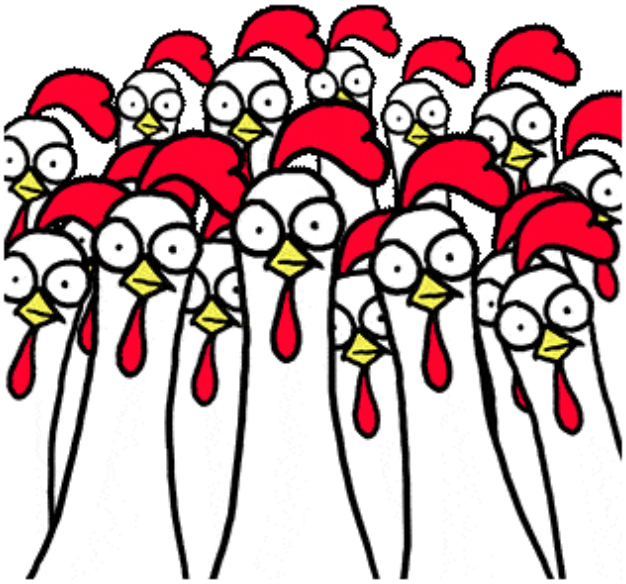
Durchführungszeitraum
08.09.2025 - 01.11.2025

Lehrer
Herr Schmiech
Herr Maier

Name
Pauline Köchling



TRAUM-NINI

Dokumentation zur Projektarbeit des Schulprojektes von Herr Schmiech und Herr Maier

THEMA: PROJEKTARBEIT
TRAUM-NINI – EINE ANWENDUNG ZUR ERSTELLUNG DEINES PERFEKTEN PANINIS

Inhaltsverzeichnis

1. Ausgangssituation

1.1. Planung

Im Zuge einer Projektarbeit für die Lernfelder 10 – „Benutzerschnittstellen gestalten und entwickeln“, 11 – „Funktionalität in Anwendungen realisieren“ und 12 – „Kundenspezifische Anwendungsentwicklung durchführen“ wurde die Planung am 08.09.2025 angefangen.

Dieses Projekt hat folgende Voraussetzungen:

- Grafische Nutzeroberfläche
- Datenbank mit Schreib- und Lesefunktion
- Schnittstelle zwischen Anwendung und Datenbank
- Freie Wahl der Programmiersprache
- 12-seitige Dokumentation
- Präsentation mit Fachgespräch

Nach kurzer Ideensammlungsphase hatte ich mich für eine Panini-Applikation entschieden. In dieser soll es dann möglich sein, sein Panini zusammenzustellen und eine Bestellung abzuschicken. Außerdem wollte ich es mit Hilfe des Angular Frameworks umsetzen.

1.2. Ausgangslage

Auf Grund meiner Vorlieben zu Hähnchen Paninis, habe ich mich entschieden, ein Programm zu erstellen, welche diese Leidenschaft verkörpert. Nun ist es möglich mit verschiedenen Sorten von Paninis herumzuexperimentieren und zu bestellen. Diese Bestellungen können dann von Administrations Konten eingesehen und angenommen werden.

2. Projektplanung

2.1. Ist-Zustand

Bisher gibt es kein System, um sich Paninis zusammenzustellen, zu bestellen und Bestellungen zu verwalten. Dies passiert lediglich in Person, wenn man sich ein vorgefertigtes Panini direkt kauft.
Eine Datenbank musste ebenfalls aufgesetzt werden.

2.2. Voraussetzungen

Um ein Gelingen des Projektes zu gewährleisten, sind einige Voraussetzungen zu setzen.

Eine dieser Voraussetzungen ist es Erfahrungen mit „Typescript“, „Javascript“ und dem Angular-Framework zu haben um die grobe Programmierung und Struktur umsetzen.

Außerdem sind Kenntnisse zu „HTML“, „CSS“ bzw. „SCSS“ wichtig, da diese die Oberflächenprogrammierung realisieren.

Da die Datenbank auf einem SQLite-Server läuft, sind ebenfalls „SQL“ Grundlagen benötigt, um die Tabellen und Abfragen zu erstellen.

