TAU INF303 Software Engineering Projekt

Projektdokumentation
WS 2021-2022

Projekt: big bite

Product Owner: Önder Tombuş

Autoren:

Name (alphab.)	**Email**
Ahmet Talha LAP	e190501056@stud.tau.edu.tr
Ataberk GÜMÜŞ	e170504008@stud.tau.edu.tr
Berk SADIKOĞLU	e180502039@stud.tau.edu.tr
NAZLI AKKAM	e170503107@stud.tau.edu.tr
Seza Nihan BEKARLAR	e180503001@stud.tau.edu.tr

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
0.0.1	gültig	17/11/2021	Nazlı Arakı	-
0.0.2	gültig	9/12/2021	Berk Sadıkoğlu	-
0.0.3	gültig	11/12/2021	Berk Sadıkoğlu	-
0.0.4	gültig	22/12/2021	Berk Sadıkoğlu	
0.0.5				

**) Sofern im Projekt nicht anders vereinbart sind folgende Statusbezeichnungen zu verwenden (in obiger Tabelle und am Deckblatt):*
Entwurf / in Review / gültig / ungültig

Dokument wurde mit folgenden Tools erstellt:

Microsoft Office Word
Microsoft Excel
Lucidchart
Snipping Tool

<tool-name>

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Wie jeder weiß, arrangieren sich viele große Restaurants mit Lieferunternehmen, die mehrere Restaurants bedienen, anstatt eigene digitale Apps zu erstellen. Deswegen sind sie bei der gewünschten Schnittstelle, der Zahlung oder der Lieferung des Produkts an den Kunden auf andere Organisationen angewiesen. Wenn ein Problem auftritt, wenden sich Kunden zuerst an andere Unternehmen und nicht an das Restaurant. Nämlich mithilfe der entwickelnden Applikation wird unser Restaurant seinen Kunden, die bei ihnen Essen bestellen möchten, die Möglichkeit bieten, direkt mit ihnen zu kontaktieren. Darüber hinaus bleibt bei ihnen dank dieser Anwendung die Steuergebühr, die das Restaurant für andere Anwendungen entrichtet. Tatsächlich haben viele Apps zur Essensbestellung eine immer komplexer werdende Benutzeroberfläche. Dies schafft eine Barriere für Kunden, ihre Bestellungen einfach abzuschließen. Sie verbringen mehr Zeit mit Anwendungen, anstatt schnell auf Produkte zuzugreifen. Daher ist es ein wichtiger Punkt für unsere Aplikation, sehr einfach und verständlich zu sein. Eine einfache und verständliche Benutzeroberfläche, Erklärungen zu Produkten, Lieferservice, verschiedene Zahlungsmethoden und viele zusätzliche Funktionen werden mit BigBite verfügbar sein. Beispielsweise im Hauptseite wird der Kunde sehen können, welche Produkte und Sorten gerade im Restaurant verfügbar sind und das gewünschte Produkt in seinen Warenkorb legen. Wenn Sie den Bestellvorgang abschließen, erscheint die Zahlungsmaske, in der der Kunde die Bereitstellung- und Lieferungskosten sehen. Benutzer können uns durch Rezensionen ihre Meinungen und Empfehlungen über ihre Bestellungen und Produkte, die sie mögen oder nicht mögen, als Kommentar mitteilen. Mit dem von uns erstellten Bewertungssystem können Kunden ihre Erfahrungen mit uns teilen. Wir möchten diese Anwendung so gestalten, dass sie in alle Betriebssysteme integriert werden kann, nicht nur in bestimmte Betriebssysteme, damit jeder Kunde, der das Essen des Restaurants liebt, jederzeit bestellen kann.

[REQ1] Motivation ist ausführlich dargestellt, ca. ¾ Seite.

1.2 Zweck des Dokuments

Der Zweck unserer Dokumentation ist es, die Projektplanung zu dokumentieren, die Anforderungsanalyse und die Architektur mit Diagrammen zu erstellen und Deployment View zu zeigen. Die Anforderungsanalyse werden mit verschiedene Szenarien erstellt. Um technische Eigenschaften und geforderte Funktionen des Projektes zu schaffen, wird die entschiedene Software Technologien bestimmt.

1.3 Begriffsbestimmungen und Abkürzungen

UD	Use Case Diagram
UML	Unified Modelling Language

Auflistung von Begriffsdefinitionen und Abkürzungen, die für das Verständnis der Grobspezifikation wichtig sind (EDV-Fachausdrücke und Begriffe aus der Anwendungsdomäne) - unabhängig davon, ob diese in einem anderen Dokument (z.B. Pflichtenheft) bereits erklärt wurden.

Beispiel:

COTS	*Commercial Off-the-shelf Software (Kaufsoftware, Standardsoftware, OSS)*
IT	*Informationstechnologie*
DD	*Deployment Diagram*

AD	*Activity Diagram*
CD	*Class Diagram*
SD	*Sequence Diagram*
UD	*Use Case Diagram*

2 Projekt Management

Dieses Kapitel dient dazu die für die Produktrealisierung notwendigen Projektaktivitäten und Entwicklungsaufgaben zu definieren und planen. Die Planung wird in Bezug auf die agilen Entwicklungsmethodik „Scrum“ definiert.

Ziel ist es, die oben definierten Aufgaben (tasks) zu den Scrum Sprints zuzuordnen, wobei als die Sprint-Dauer ein Zeitfenster von 2 Arbeitswochen (10 Arbeitstage) empfohlen wird.

[REQ2] Als Projektanforderung soll die Darstellung der Sprints mit Zeitangaben in der vorgegebenen Excel Vorlage erfolgen, dabei sollen die Meilensteine berücksichtigt werden, siehe INF303-[projektname]-Sprint-Plan.xlsx

3 Produktumfang

In diesem Teil beschreiben wir die Architektur unseres Produkts explizit anhand von UML-Diagrammen und detaillierten Anwendungsszenarien. Die Nutzung des Produkts setzt voraus, dass die Kunden, Restaurant zumindest während der Nutzung über eine dauerhafte Internetverbindung verfügen und das Softwareprodukt installiert haben, damit sie nach der Registrierung eines Kontos mit ihrer geteilte Personal Informationen die gewünschten Aufgaben ausführen können. Die Verwendung der ist E-Mail Adresse notwendig und der sicherste Weg, um alle Bestellungen eines Kunden und alle vorbereitete und gelieferte Bestellungen für den Restaurant, was die Nutzung sicher und einfach macht. Es gibt viele Szenarien für die Verwendung dieser Eigenschaften, die jeweils angesprochen werden, und wir werden die am einfachsten zu verstehenden und wesentlichen Verwendungen bereitstellen.

Dieses Kapitel dient dazu
- *das Produkt zu beschreiben*
- *die Anforderungen zusammenzufassen*
- *die Einschränkungen / Randbedingungen für die Entwicklung der Architektur zu beschreiben*

Ziel ist es, alle Einflussfaktoren für die Festlegung der Architektur und der damit zusammenhängenden Design-Schritte zu sammeln.

3.1 Technische Infrastruktur

Wie fast jeder Entwickler für mobile Anwendungen haben wir auch damit begonnen, Flatter-App über Android Studio zu erstellen. Android Studio hilft uns auch, die Funktionalität unserer Anwendung sofort mit Emulator Unterstützung zu testen. Wir haben uns für MySQL als Datenbank entschieden, wir organisieren unsere Daten sowohl über MySQL Workbench als auch über PhpMyAdmin. Wir stellen die Verbindung zwischen unserer Datenbank und unserer App über die Xampp webserver service. Wir benutzen als Framework Laravel 8. Für die Laravel 8 organisieren wir unsere Code durch PhpStorm.

