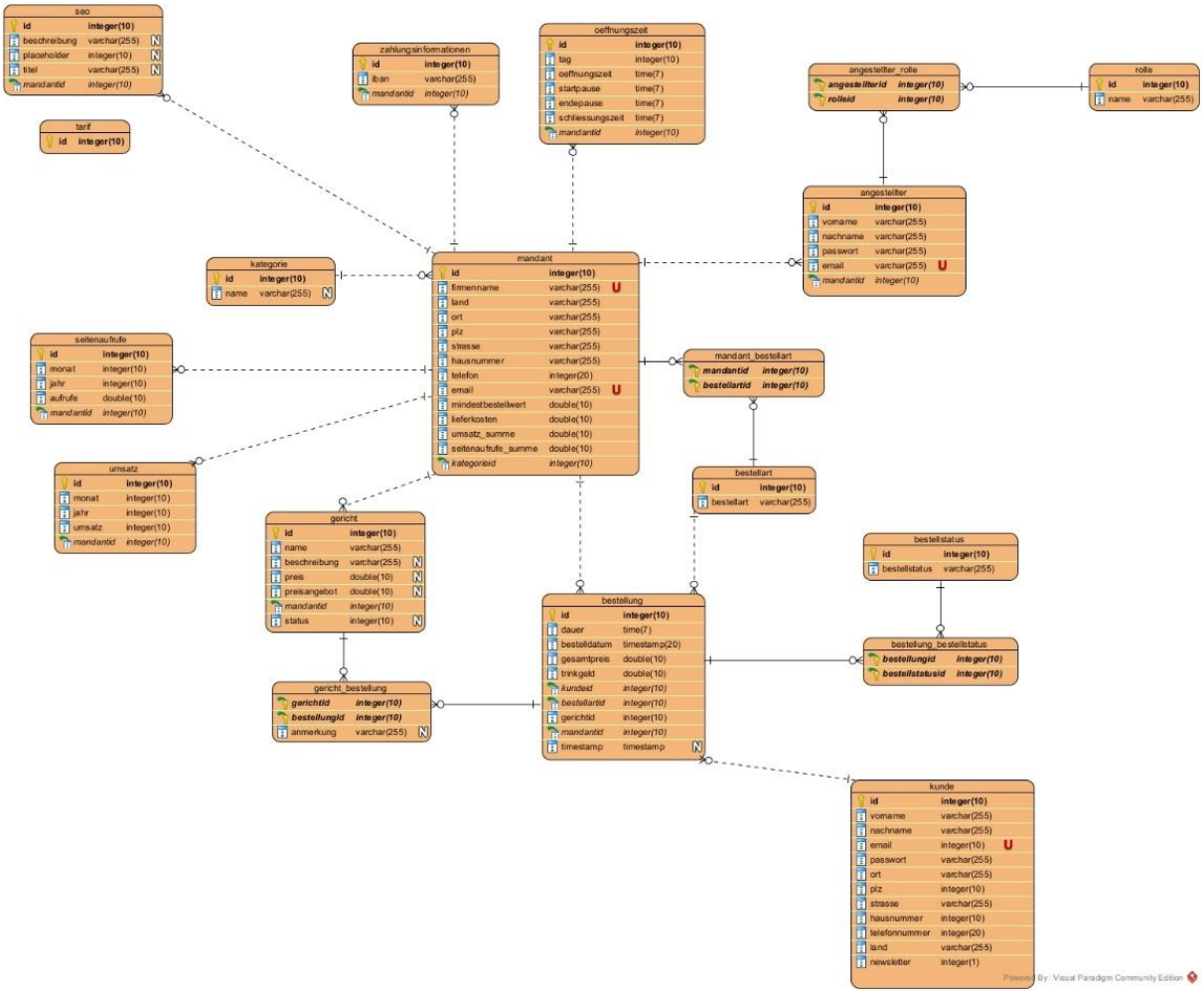

Abbildung 19: Vollständiges ER-Diagramm

6.1.4 Konzept für Ausnahmebehandlung

6.1.5 Sicherheitskonzept

Da wir verschiedene Benutzer haben, die verschiedene Berechtigungen besitzen, hat jeder Benutzer auch eine Rolle. Es gibt die Rollen User, Mandant und Angestellter. Als User soll man in der Lage sein Essen zu bestellen. Als Mandant muss man alle eingehenden Bestellungen und Zahlungen sehen, und seine Restaurant-Seite bearbeiten können. Als Angestellter soll man wissen, welche Gerichte man kochen muss und sie, als erledigt markieren, sobald die Bestellung zubereitet ist.