2.3. Definition des Soll-Zustandes

Das Ziel und der Soll-Zustand des Projektes ist so definiert, dass es eine Webanwendung gibt, welche über den Browser zugänglich ist. In dieser Anwendung gibt es die Möglichkeit sich anzumelden, abzumelden, einen Nutzer anzulegen.

Als Nutzer gibt es dann die Option Bestellungen abzuschicken, indem man zuvor ein Panini erstellt mit bestimmten Zutaten. Alternativ kann man auch eine Vorlage auswählen, welche bereits ein zusammengestelltes Panini beinhaltet. Vor dem Abschicken eine Bestellung gibt es dann auch noch die Möglichkeit, eine Vorlage aus der neuen Kreation abzuspeichern.

Auf einer weiteren Seite ist es dann möglich eine Liste, aller Bestellungen eines Nutzers einzusehen mit Informationen zu Preis, Bestelldatum und Bestellstatus.

Ein Administrator hat auf dieser Seite dann noch die Funktion Bestellungen anzunehmen, da dieser alle Bestellungen sieht und nicht nur die eigenen.

Soll Zustand:

- Nutzereinbindung mit einem Administrator
- Möglichkeit ein Panini mit Zutaten in bestimmter Reihenfolge zu erstellen
- Panini als Bestellung abschicken
- Panini als Vorlage speichern
- Vorlagen auf die Erstellungsmaske anwenden
- Bestellungen einsehen
- Bestellungen annehmen

2.4. Meilensteine im Projekt

Meilenstein 1: Angular-Projekt aufsetzen
Zuerst ist es wichtig eine allgemeine Struktur für das Projekt zu etablieren. Sodass nun eine Orderstruktur und eine rustikale Anwendung im Browser zusehen ist. Wichtig hier ist, dass alle Plug-Ins wie z.B. „Tailwind“ funktionieren. „Tailwind“ ist ein CSS-Framework, welches das Anpassen von HTML-Elementen vereinfacht und besser lesbar macht.

Meilenstein 2: Datenbank erstellen
Als zweiten Meilenstein ist das Anlegen der SQLite Datenbank definiert. Hierfür werden die Tabellen erstellt mit ihren Beziehungen untereinander, durch logisch gesetzte Fremdschlüssel.

Meilenstein 3: Nutzer-Authentifizierung implementieren
Da nun eine Datenbank existiert, auf welche man zugreifen kann, ist es möglich, Nutzer anzulegen und abzurufen. Daher kann man sich Anmelden und Registrieren. Hierfür wurden Datenbankabrufe für das Überprüfen von Anmeldedaten, als auch für das Anlegen eines neuen Nutzers erstellt.

Meilenstein 4: Bestellungen zusammenstellen
Als nächster Schritt und Meilenstein wurde festgelegt, dass der Nutzer als nächstes eine Bestellung erstellen und abschicken können soll. Hierfür müssen eine Datenbankabfragen erfolgen, wie z.B. eine für die Auswahl an Zutaten und das Hinzufügen einer Bestellung. Außerdem muss es in der Oberfläche möglich sein, über Knöpfe mit den Zutaten, ein Panini zu erstellen, welches man sehen kann und auch den aktuellen Preis mitgeteilt bekommt.

Meilenstein 5: Bestellungen einsehen
Der fünfte Meilenstein ist eine weitere Unterseite, auf welcher man all seine Bestellungen einsehen kann. Hier kann immer den Status seiner Bestellungen beobachten, sodass der Nutzer weiß, wann seine Bestellung akzeptiert worden ist.
Außerdem muss beachtet werden, dass ein Administrator nicht nur die eigenen Bestellungen einsehen kann, sondern die von allen Nutzern. Außerdem sind bei dieser Art Nutzer noch ein weiterer Knopf in der Oberfläche, um Bestellungen anzunehmen.
Meilenstein 6: Vorlagen erstellen und anwenden
Als letzter Meilenstein ist die Implementierung von Vorlagen angedacht. Hierbei handelt es sich, um die Möglichkeit eine Komposition von Zutaten abzuspeichern, um diese zu einem späteren Zeitpunkt erneut zuladen. Es ist dadurch möglich nicht immer dieselben Zutaten anzuklicken, sondern durch einen Klick diese Vorlage auszuwählen. Infolgedessen muss man eine Abfrage machen, um alle Vorlagen zu laden. Gegebenenfalls auch dessen Details mit Zutaten und noch eine Anfragen, um eine Vorlage zu erstellen.