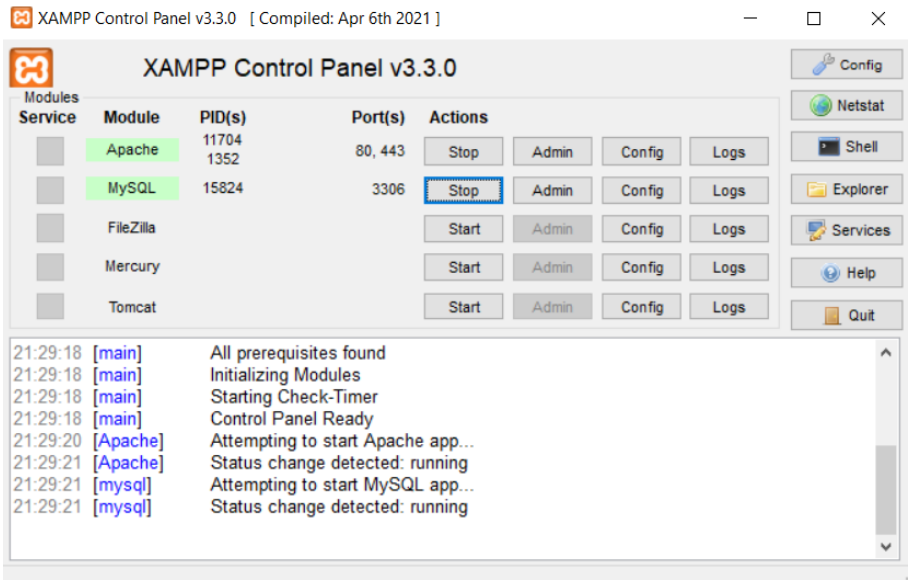


Abb. 1: Xammp

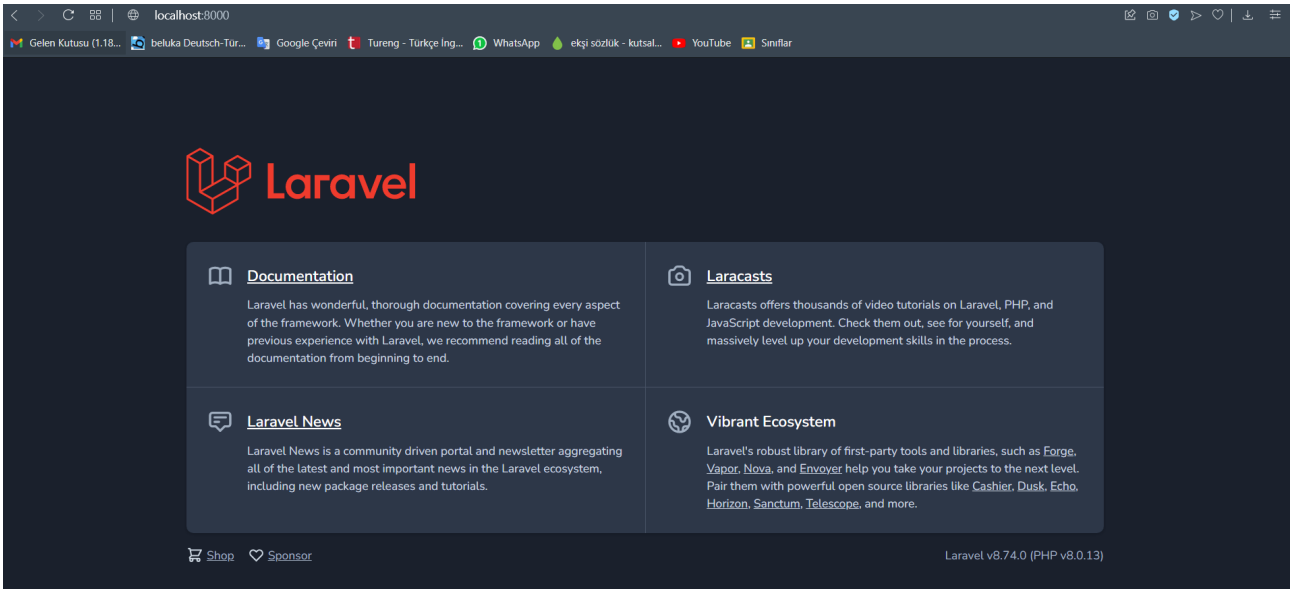


Abb. 2: Laravel

mithilfe von Xammp besorgen wir dafür, zu Web Server für Datenbank und Laravel zu erreichen und mithilfe von Laravel, was ein Framework ist, kann man die database, localhost und API noch einfacher implementieren.

In diesem Abschnitt werden die grundlegenden Elemente zur Entwicklung der Architektur des Produkts beschrieben. Die folgenden Fragen zeigen Beispiele, die zu berücksichtigen sind:

- *Was ist die Laufzeit- und Entwicklungsumgebung des <Systems/Produkts>?*
- *Welche technischen und funktionalen Komponenten von Drittanbietern werden verwendet?*
- *Welche vorhandenen Komponenten werden wiederverwendet und welche ihrer Eigenschaften werden verwendet?*
- *Welche Grundsätze verfolgt die Architektur?*
- *etc.*

3.2 Anforderungsanalyse / Use Cases

Was sind die wichtigsten Anwendungsfälle / Use Cases des BigBites?

Die Use Cases sollten Interaktionen zwischen einem Akteur (ein Benutzer oder ein externes System) und dem <Produkts> präsentieren. Die Use Cases zeigen Einflüsse aus und Anforderungen gegenüber anderen Systemen. Typischerweise werden die Use Cases mit Szenarien in tabellarischer Form beschrieben.

[REQ3.2.a] Als Projektanforderung sollen die Use Cases und Akteure in UML Use Case Diagram dargestellt werden.