In Spring Security kann man nun konfigurieren, welche Ressourcen welche Rollen aufrufen dürfen, oder welche sie nicht sehen dürfen. Dafür ist die diese Methode aus der SecurityConfiguration Klasse zuständig.

```java
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.
      authorizeRequests()
      .antMatchers("/index.html").permitAll()
      .antMatchers("/profile/**").authenticated()
      .antMatchers("/dashboard/**").hasRole("MANDANT")
      .and()
      .formLogin()
      .loginPage("/login").permitAll()
      .and().csrf().disable();

}
```

Abbildung 20: Spring Boot Security Konfiguration

Diese Konfiguration besagt, dass jeder Zugriff auf die index.html und die login.html hat. Seiten, die /profile/ in der URL beinhalten, dürfen nur von Authentifizierte User aufgerufen werden und /dashboard/ dürfen nur Benutzer sehen, die die Rolle Mandant haben.

Spring Security wird verwendet, da es die einfachste Methode ist Benzer in Spring Boot zu Authentifizieren.

6.1.6 Design der Testumgebung

6.1.7 Design der Ausführumgebung

6.2 Detailentwurf

6.3 Frontend

Unser Frontend ist prinzipiell in drei Teile aufgeteilt. Die Hauptseite für die Kunden, das Baukastensystem und das Dashboard für das Resteraunt. Dabei teilen sich erstere zwei die gleichen Technologien, während für das Dashboard teilweise unterschiedliche Technologien verwendet werden. Die zugehörigen Mockups für das Frontend sind im Abschnitt *User-Interface-Design* zu finden. In folgendem Abschnitt sind die verwendeten Technologien sowie deren Einsatzbereiche zu sehen.

Die verwendeten Technologien im Frontend werden in folgender Tabelle *(Tabelle 10*: Verwendete Frontend-Technologien*)* dargestellt. Dabei ist der Bereich und die für den jeweiligen Bereich benützen Technologien zu sehen.

Bereich	Materialize	Materialize Stepper	Halfmoon	JQuery
Hauptseite	✓	✓		✓
Baukastensystem	✓			✓
Dashboard			✓	✓

Tabelle 10: Verwendete Frontend-Technologien

6.3.1 Einbindung der Frontend-Technologien

Alle im Frontend verwendeten Technologien sind über CDN (Content Delivery Network) eingebunden. Dies ermöglicht einen leichten Versionswechsel jener Technologien. Zudem hält es die Struktur im Projekt schlanker.

6.3.2 Struktureller Aufbau der Dateien

Grundsätzlich befindet sich das Frontend in einer Spring Boot Applikation immer unter der „resources“ Ordner. Dort finden sich nun wieder zwei Ordner. Einmal „static“ und einmal ein „templates“ Ordner. Im „static“ Ordner sind die CSS- und JavaScript-Dateien zu finden. Im „templates“ Ordner sind die HTML Dateien zu finden. In den zwei zuvor genannten Ordnern gibt es jeweils eine Unterteilung zwischen Hauptseite, Dashboard und Baukastensystem.

6.3.3 Verwendete Versionen

In folgender Tabelle *(Tabelle 11*: Frontend-Technologien Versionen*)* werden die jeweiligen Versionen der Frontend-Technologien dargestellt. Dabei sind die jeweiligen Bereiche und die jeweiligen Versionen der Technologien zu sehen.

Bereich	Materialize	Materialize Stepper	Halfmoon	JQuery
Hauptseite	V 1.0.0	V 3.1.0		V 3.5.1
Baukastensystem	V 1.0.0			V 3.5.1
Dashboard			V 1.1.1	V 3.5.1

Tabelle 11: Frontend-Technologien Versionen

6.4 Backend

Das Backend ist die Datenzugriffsebene für die http-Anfragen des Frontend. Es stellt somit das Rückgrat unserer Applikation dar. Über die Benutzeroberfläche des Frontends soll es somit möglich sein Daten zu bearbeiten, erstellen und löschen zu können. Diese Daten sind in einer Datenbank gespeichert. Bei einem Aufruf der Webseite wird eine Anfrage an das Webinterface[2] des Webserver gesendet. Eine Spring-Boot Business-Anwendung nimmt diese Anfrage an und verarbeitet sie weiter. Je nach Art des Request werden unterschiedliche Aktionen in der Applikation ausgeführt. Die Software beinhaltet die Logik, um die Daten der Datenbank zu manipulieren. Somit kann eine Spezifische Antwort dem Client zurückgesendet werden. Ein persistenter Datenfluss zwischen dem Frontend und der Datenbank ist dadurch gewährleistet.

Wie schon zuvor erklärt wird das MVC-Pattern verwendet. In Spring Boot werden dabei die Model und Controller Komponenten implementiert. Zusätzlich wird aber noch ein Service-Layer eingebaut, der sich zwischen Model und Controller befindet und die Businesslogik beinhaltet. In Spring Boot spricht man daher von folgendem Layer.