2.5. Programm Struktur

Ich habe mich dazu entschlossen, ein Programmablaufplan zu erstellen, um eine problemlose Umsetzung des Programmierteils zu gewährleisten.

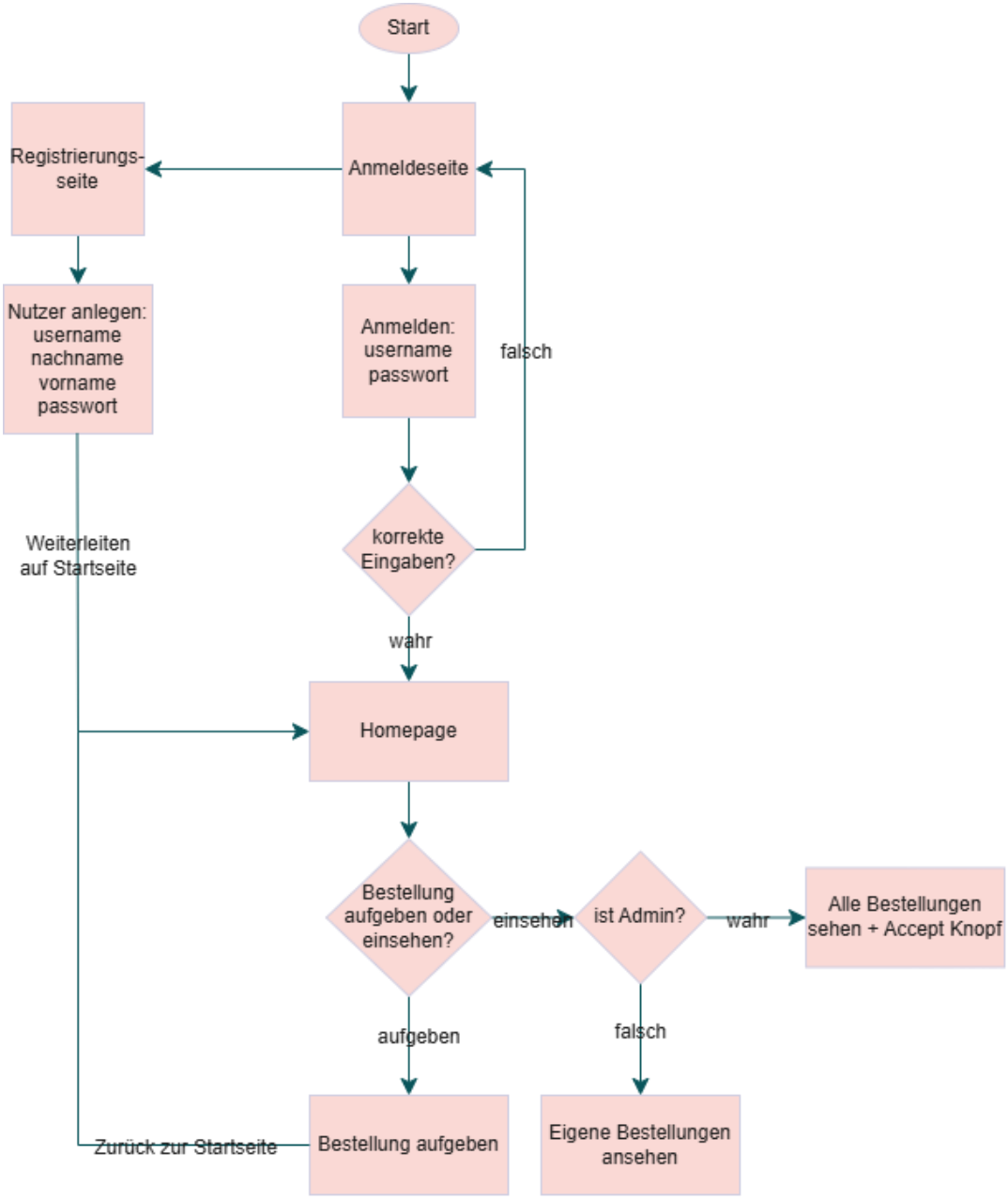


Abbildung 1: Programmablaufplan der Anwendung

2.6. Datenbank Struktur

Für eine unkomplizierte Umsetzung des Projektes wurde die Datenbank Struktur ausführlich ausgearbeitet. Hierzu wurde sich auf sechs Tabellen beschränkt.