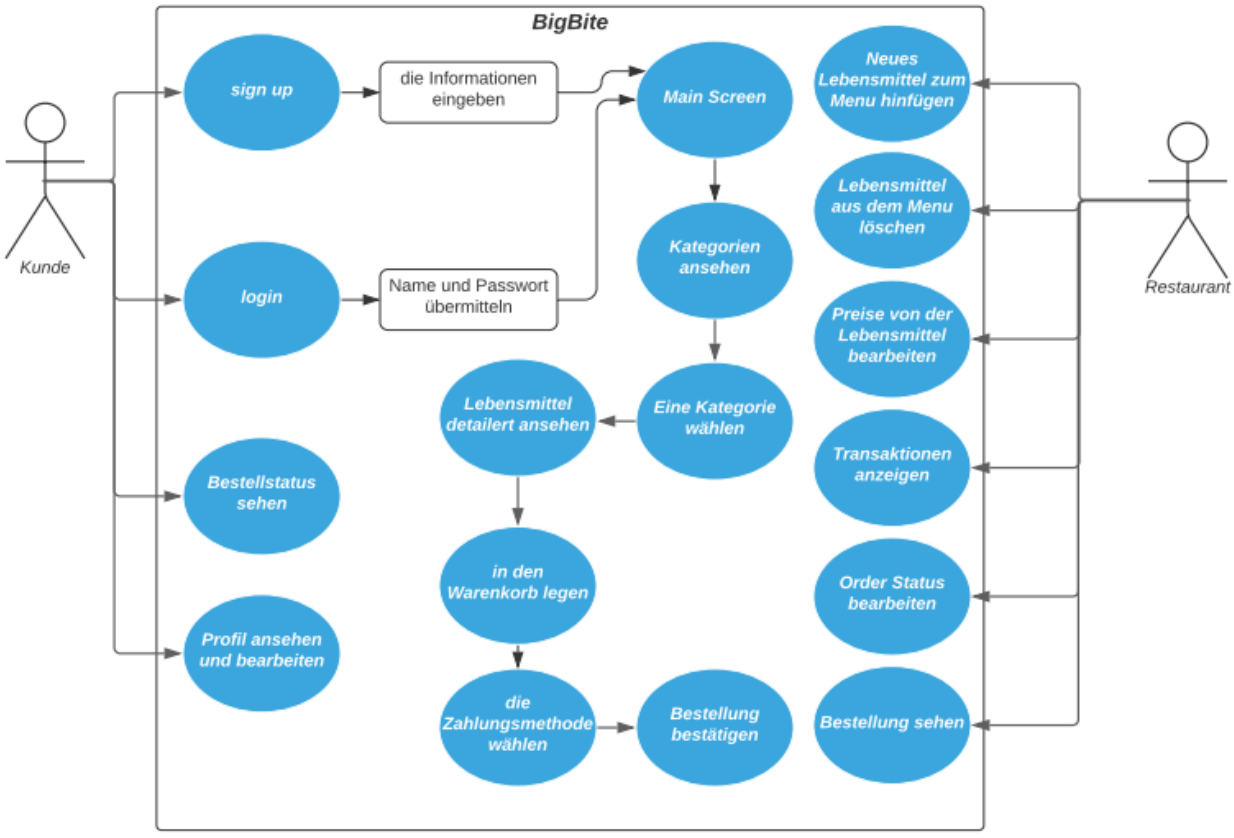


Diagramm.1: Use Case Diagram

[REQ3.2.b] Als Projektanforderung sollen die Szenarien in UML Activity Diagrams dargestellt werden (Mindestanzahl: 2)

<UseCaseID>	<Name>	<Beschreibung>
SZENARIO.1	Essen bestellen	Der Benutzer will ein Essen bestellen.
Anfangs bedingungen		
Schritt 1	Benutzer meldet sich an	
Schritt 2	Kategorien ansehen	Benutzer klickt auf den Kategorie-Button
Schritt 3	Kategorie Auswählen	Benutzer sieht die Kategorien und wählt eine aus
Schritt 4	Optionen der Kategorien auswählen	Seine Optionen wird zum Warenkorb eingetragen
Schritt 5	Benutzer klickt auf Warenkorb-Button	Benutzer sieht seine Wahlen in den Warenkorb
Schritt 6	Bestellung bestätigen	Benutzer bestätigt seine Bestellung
Schritt 7	Lieferung der Bestellung zur Restaurant, und Status Bearbeitung	Die Bestellung wird zu dem Restaurant geliefert und Restaurant teilt Bestellungsstatus mit dem Benutzer mit.

Tabelle 3-1: Tabelle der Use Case Szenario.1

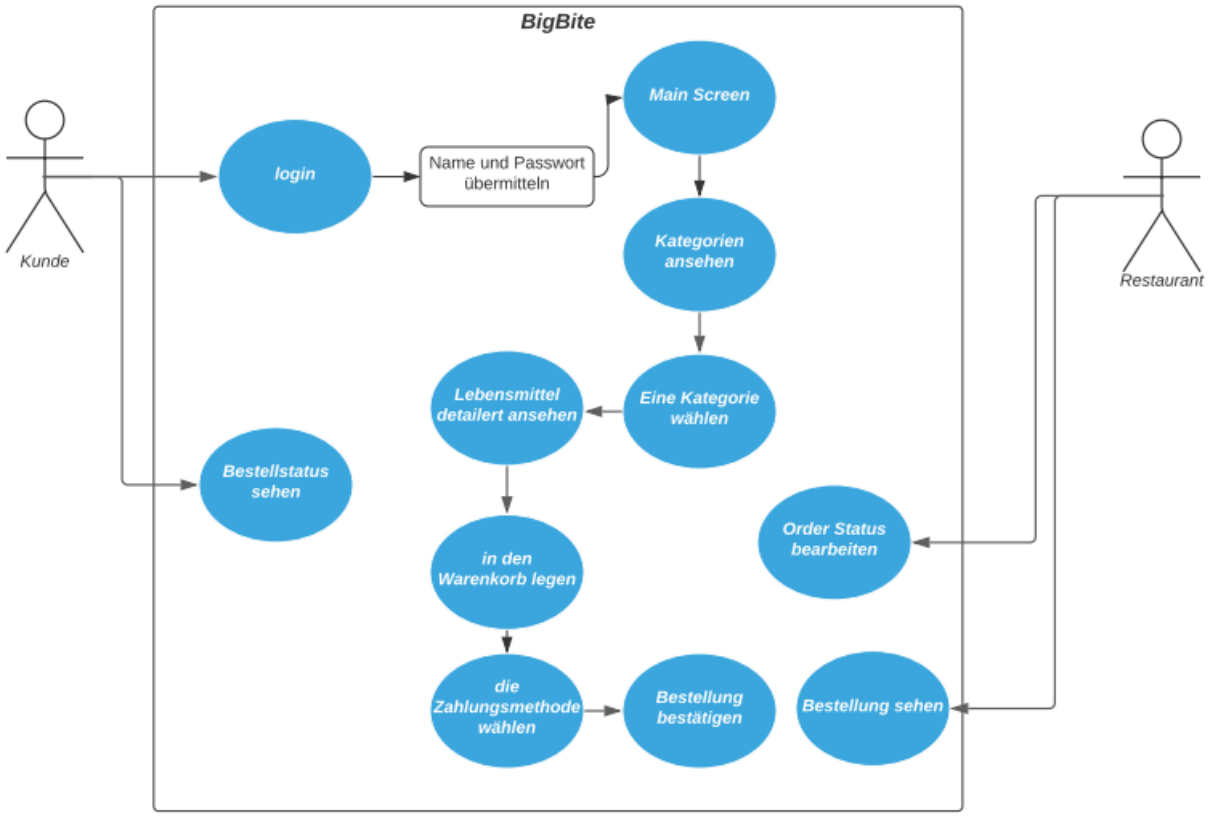


Diagramm.2: Use Case Diagramm für Szenario.1

<UseCaseID>	<Name>	<Beschreibung>
SZENARIO.2	sich registrieren	Personal Informationen ins Applikation eintragen
Anfangs bedingungen		
Schritt 1	Benutzer klickt auf den Sign up-Button	
Schritt 2	Die Benutzerinformationen werden bei BigBite nachgefragt.	ausfüllbare Felder werden erstellt
Schritt 3	Benutzer füllt die notwendigen Informationen aus	Benutzer teilt seine personale Informationen mit
Schritt 4	Benutzer wurde registriert	Benutzer speichert seine personale Informationen