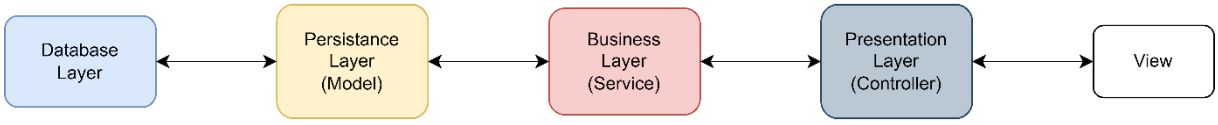

Abbildung 21: MVC-Pattern Spirng Boot

Der Database Layer beinhaltet die konkrete Datenbank.

[2] http Schnittstelle

Im Persistance Layer befinden sich alle Model-Klassen, die die Entitäten der Datenbank widerspiegeln. Diese Klassen stellen ein POJO[3] dar und könen wie gewohnt verwendet werden. Sie beingalten auch Validierungs Annotationen, damit nur Valide Daten gespeichert werden. Diese Klassen werden für CRUD-Operationen verwendet.

Der Business Layer implementiert die Service Klassen welche für die Validierung, Business Logik und die Autorisierung zuständig ist.

Die Controller-Klassen befinden sich im Presentation Layer. Dieser Stellt die Schnittstelle zu den Views dar und konvertiert die JSON-Requests zu Java verständlichen Code.

Die Views stellt die Client Seite dar.

Folgende Grafik Abbildung 22: Konkrekte Spring Boot Architektur stellt nun die Konkrete Implementierung dar.

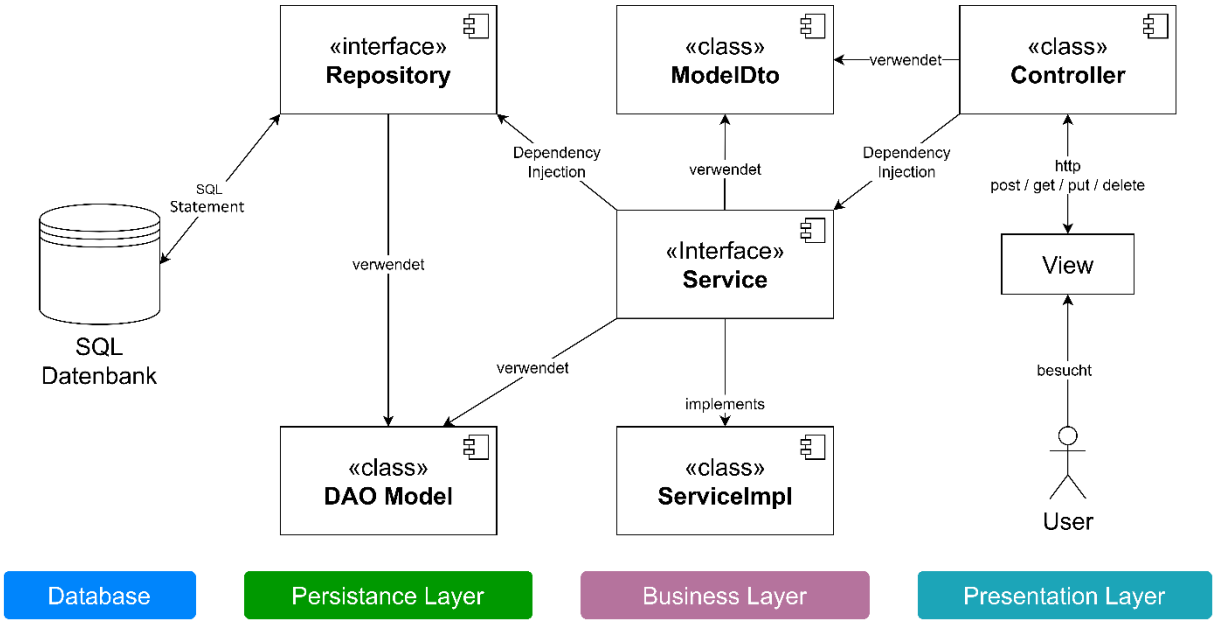

Abbildung 22: Konkrekte Spring Boot Architektur

In *Abbildung 22*: Konkrekte Spring Boot Architektur sind wieder die vier Layer von Spring Boot zu sehen. Es Zeigt, welche Komponenten für eine Anfrage. Um das Modell zu vereinfachen, wird nur jede Komponente einmal gezeigt. In der Realität gibt es eine Vielzahl an Klassen, die die Logik widerspiegeln.

Herzstück unserer Anwendung ist das Service Interface. Dieses Interface beinhaltet alle Methoden, die notwendig sind. Damit der Controller Zugriff auf diese Klasse hat wird mittels der @Autowired Annotation von Spring Boot eine Dependency Injection durchgrführt. Somit entsteht eine geringe Koppelung zwischen den Klassen. Zwischen dem Controller und dem Service werden i.d.R. DTO's[4] Objekte übergeben. Dies vereinfacht die Programmierung and gewissen stellen und bietet der Schnittstelle zusätzliche Sicherheit.