Auf Grund von zwei n-m Beziehungen gibt es zwei Kreuztabellen: „orderTable“-„orderPos“ und „template“-„templatePos“.

Tabelle „ingredient“ → Zutaten

- ID (Primärschlüssel)
- Name
- Preis

Tabelle „orderTable“ → Bestellungen

- orderID (Primärschlüssel)
- username (Fremdschlüssel zu „User“-Tabelle)
- timeOfOrder (Zeitpunkt der Bestellung)
- price
- confirmed

Tabelle „orderPos“ → Bestellungenspositionen

- orderPosID (Primärschlüssel)
- orderPos (Fremdschlüssel zu „orderTable“-Tabelle)
- ingredientID (Fremdschlüssel zu „ingredient“-Tabelle)
- position (Reihenfolge, wann es auf das Panini kommt)

Tabelle „template“ → Bestellungen

- templateID (Primärschlüssel)
- name
- username (Fremdschlüssel zu „user“-Tabelle)

Tabelle „templatePos“ → Bestellungen

- templatePosID (Primärschlüssel)
- templateID (Fremdschlüssel zu „template“-Tabelle)
- ingredientID (Fremdschlüssel zu „ingredient“-Tabelle)
- position (Reihenfolge, wann es auf das Panini kommt)

Tabelle „user“ → Bestellungen

- username (Primärschlüssel)
- firstName (Vorname)
- lastName (Nachname)
- password
- isAdmin

Als Hilfestellung eines eindeutigen Verständnisses wurde ein relationales Modell erstellt.