Tabelle 3-1: Tabelle der Use Case Szenario.2

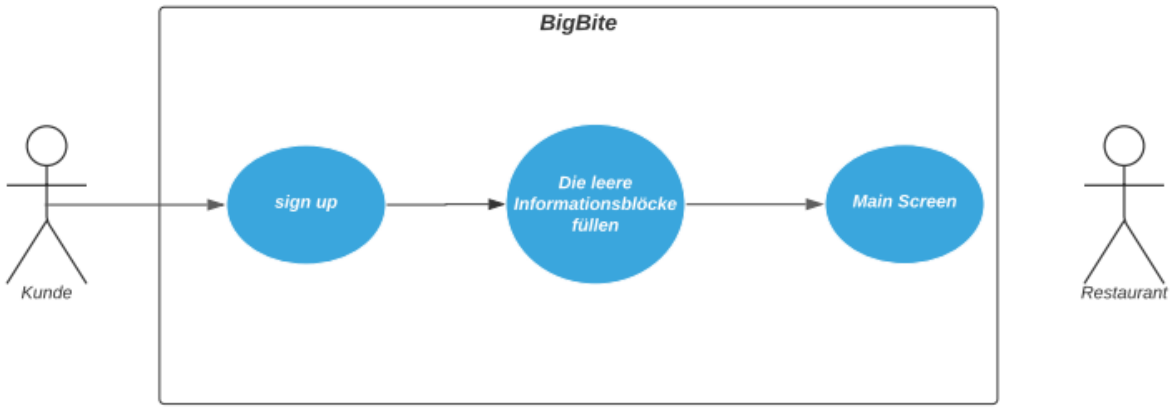


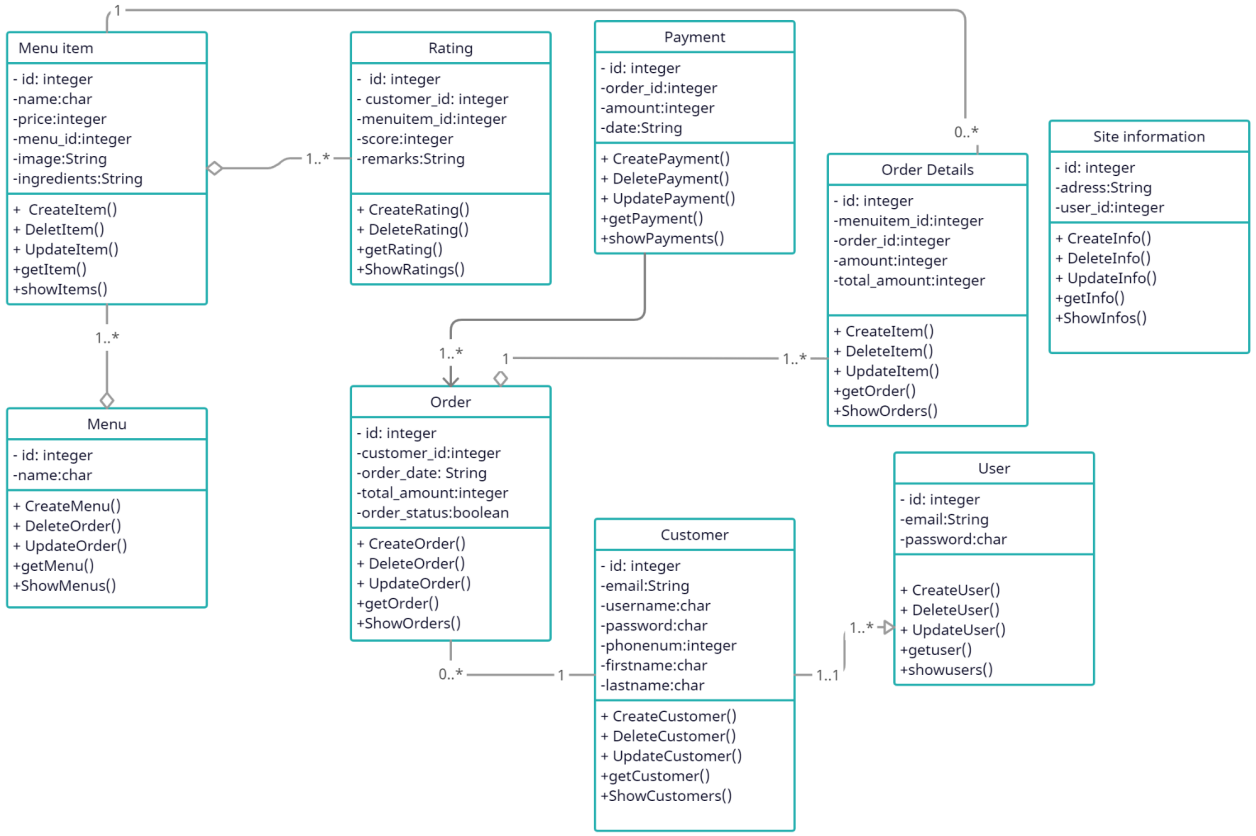
Diagramm.3: Use Case Diagramm für Szenario.2

Was sind die wichtigsten Datenstrukturen / Use Cases des BigBites?

Die Datenstrukturen sind die Bausteine der Informationen. Sie werden in der Kommunikation mit der Umgebung gebraucht und sollen persistent abgespeichert werden, dafür soll das Domänenmodel des Produkts entworfen und mit dem Stakeholder abgestimmt werden. Das Domänenmodel soll die Datenstrukturen an den externen Stellen und in der Datenbank enthalten.

[REQ3.2.c] Als Projektanforderung soll das Domänenmodel in UML Class Diagramms dargestellt werden.