Das Repository Interface erweitert sich zum Spring Boot proprietären JPA-Repository. Dieses JPA-Repository beinhaltet die wichtigsten CRUD-Operationen, die man für ein DAO-Model verwenden kann. Im Repository werden Spezifische Methoden angegeben, die nicht Standardmäßig im JPA-Repository vorhanden sind. Spring Boot leistet dann im Hintergrund die ganze Arbeit löst die Methode

[3] Plain Old Java Object
[4] Data Transfer Object

in ein valides SQL-Statement auf. Mittels Depandency Injection wird im Service layer die passende Funktion aus dem Repository aufgerufen, und auf die Datenbank zugegriffen werden. Dabei werden die DAO-Model-Klassen verwendet.

Spring Boot wird verwendet, da es eine sehr umfangreiche und komplexe Backend-Applikation realisieren kann. Desweiterem haben alle Projektanten gute Kenntnisse in der Java Entwicklung. Während dem Programmierunterricht haben wurden ebenfalls schon einfache Projekte umgesetzt, was die Auswahl des Backendsystems nur noch einfacher gemacht hat.

6.4.1 REST

REST ist eine Schnittstelle womit der Datentausch im Internet zwischen zwei Geräte ermöglicht wird. REST verwendet Hauptsächlich die Protokolle HTTP und HTTPS. Rest ist auch so gut mit jeder Firewall kompatibel. Mit REST kommen auch die DNS-Versionierungen sowie die URL-Versionierungen dazu ein Beispiel ist die DNS http://v1.api.foo.com/customer/1234. Mit REST können viele Methoden wie GET, POST, PUT, DELETE, HEAD genutzt werden. In der folgenden Tabelle sind die Methoden dargestellt.

Methode	Beschreibung
GET	Bei einer GET Anfrage können die Daten vom Server mittels GET Methode angefordert werden.
PUT	Mit PUT können die Ressourcen, die schon vorhanden sind abgeändert werden, falls keine Ressource vorhanden ist wird eine neue Ressource angelegt.
DELTE	Mittels DELETE kann die angegebene Ressource gelöscht werden.
HEAD	Mithilfe HEAD können die Metadaten von der Ressource angefordert werden.

Tabelle 12: REST Methoden

Der Vorteil bei REST ist, dass REST zustandlos ist, das heißt, dass jede REST-Nachricht alle Information beinhaltet die für den Server sowie den Client notwendig sind. In diesem Verfahren werden keine Zustandsinformationen zwischen den zwei Nachrichten gespeichert. Dies wird auch als zustandloses Protokoll bezeichnet. Durch dieses zustand können Webservices besser skaliert werden.

6.4.2 Baukastensystem

Das Baukastensystem stellt eine zentrale Komponente des Projekts dar. Über den Baukasten werden von den Mandaten ihre individuellen Webseiten angefertigt. Um dies zu speichern, wurde das in Abbildung 23: ER-Modell Baukasten Modell entwickelt. Jeder Mandant kann ein Layout wählen, das verschiedene Positionen beinhaltet. Jede einer solcher Position enthält ein Fragment (oder auch Modul). Diese Fragmente werden vom Mandanten erstellt und diesem zugeordnet. Jedes Fragment kann ein anderer Fragmenttyp sein (Text, Bild, usw.).