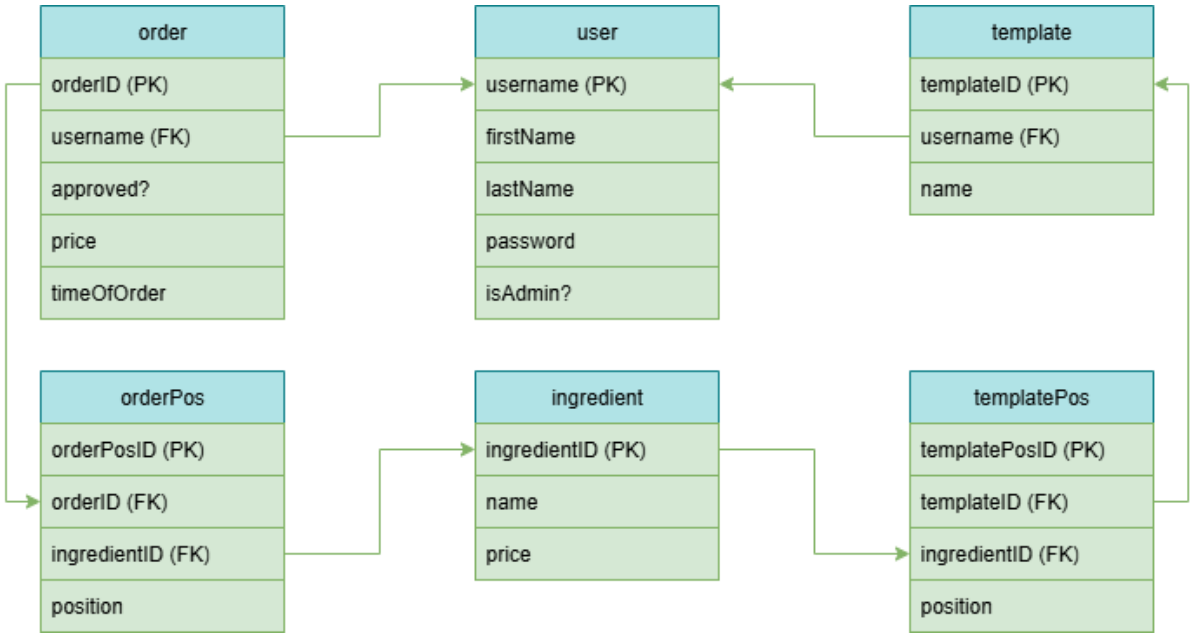


Abbildung 2: Abbildung des relationalen Modells

3. Projektdurchführung

3.1. Aufsetzen der Anwendung

Um das Aufsetzen der Grundstruktur zu vereinfachen, habe ich ich mich entschieden ein bestehendes Angular-Tutorial-Projekt herunterzuladen und als Startpunkt der Programmierung zu nehmen.

Hierdurch wurden einige Aspekte, wie der Grundaufbau einer Komponente klar und einfach replizierbar. Außerdem wurde dadurch das Testen, der Plug-Ins vereinfacht.

Das Hinzufügen von des „Tailwind“-Frameworks hat zwar Anfangs nicht ideal funktioniert, durch falsche Konfigurationen, jedoch konnte man dies durch einige Tutorials einfach beheben.

3.2. Implementierung Datenbank-Schnittstelle

Durch die gute Planung der Datenbankstruktur, ist dieser schnell und einfach gelungen.

Sodass man im ersten Schritt die Datenbank angelegt hat, anhand der erarbeiteten Tabellen.

Als nächstes wurden dann im Projekt eine Javascript-Datei erstellt, welche mit dem Server kommuniziert. Dort befinden sich die auszuführenden SQL-Befehle, die bei Aufruf ausgeführt werden.

Die Dateien werden dann über sogenannte „Services“, die man ebenfalls erstellen muss, aufgerufen. Dort werden nur die „http“-Abfragen verschickt. Diese Struktur ist in Angular üblich und macht Projekte sehr strukturiert, lesbar und erweiterbar.

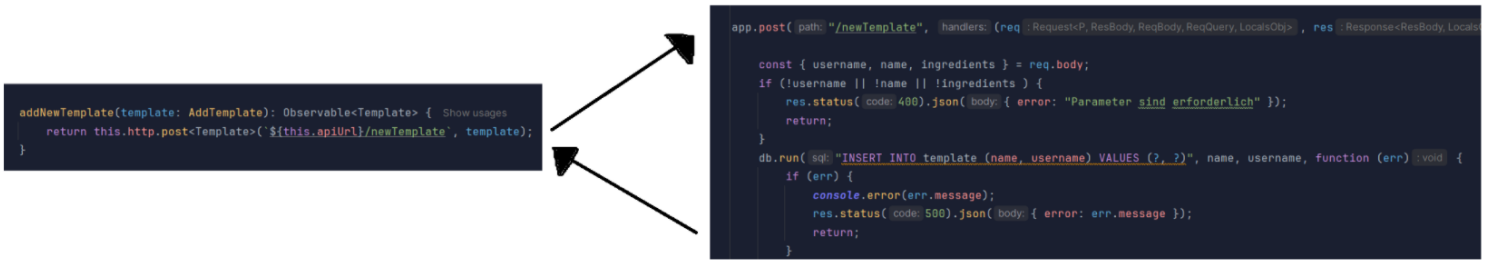


Abbildung 3: Zusammenspiel von Service und Server.js am Beispiel einer Vorlagen-Anlegung

Als weitere Hilfe habe ich einen Ordner angelegt, wo die Datenstrukturen von den unterschiedlichen Objekten definiert werden. Somit gibt es keine Fehler der Datentypen und alles ist wieder gut strukturiert und verwaltbar.