Diagramm 4:Class Diagram



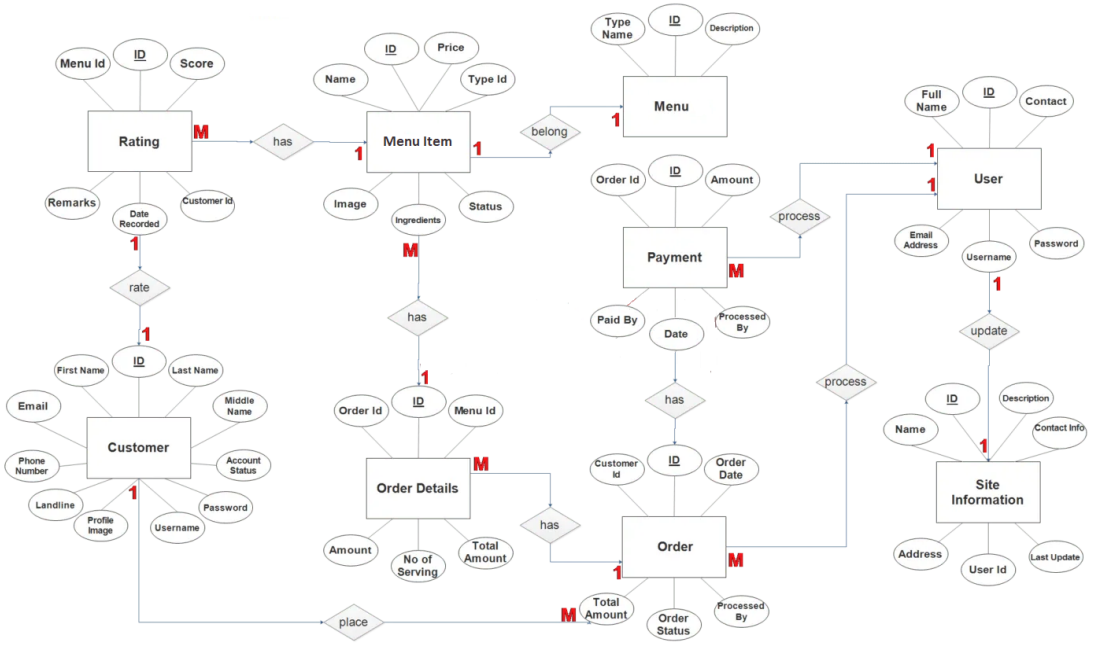


Diagramm 5: ER Diagram

3.3 Architektur

Diagramm 6: eine simple Beziehung zwischen Kunde und Restaurant

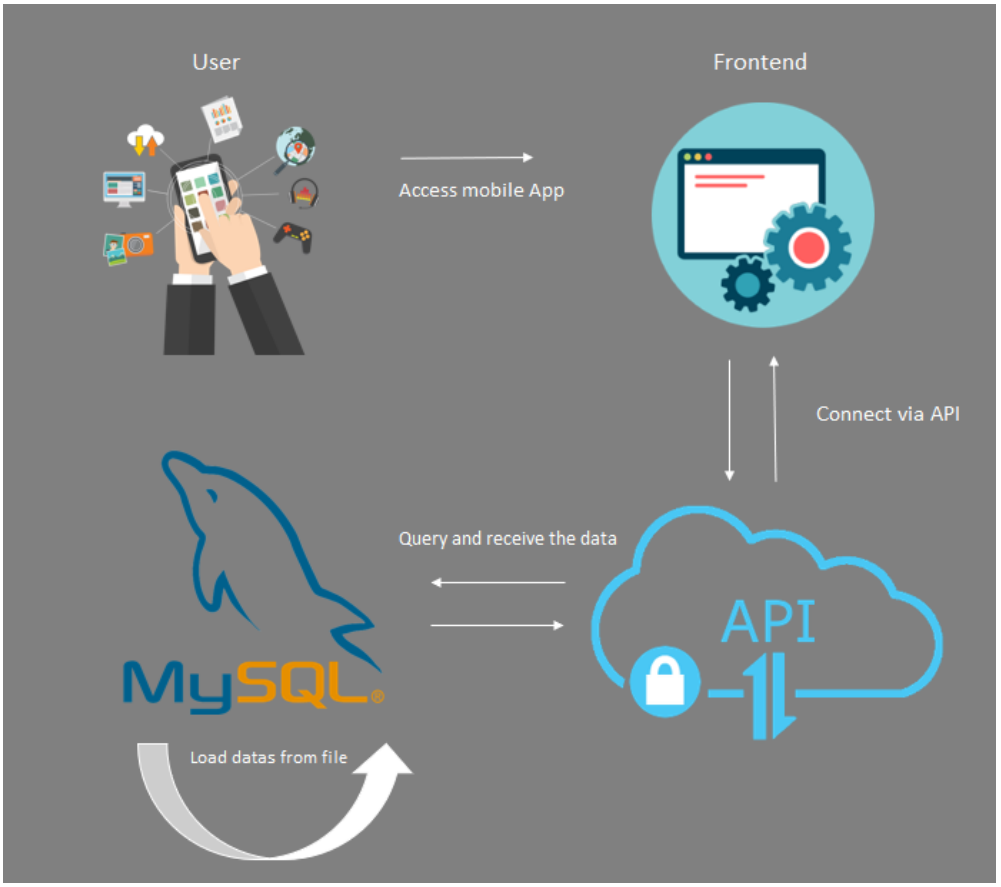


Abb. 3: Systemarchitektur

3.3.1 Design und Entwürfe

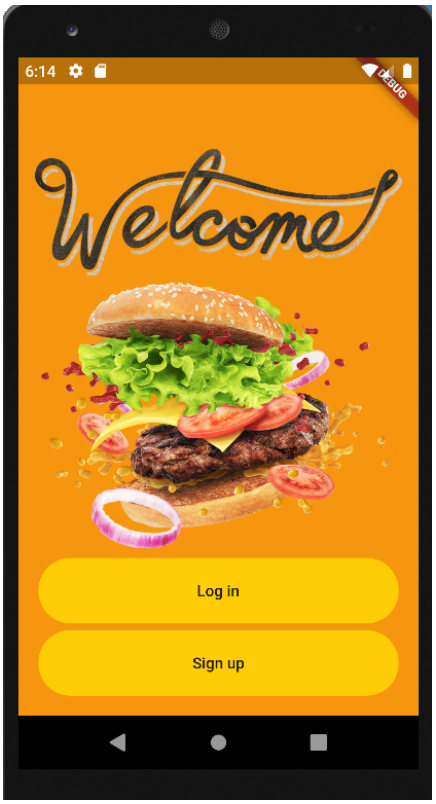


Abb. 4: Entwurf der Hauptseite aus Emulator

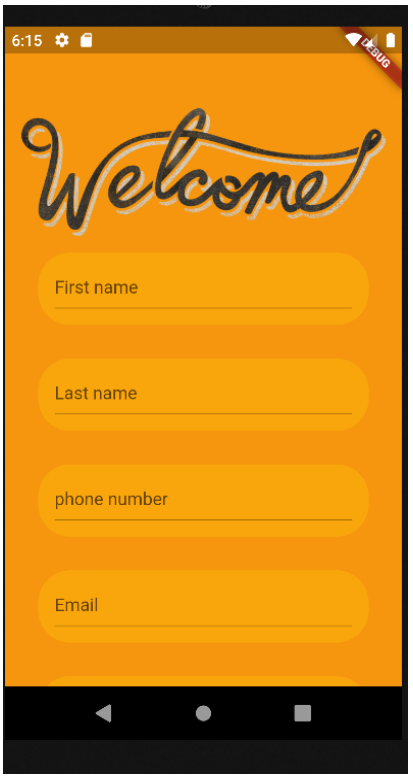


Abb. 5: Entwurf der Registrierungsseite

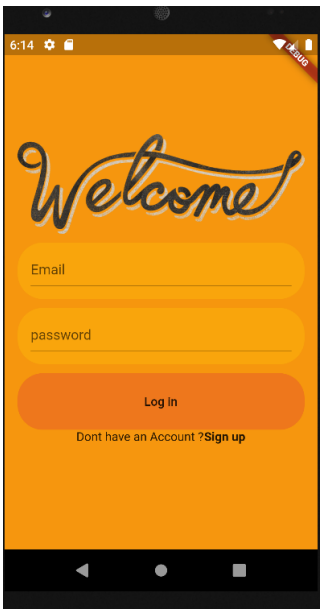


Abb. 6: Entwurf der LogIn Seite

Entwürfe wurden durch Flutter programmiert. Hier kann man einfach die Screenshots aus der Flutter Emulator sehen.

Hauptseite:

Nu	Benutzer	Anwendungsfall	Ziel
1	Kunde	Login	Leiten Sie den Benutzer zum Kunden-Login-Bildschirm.
2	Kunde	Sign up	Leiten Sie den Benutzer zum Registrierungsbildschirm.

In diesem Abschnitt wird die „Zerlegung“ / Strukturierung (Dekomposition) des <Projekts> gezeigt: seine Teile, deren Verantwortlichkeiten /Funktionen, Gruppierung, Beziehungen und wichtigsten Interaktionen - das „big picture“!