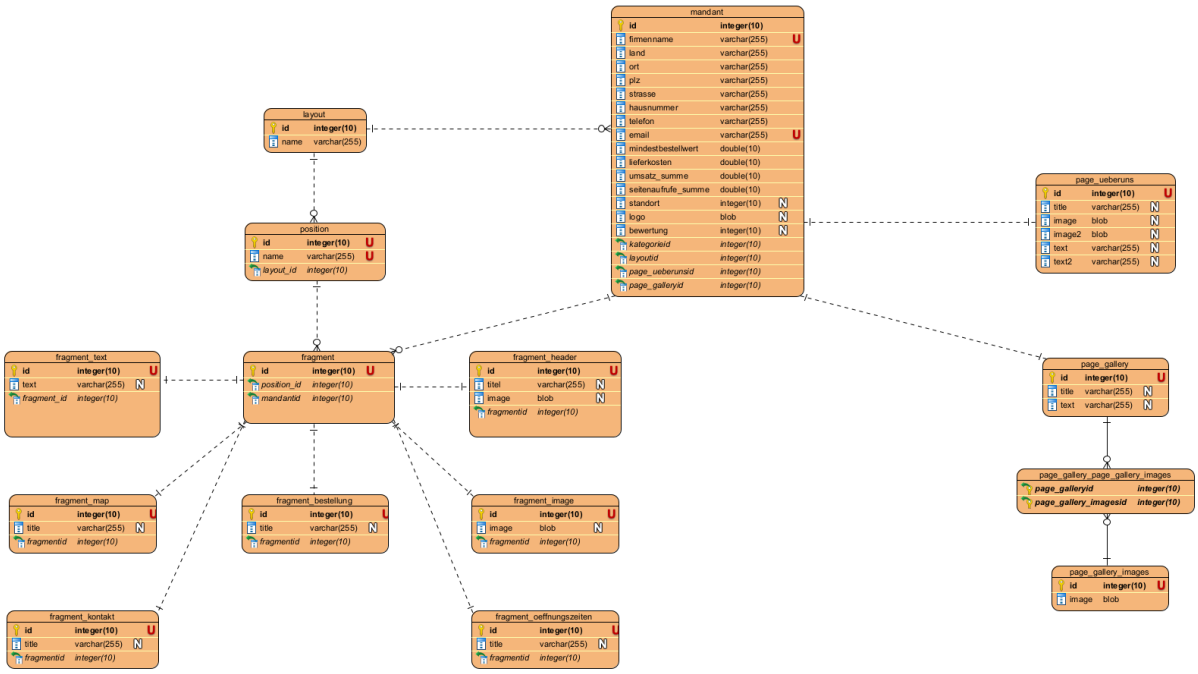

Abbildung 23: ER-Modell Baukasten

7 Implementierung

7.1 Frontend

7.1.1 Struktur

7.1.2 Darstellung

7.1.3 JQuery REST request

Im Projekt wird REST hauptsächlich für die Datenübertragung an JQuery verwendet. JQuery stellt dann die einzelnen Daten im Modal dar. REST vereinfacht die Darstellung im Modal dadurch muss keine weitere Webpage erstellt werden. Falls die Methode mit einer neuen Webpage gewählt wird, ist REST dann überflüssig, da die Daten direkt vom Objekt an Thymeleaf übergeben werden kann. Die einzige Lösung asynchron Daten in ein Modal zu laden, ist es JQuery sowie REST zu verwenden. Mithilfe der GET Methode von JQuery können einfach die Daten von REST ausgelesen und dargestellt werden. In der folgenden Abbildung 24: Codebeispiel für die Darstellung im Modal ist ein Codebeispiel zu diesem Problem.

```
$.get("http://localhost:8080/api/gericht/" +id, function(data, status){
    console.log(data)
    $('#gericht-info-name').val(data.name);
    $('#gericht-info-beschreibung').val(data.beschreibung);
    $('#gericht-info-preis').val(data.preis);
    $('#gericht-info-aktionspreis').val(data.preisangebot);
}
```

Abbildung 24: Codebeispiel für die Darstellung im Modal

7.2 Backend

7.2.1 Model Klassen

7.2.2

7.3 Baukastensystem

7.3.1 Ausgabe einer Restaurantseite

Wurde eine Webseite erstellt, kann nun über den Restaurantnamen auf diese zugegriffen werden. Um den Zugriff kümmert sich hier ein Kontroller, der die Konfiguration ausliest und eine View mit den Daten zurückgibt. Der Kontroller in Abbildung 25: Baukasten Kontroller liest hier Informationen über Layout, Fragmente und Positionen aus.

```java
@GetMapping("/baukasten/{restaurant}")
public String showBaukasten(Model model, @PathVariable String restaurant){

    // Get Mandant ueber Name in der URL
    Mandant mandant = mandantServiceImpl.findMandantByFirmenname(restaurant).get();

    // Alle Fragmente des Mandaten ueber dessen ID
    List<Fragment> fragments = fragmentServiceImpl.findFragmentByMandant_id(mandant.getId());

    // Layout des Mandanten der VIEW uebergeben
    model.addAttribute("layout", mandant.getLayout().getName());

    // Alle Fragmente mit Position der VIEW uebergeben
    for (Fragment fragment : fragments) {
        model.addAttribute(fragment.getPosition().getName(), fragment);

        // Datenermittlung bei Ausgabe von Karte, Kontakt, etc.
        if (fragment.getFragmenttype().getType().equals("karte")) {
            model.addAttribute("gerichte", mandant.getGerichte());
        } else if(fragment.getFragmenttype().getType().equals("kontakt")){
            // TODO: Kontakte des Mandanten ausgeben
            model.addAttribute("kontakt", "getKontakt");
        }
    }

    return "baukasten/frame.html";
}
```

Abbildung 25: Baukasten Kontroller

Für die richtige Darstellung dieser übergebenen Daten wird Thymeleaf verwendet. In einen Frame, einer einfache Webseite, wird über das „th:replace“ Attribute das entsprechende Layout geladen. In dieses werden über das gleiche Attribut leere Fragmente in die richtigen Positionen eingefügt. Diese Fragmente können im letzten Schritt mit den Daten des Mandaten befüllt werden. In Abbildung 26: Thymeleaf Ausgabe Restaurant wird der Aufbau eines Layouts dargestellt.

```html
<div th:fragment="layoutEINS">
  <div class="row ">
      <div class="col l12 m12 s12 main-right">
        <div th:replace="'baukasten/fragments/modules/' + ${r1c1.fragmenttype.type}(content=${r1c1})">
        </div>
      </div>
  </div>
  <div class="row">
      <div class="col s12 m7 l9 main-right">
        <div th:replace="'baukasten/fragments/modules/' + ${r2c1.fragmenttype.type}(content=${r2c1})">
        </div>
      </div>
      <div class="col s12 m5 l3 main-right">
        <div th:replace="'baukasten/fragments/modules/' + ${r2c2.fragmenttype.type}(content=${r2c2})">
        </div>
      </div>
    </div>
</div
```