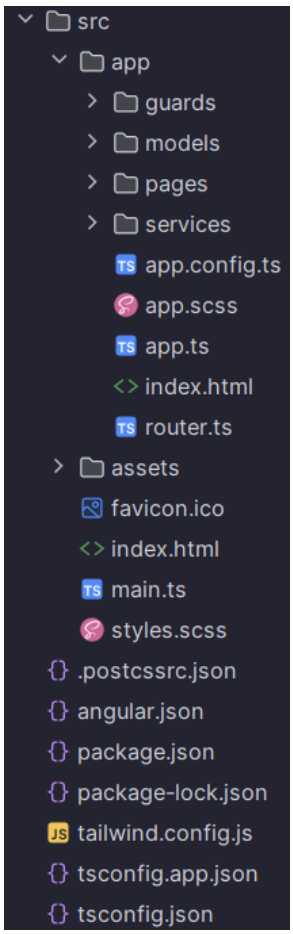


Abbildung 4: Die Orderstruktur des Projektes mit Models (Objektstrukturen), Pages (Komponenten) und Services (Datenbankabfragen)

3.3. Implementierung Authentifizierung

Damit nur angemeldete Nutzer Bestellungen aufgeben und einsehen können wurde ein „AuthGuard“ implementiert, der kontrolliert, ob ein Nutzer angemeldet ist. Ist dies nicht der Fall, wird der Nutzer zum Anmelde Fenster weitergeleitet. Dort kann er sich dann anmelden oder sich registrieren.

Abbildung 7: Anmeldefenster

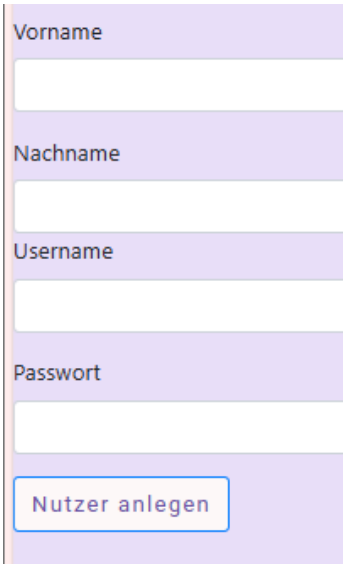


Abbildung 6: Registrierungsfenster

```
{
    path: 'order',
    canActivate: [AuthGuard],
    component: OrderComponent,
},
{
    path: 'viewOrders',
    canActivate: [AuthGuard],
    component: ViewOrderComponent,
},
```

Abbildung 5: AuthGuard authentifiziert Bestellungskomponenten

3.4. Implementierung Bestellungen

Der erste Schritt für die Implementierung der Bestellfunktion, war die Erstellung der Seite. Hier wird ein Knopf angezeigt je Zutat mit deren Namen und dessen Preis in Euro. Wenn man diese Knöpfe drückt, dann wird die Zutat in die Zutatenliste aufgenommen und in der Auswahl angezeigt. Außerdem wird der Preis automatisch aktualisiert.

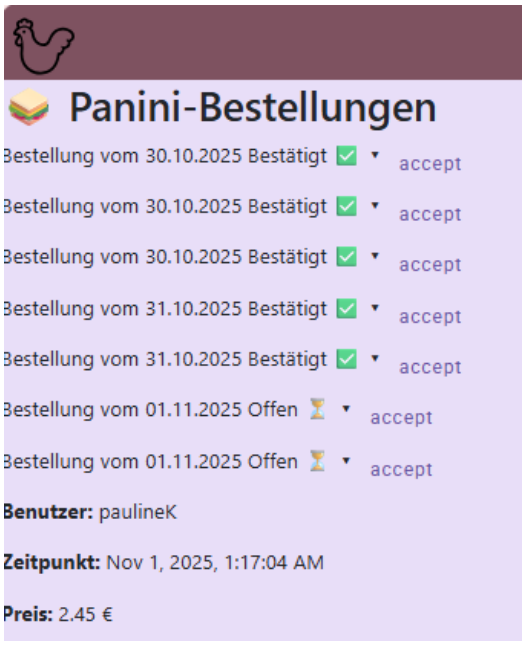
Die Zutaten werden zu Beginn von der Datenbank abgefragt, somit kann man als Administrator mit Datenbankzugriff beliebig viele Zutaten einfach hinzufügen. Damit ist dies sehr erweiterbar gelungen.

Bei dem Drücken des „Panini bestellen“ Knopfes wird dann ein Eintrag in der „Order“-Tabelle und Einträge in der „OrderPos“-Tabelle erstellt. Danach wird man automatisch wieder in das Hauptmenü zurückgeleitet.