Die wichtigsten Teile der Architektur sollen identifiziert und beschrieben werden (z.B. Teilsysteme, Services, Module oder Komponente) und ihre strukturellen Beziehungen (z.B. mit UML-Komponenten- Diagrammen) sowie die Beziehungen im Verhalten (z.B. mit UML-Interaktionsdiagramme) dargestellt werden.

Die wichtigsten Teilsysteme des <Produkts> sollen genau beschrieben werden.

[REQ3.3.a] Als Projektanforderung sollen die Interaktionen zwischen den Teilsysteme in UML Sequence Diagrams dargestellt werden, dabei soll ein Use Case als „trigger“ verwendet werden (Mindestanzahl: 1)

[REQ3.3.b] Als Projektanforderung soll die Beschreibung durch „inline documentation“ erfolgen!

3.3.4 Database Tables

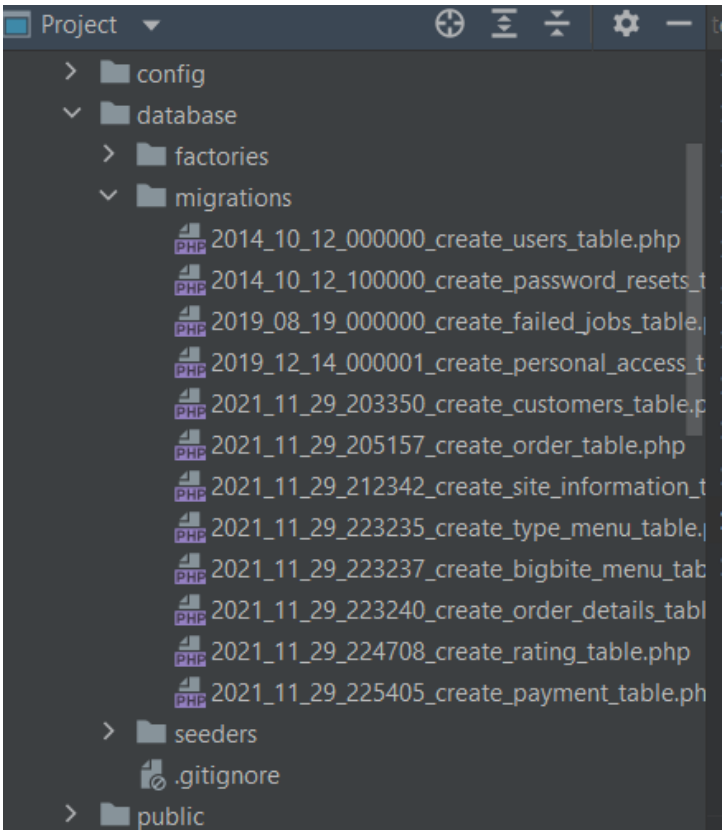