Abbildung 26: Thymeleaf Ausgabe Restaurant

8 Deployment

9 Tests

Tests werden in der Systementwicklung häufig genutzt um Fehler, oder sogenanngte Bugs, zu finden und diese zu beseitigen. Dies wird gemacht, um die Funktion aller Komponenten zu garantieren und am Ende ein vollständiges Produkt abliefern zu können.

9.1 Systemtests

Im Bereich Systemtest werden die einzelnen Komponenten einer Anwendung auf ihr zusammenwirken getestet. Die Systemtests konzentrieren sich auf die Funktionalität der Anwendung. Mithilfe von Black-Box-Tests, eine Methode für Softwaretests bei denen der Tester nicht die genaue Implementierung von den verwendeten Methoden kennen muss, werden die einzelnen Kompenenten anhand der Spezifikationen bzw. Anforderungen getestet und weiterentwickelt.

Im Rahmen dieses Projektes, fallen viele Tests an. Im Folgendem werden einige Testfälle aufegelistet. Ein Tesfall ist der Mandanten Kontroller, dort wird getestet ob die Stammdaten des Mandanten veränderbar sind.

Stammdaten des Mandanten ändern			
Nr:	1	Erfolgreich getestet:	✓
Beschreibung:	Es soll getestet werden ob eine Änderung der Stammdaten des Mandanten auf dem Dashboard korrekt gespeichert und wieder angezeigt werden.		
Betroffener Programmteil:	MandantController		
Vorbedingung:	Damit dieser Testfall getestet werden kann muss man sich zuvor einloggen. Andernfalls gelangt man nicht aufs Dashboard.		
Tester:	Michael Bogensberger	Datum:	17.05.2022
		Test Schritte:	
Schritt	Aktion		Erwartetes Ergebnis
1	Einloggen mit Email und Passwort auf der Kundenseite.		Session wird erstellt und man wird aufs Dashboard weitergeleitet.
2	In der Navbar auf die "Mandant" Seite klicken.		Man gelangt auf die Mandanten-Seite.
3	Addresse, Lieferkosten sowie Mindestbestellwert ändern und auf speichern klicken.		Die Seite wird neu geladen und eine Meldung mit "Daten gespeichert" erscheint.
4	Checken ob die eingegeben Daten auf der Seite angezeigt werden.		Die zuvor eingegeben Daten werden korrekt angezeigt.

Tabelle 13 Stammdaten des Mandanten ändern

Ein weiterer Testfall ist der Bestell Kontroller, dort wird getestet ob ein Kunde eine Bestellung tätigen kann.

Bestellung tätigen			
Nr:	1	Erfolgreich getestet:	✓
Beschreibung:	Es soll getestet werden ob eine Bestellung über die Webseite erfolgreich angelegt werden kann. Dazu wird übers Frontend eine Bestellung mit verschiedenen Gerichten angelegt und diese Abgeschichkt. Danach wird zu jener Seite gewechselt, bei der man die Bestellungen einsehen kann. Nun wird getestet ob die Bestellung hier auch angezeigt wird		
Betroffener Programmteil:	BestellController		
Vorbedingung:	Damit dieser Testfall getestet werden kann muss man sich zuvor einloggen. Ansonsten kann man keine Bestellung tätigen.		
Tester:	Michael Bogensberger	Datum:	17.05.2022
	Test Schritte:		
Schritt	Aktion		Erwartetes Ergebnis
1	Einloggen mit Email und Passwort auf der Kundenseite.		Session wird erstellt und man wird auf die Startseite weitergeleitet.
2	Auf der Hauptseite eine Addresse eingeben und nach einem Restaurant suchen.		Die "Suchen" Seite mit Restaurants wird angezeigt.
3	Resteraunt anklicken.		Man gelangt auf die Seite des Restaurants.
4	Im Menü-Formular Gerichte auswählen und auf hinzufügen klicken.		Gerichte werden dem Warenkorb hinzugefügt.
5	Auf Bestellen klicken.		Man gelangt auf die "Bestellung abschließen" Seite.
6	Mit PayPal bezahlen. Dazu drückt man auf bezahlen.		Nun öffnet sich ein Fenster von der PayPal API.
7	Mit PayPal Daten anmelden und auf bezahlen klicken.		Das Fenster schließt sich und man gelangt zurück auf die Startseite.
8	Auf die Bestellung Seite wechseln. Dazu Klickt man in der Navbar auf Profil und dann auf Bestellungen.		Man gelangt zur "Bestellungen" Seite.
9	Checken ob die Bestellung sichtbar ist.		Die zuvor getätigte Bestellung wird nun angezeigt.