Abbildung 8: Bestellungsseite mit Zutatenauswahl

Nachdem man nun Bestellungen anlegen kann, musste logischerweise nun die Bestellübersicht erstellt werden. Hierfür werden zuerst nur alle Bestellungen angezeigt mit Informationen zum Bestelldatum und Bestellstatus. Die Details einer Bestellung werden erst angezeigt, wenn man diese anklickt. Zu den Details gehört der Nutzername des Bestellers, das genaue Datum und der Preis.

Wenn man mit einem Administrator Nutzer angemeldet ist, werden einem ebenfalls neben jeder Bestellung noch ein „accept“-Knopf angezeigt. Wenn man diesen drückt, wird die Bestellung auf der Datenbank als akzeptiert gespeichert und der Status wird angepasst.



3.5. Implementierung Vorlagen

Zuletzt mussten noch die Vorlagen implementiert werden. Hierfür gibt es ein „Dropdown“-Menü, in welchem alle Vorlagen eines Nutzers zuerst von der Datenbank abgefragt werden und dort angezeigt werden. Wenn man eine Vorlage auswählt, werden in dem Panini alle Zutaten hinzugefügt und man könnte unkompliziert die Bestellung so abschicken.

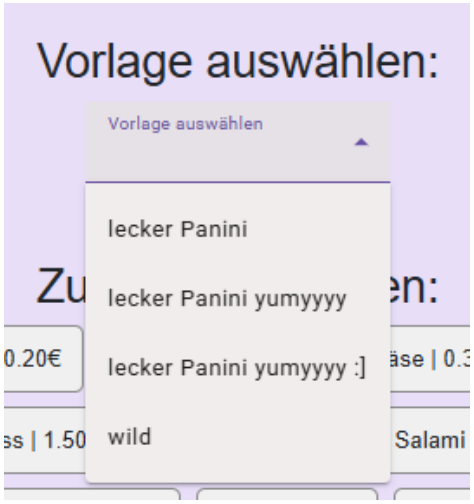


Abbildung 9: Dropdown-Menü der Vorlagen

Will man jedoch eine neue Vorlage anlegen muss man zuerst alle gewollten Zutaten auswählen und dann in dem Feld, neben dem „Vorlage speichern“ Knopf noch einen Namen für die Vorlage festlegen. Falls man keinen Namen festlegt, wird der Standardname verwendet. Die Vorlage ist dann vorhanden, wenn man auf diese Seite das nächste Mal zugreift.

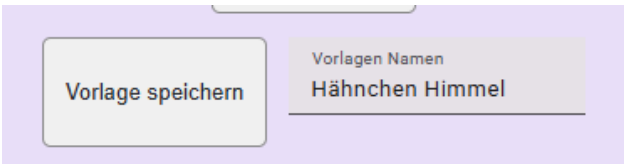
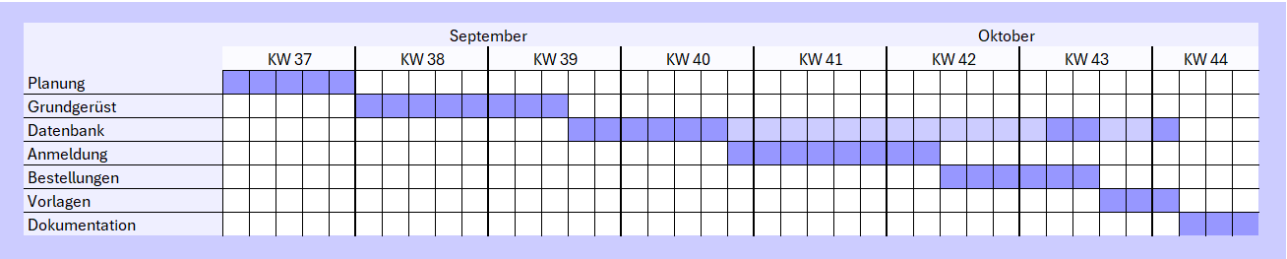


Abbildung 10: "Vorlagen speichern"-Knopf mit Namensfeld

4. Fazit

4.1. Zeitmanagement



4.2. Soll-Ist-Vergleich

Bei einem Vergleich zwischen dem momentan Ist- und den geplanten Soll-Zustand sind alle Funktionen an sich vorhanden. Jedoch muss man erwähnen, dass alle Funktionen zwar funktionieren, aber noch stark ausbaufähig sind. Für eine sinnvolle Nutzung würden beispielsweise noch einige Nutzereingaben-Überprüfungen stattfinden müssen.