Abb. 7: Database Tables durch Php

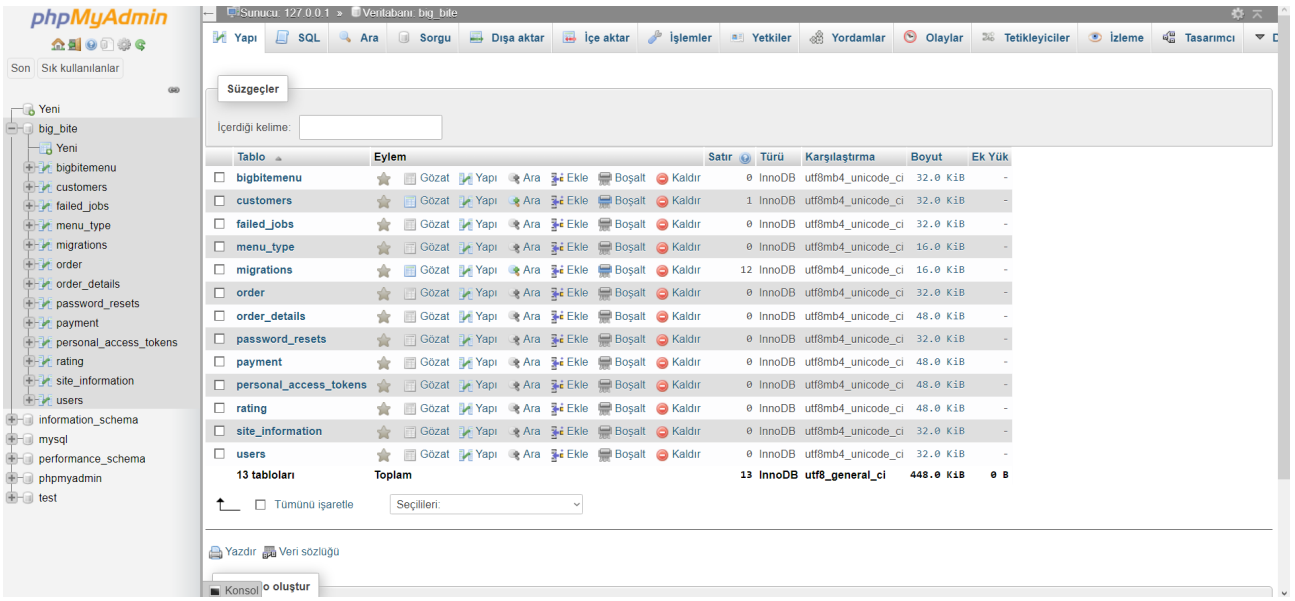


Abb. 8: Aussehen von Database Tables in phpMyAdmin

3.3.5 Admin Panel

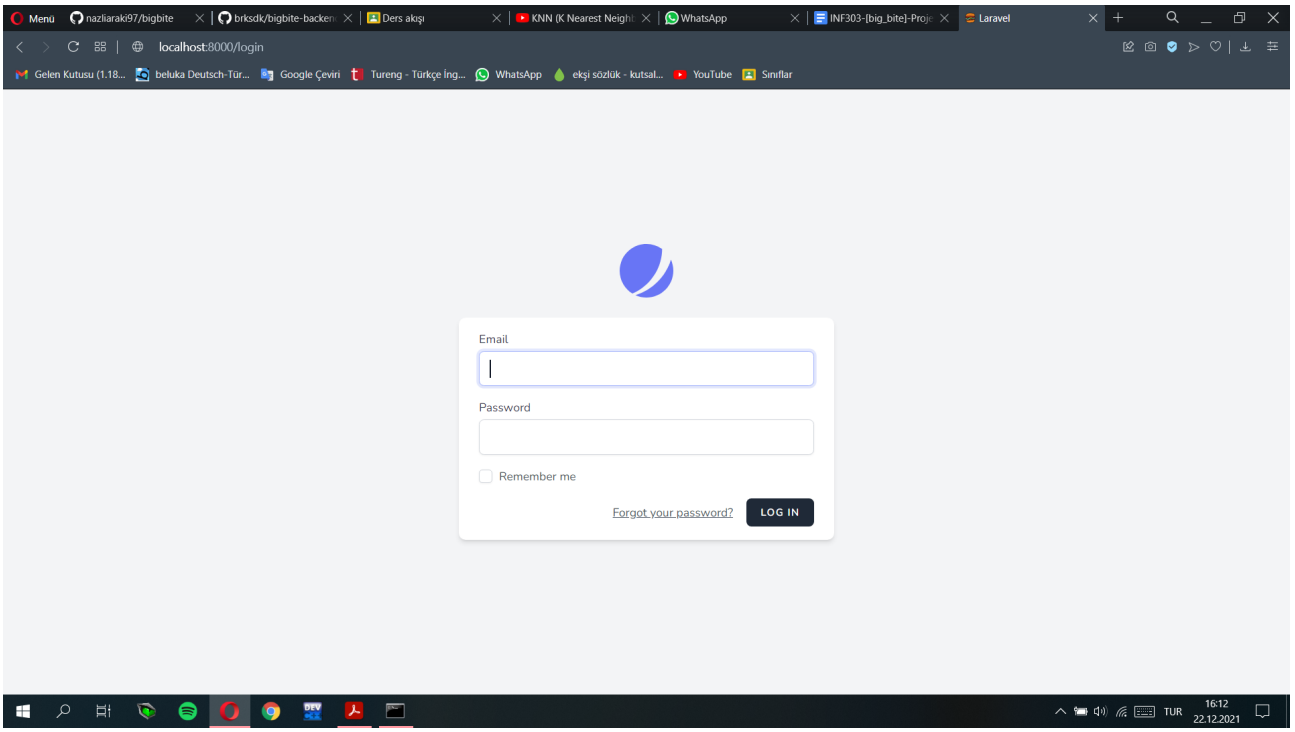


Abb.9 Admin Panel login Seite

Wenn man admin ist, kann einfach das Panel verfügen, und für mobile App kann es reparieren und bearbeiten, was notwendig ist.

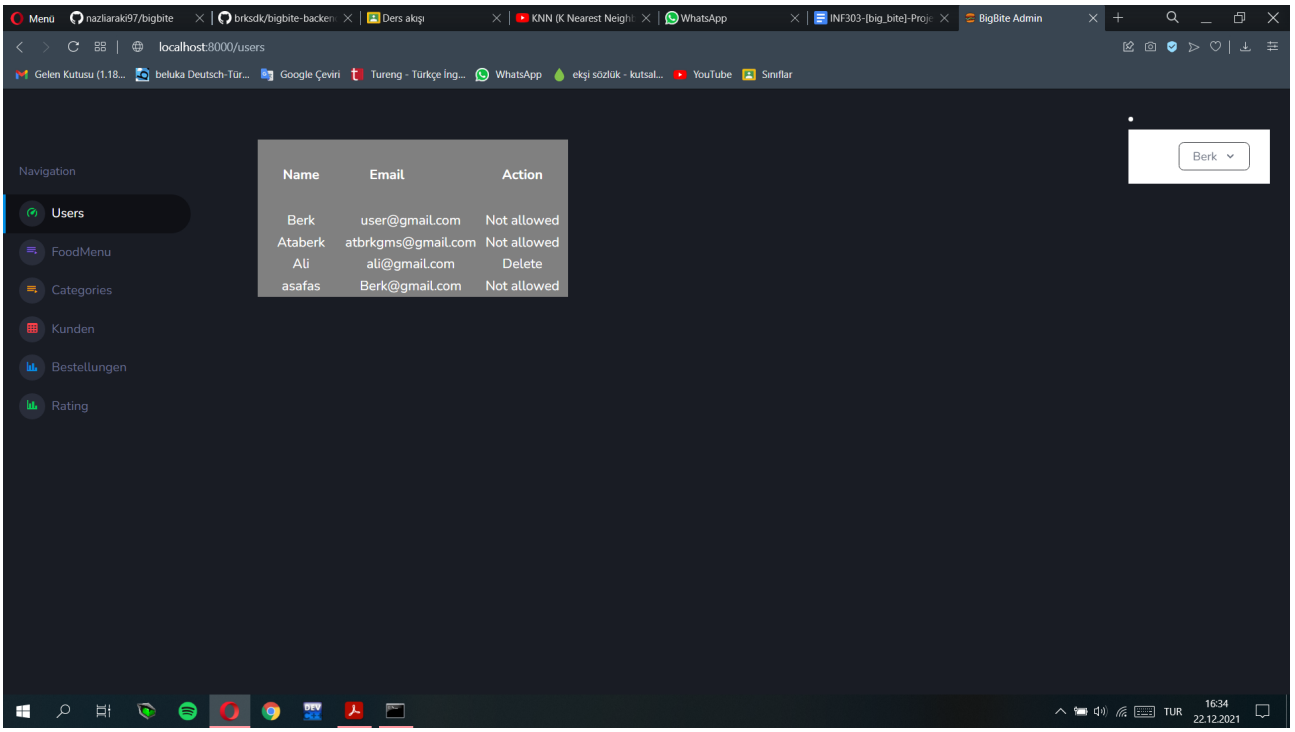


Abb. 10 Admin Panel User Seite

in dieser Seite kann Admin Users bearbeiten, welche sich im Datenbank befinden.

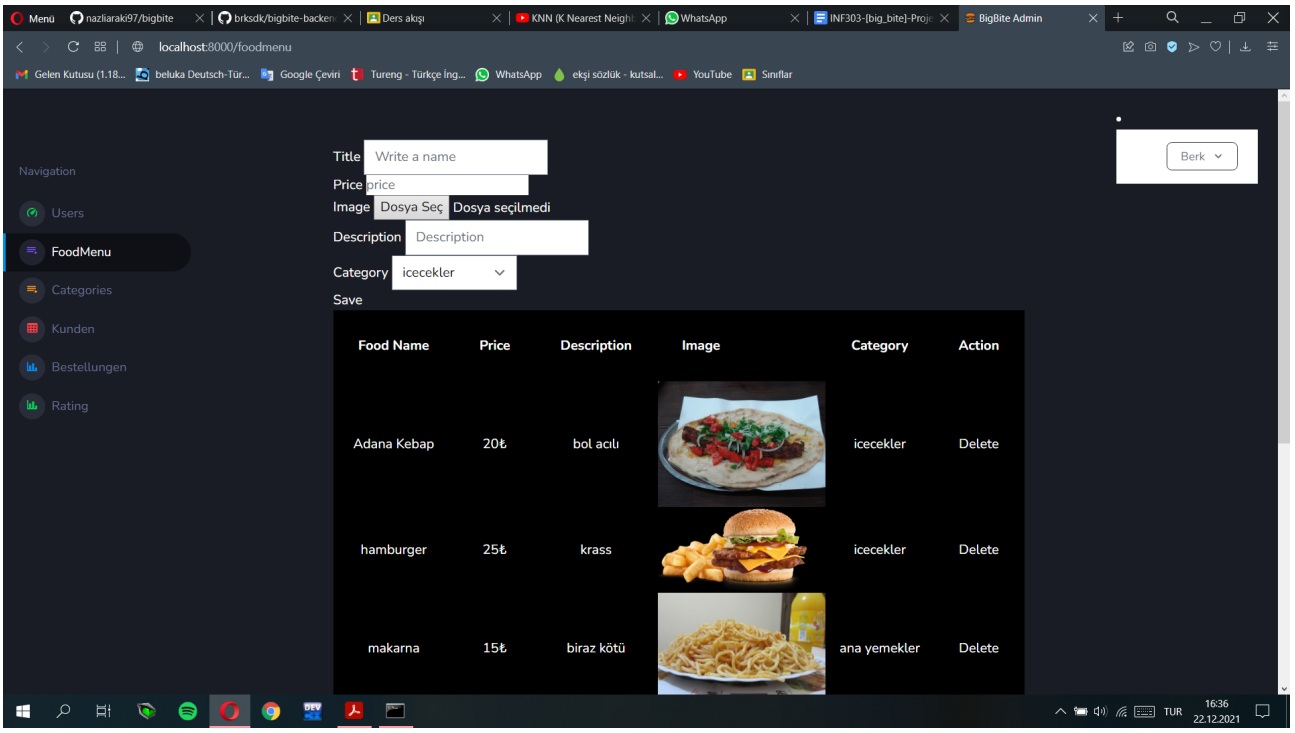


Abb. 11 Admin Panel Foodmenu Seite

in dieser Seite kann Admin Foods bearbeiten, wenn er wollt, kann er diese Foods entfernen, oder auf der Seite und dem Datenbank neues hinzufügen.