Tabelle 14 Bestellung tätigen

Im nächsten Fall wird getestet, ob der Mandant seine Öffnungszeiten anpassen kann.

Öffnungszeiten des Restaurants ändern			
Nr:	1	Erfolgreich getestet:	✓
Beschreibung:	Es soll getestet werden ob eine Änderung der Öffnungszeiten des Mandanten auf dem Dashboard korrekt gespeichert und wieder angezeigt werden.		
Betroffener Programmteil:	OeffnungszeitenController		
Vorbedingung:	Damit dieser Testfall getestet werden kann muss man sich zuvor einloggen. Andernfalls gelangt man nicht aufs Dashboard.		
Tester:	Michael Bogensberger	Datum:	17.05.2022
	Test Schritte:		
Schritt	Aktion		Erwartetes Ergebnis
1	Einloggen mit Email und Passwort auf der Kundenseite.		Session wird erstellt und man wird aufs Dashboard weitergeleitet.
2	In der Navbar auf die "Öffnungszeiten" Seite klicken.		Man gelangt auf die Öffnungszeiten-Seite.
3	Öffnungszeiten an verschiedenen Tagen ändern und zu guter letzt auf speichern klicken. Dazu sollte man bei zwei Tagen die Öffnungszeiten ändern und an einem Tag das Restaurant als geschlossen makieren.		Die Seite wird neu geladen und eine Meldung mit "Daten gespeichert" erscheint.
4	Checken ob die eingegeben Daten auf der Seite angezeigt werden.		Die zuvor eingegeben Daten werden korrekt angezeigt.

Tabelle 15 Öffnungszeiten des Restaurants ändern

Auch getestet wird die Erstellung einer Webseite mit dem Baukastensystem. Hier muss überprüft werden, ob die erstellte Seite korrekt gepeichert und anschließend angezeigt wird.

Webseite im Baukasten anlegen			
Nr:	4	Erfolgreich getestet:	X
Beschreibung:	Es soll getestet werden ob, ob Mandanten mithilfe des Baukastenszstems eine Webseite erstellen koennen. Diese muss korrekt in der Datenbank gespeichert werden damit sie anschließend aus der Sicht des Kunden ausgegeben werden kann.		
Betroffener Programmteil:	BaukastenController, RestaurantController		
Vorbedingung:	Damit dieser Testfall getestet werden kann, muss sich der Mandant vorher anmelden.		
Tester:	Niklas Heim	Datum:	23.06.2022
	Test Schritte:		
Schritt	Aktion		Erwartetes Ergebnis

1	Einloggen mit Email und Passwort als Mandant.	Session wird erstellt und man wird auf die Startseite weitergeleitet.
2	Über die Einstellungen den Baukasten öffnen.	Leere Baukastenseite öffnet sich.
3	Allgemeine Einstellungen treffen.	Allgemeine Einstellungen werden übernommen.
4	Hinzufügen der Module mit entsprechenden Werten.	Ausgabe der erstellten Module.
5	Webseite vollständig konfiguriert.	Erstellte Webseite wird in der Datenbank gepeichert.
6	Suchen des Restaurant über das Such-Menü.	Entsprechendes Restaurant wird ausgegeben.
7	Zuvor erstellte Webseite des Restaurants wird angezeigt.	Webseite des Restaurants wird ohne Fehler ausgegeben.
8	Auf die Bestellung Seite wechseln. Dazu Klickt man in der Navbar auf Profil und dann auf Bestellungen.	Man gelangt zur "Bestellungen" Seite.

Tabelle 16 Webseite im Baukasten anlegen

In Tabelle 17 wird der Registrierung und Login Use-Case für Kunden getestet. Dabei wird der gesamte Ablauf vom Erstellen eines Benutzers bis zum Anmeldevorgang beschrieben. Das erwartete Ergebnis ist ein Valider User, der in korrekt in der Datenbank gespeichert ist. Desweiterem soll eine sichere Session erzeugt werden, die den angemeldeten User verifiziert.

User Registrierung und Login			
Nr:	1	Erfolgreich getestet:	✓
Beschreibung:	Es soll überprüft werden, ob sich ein Kunde oder ein Restaurant Mitarbeiter erfolgreich ins System einloggen kann.		
Betroffener Programmteil:	SecurityController, UserPrincipalDetailsService		
Vorbedingung:	Es darf kein Benutzer auf der Webseite bereits angemeldet sein. Ein User mit derselben E-Mail darf nicht vorhanden sein.		
Tester:	Julian Meilinger	Datum:	23.05.2022
	Test Schritte:		
Schritt	Aktion	Erwartetes Ergebnis	
1	Auf der Homepage klickt man auf den "Registrieren" Button	Man wird zur Registrierungs-Seite weitergeleitet.	
2	Man gibt gültige Anmeldeinformationen ein und akzeptiert die ABG's. Danach klickt man auf dem Button "Registrieren".	Es erscheint die Meldung "Benutzer erfolgreich angemeldet".	
3	Man geht zurück zur Homepage und klickt rechts oben auf den "Einloggen" Button.	Man wird zur Login-Seite weitergeleitet.	