Die Soll-Kriterien, die erfüllt sind, waren das Anmelde und Registrierungsverfahren, welche sehr gut funktionieren und durch den AuthGuard auch abgesichert sind.

Ein Weiteres Kriterium ist die Funktion Bestellungen zusammenzustellen und abzuschicken, was an sich auch funktioniert, jedoch noch einfach ausgebaut werden kann. Nichts destotrotz wurde dieses Kriterium erfüllt.

Außerdem kann man zwar sich Bestellungen angucken und als Admin diese auch akzeptieren und allgemein einsehen.

Damit ist die allgemeine Funktion der Webanwendung gewährleisten und es stehen nur noch optionale Funktionen, wie weitere Nutzereingaben Überprüfungen, Sicherheitsfunktionen und erweiterte Bestelloptionen aus.

Was man auch noch erwähnen muss, ist dass die Anwendung nicht besonders visuell ansprechend gestaltet ist. Obwohl ursprünglich viel Zeit in die Implementierung von „Tailwind“ investiert worden ist und sich für Angular aus Design-Gründen entschieden wurde. In dem Aspekt müsste noch viel verbessert werden.

4.3. Probleme

Bei der Umsetzung des Projektes traten einige Probleme auf, welche im Folgenden beschrieben werden.

Das wohl größte Problem war ein schlechtes Zeitmanagement. Es wurde sich sehr lange mit Fehlermeldungen beschäftigt, bis Fehler behoben worden sind, auch wenn es eigentlich sehr einfache Fehler waren. Außerdem wurde zwischen den Schultagen weniger an dem Projekt gearbeitet, als ursprünglich angedacht, aufgrund von mehreren Projektabschlussphasen auf der Arbeit und Urlaub. Dadurch wurde der größte Fortschritt für das Projekt erst am Ende der Projektzeit absolviert, wodurch nun viele nützliche Funktionen offenblieben und nicht genug getestet worden ist.

Desto weiteren wurde zu Beginn die Datenbankstruktur nicht ganz durchdacht, wodurch diese nochmal neu gemacht werden musste es dann zu den Kreuztabellen gekommen ist, welche nochmal etwas Komplexität in die Anwendung bringen.

Außerdem wurde länger gebraucht, um sich in Angular zurechtzufinden, als geplant, da ich lange nichts mehr mit Angular gearbeitet habe. Aber trotzdem davon ausging mich schnell zurecht zu finden, obwohl ich noch nie ein Angular-Projekt von Anfang an aufgesetzt habe. Als das Grundgerüst jedoch stand, merkte man wie viel einfacher es wurde und das die schnelle, unkomplizierte Erweiterbarkeit von Funktionen und Seiten gegeben ist. Angular war möglicherweise nur nicht die Optimale Lösung für ein eher kleines Projekt und eignet sich mehr für größere Projekte, die dann von der Erweiterbarkeit mehr profitieren können.

4.4. Ausblick

Auch wenn die Funktionen nicht so ausführlich und entwickelt sind, wie ursprünglich angedacht, ist die Funktionalität gegeben und das Projekt allgemein erfüllt.

Die Projektvoraussetzungen wurden alle erfüllt, indem es Verbindung zwischen einer Oberfläche und einer Datenbank gibt. Man sich Informationen auslesen lassen, oder neue Informationen dort anlegen.

Man lernte viel darüber, wie wichtig eine gute, durchdachte Projektstruktur und Zeitplanung ist, was man zuvor nicht wirklich bedacht hatte. Aus den gemachten Fehlern kann man nur lernen und das nächste Angular Projekt wird voraussichtlich schneller und reibungsloser ablaufen.

Durch das Projekt wurde viel Verständnis über die Struktur eines Angular Projektes aufgebaut und es konnte viel gelernt werden. Dies ist der Sinn und Zweck dieses Projektes, weshalb es trotz Ausbaufähigkeit ein Erfolg ist.