4	Hier gibt man die zuvor gewählte E-Mail und das dazu passende Passwort ein. Danach klickt man auf dem Button "Login".	Danach wird man zur Homepage weitergeleitet und man ist angemeldet. Desweiterem wird ein neues Cookie gespeichert, der die Session-ID zur Verifizierung beinhaltet

Tabelle 17 User Registrierung und Login

9.2 Akzeptanztests

Im gegensatz zu den Systemtests sind die Akzeptanztests, Tests die aus der Sicht des Benutzers funktionieren sollen, das heißt die Überprüfung wird vom Benutzer getestet, sogennante Beta-Tests. Die Akzeptanztest werden meist in der letzten Phase eines Projektes durchgeführt, im besten Fall bevor der Kunde die Software benutzt.

10 Evaluation

10.1 Projektevaluation

10.2 Produktevaluation

10.3 Resümee

11 Benutzerhandbuch

12 Zusammenfassung

Restaurants können durch unser entwickeltes Produkt schnell und bequem einen Auftritt im Internet erstellen. Sie bauen sich über einen Baukasten selber eine individuelle Webseite nach ihren Wünschen. Über diese erstellte Webseite können nun Kunden bei den Restaurants auf Lieferung oder zur Abholung bestellen. Dem Restaurant werden über ein Dashboard relevante Informationen und Statistiken wie beispielsweise Seitenaufrufe oder Umsatz angezeigt. Es werden auch aktuelle Bestellungen aufgelistet.

Als Backend wird Spring Boot, ein beliebtes und robustes Java Framework, verwendet. Mithilfe von Spring Boot werden Anfragen und Daten für alle Funktionen des Systems verarbeitet. Im Projekt wird das MVC-Patten angewendet. Dieses Pattern sorgt für eine übersichtliche Struktur, indem Darstellung, Verarbeitung und Speicherung von Daten getrennt werden. Alle Informationen, die das System benötigt, werden in Form von Models oder auch Klassen dargestellt. So werden beispielsweise Kunden mit Namen, Adresse und weiterem angelegt. Diese werden in der MySQL Datenbank gespeichert. Ein Kontroller ist verantwortlich für die Verarbeitung von Informationen. Je nach Art der Anfrage, die der Kontroller erhält, liest, verarbeitet, löscht oder erstellt dieser Daten. Auf eine Anfrage an den Webserver antwortet ein Kontroller mit einer View. Dies ist eine Vorlage einer Webseite, die mit entsprechenden Daten aus der Datenbank befüllt wird, um dynamisch Informationen darstellen zu können. Um die erhaltenen Daten auf der Seite anzuzeigen, wird Thymeleaf verwendet. Die Vorlagen für die Webseiten werden mit Platzhalten ausgestattet, die später Thymeleaf mit den Daten des Kontrollers befüllt. Thymeleaf kann des Weiteren durch verschiedenste Funktionen auf verschiedenste Daten reagieren und die Vorlage dementsprechen anpassen. Auch gibt es eigene Kontroller, die als REST-Schnittstelle in Form einer API verwendet werden können. Mithilfe dieser API können zum Beispiel Daten auf der Webseite geladen werden, ohne diese neu laden zu müssen. Das fertige System

verwendet auch externe API, wie Google Analytics für Statistiken, Google Maps für die Darstellung von Karten und Paypal für das Abwickeln von Zahlungen.

A. Anhang

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltexte

Baeldung. (März 2022). Von https://www.baeldung.com abgerufen

Computer Weekly. (Mai 2022). Von https://www.computerweekly.com/de/definition/Systemtest abgerufen

Google Maps Platform. (März 2022). Von https://developers.google.com abgerufen

Halfmoon dokumentation. (April 2022). Von https://www.gethalfmoon.com/docs abgerufen

JQuery documentation. (März 2022). Von https://api.jquery.com/ abgerufen

Materialize documentation. (April 2022). Von https://materializecss.github.io/materialize/ abgerufen

Materialize Stepper documentation. (März 2022). Von shorturl.at/pPT89 abgerufen

PayPal API documentation. (März 2022). Von https://developer.paypal.com/api/rest/ abgerufen

Thymeleaf documentation. (März 2022). Von https://www.thymeleaf.org/ abgerufen

Wikipedia. (Mai 2022). Von https://de.wikipedia.org/wiki/Akzeptanztest_(Softwaretechnik) abgerufen