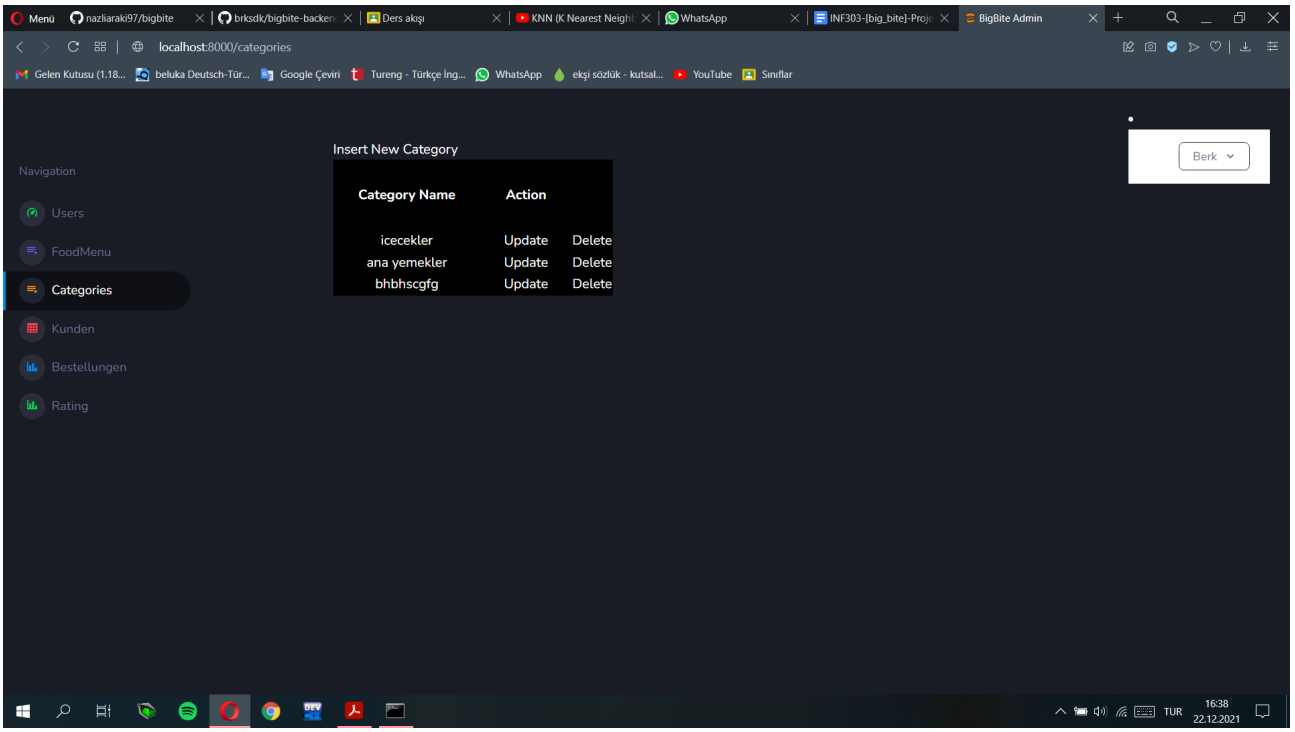


Abb. 12 Admin Panel Categories Seite

Hier darf Admin die Categories entfernen, neues Category hinzufügen oder auf dem neuesten Stand bringen.

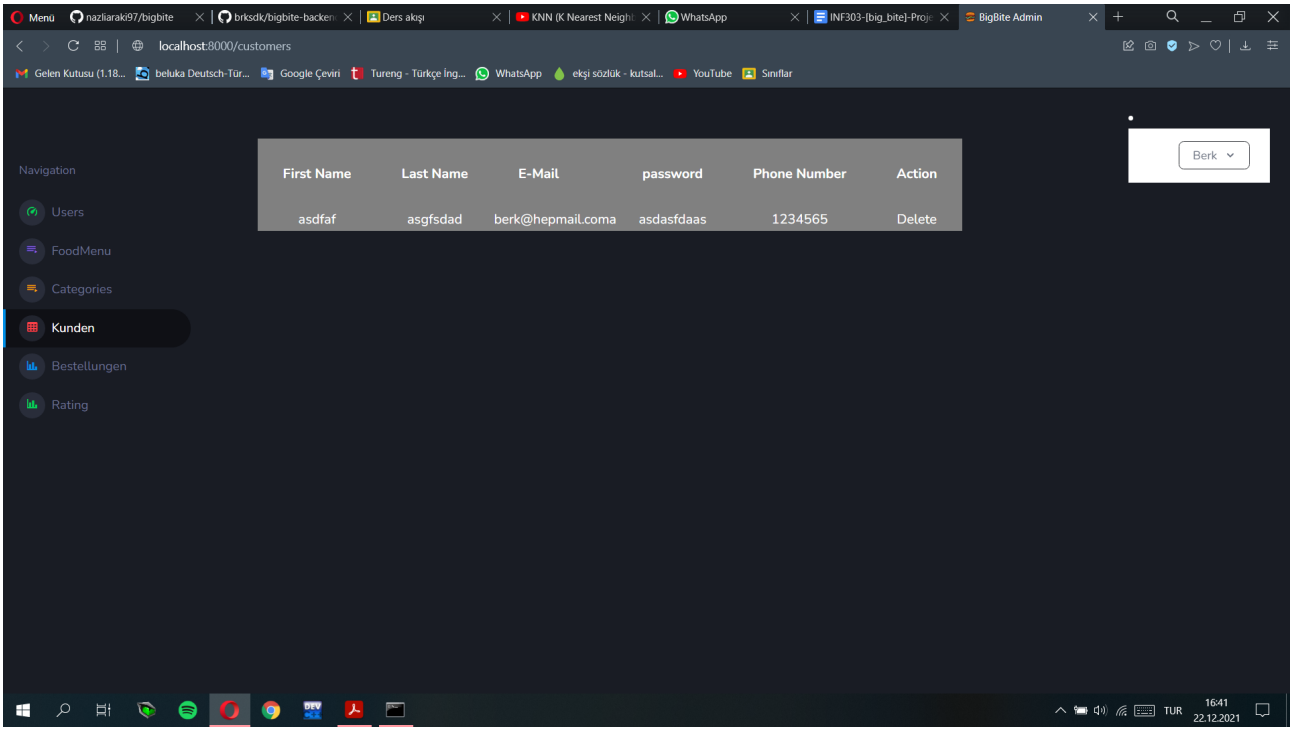


Abb. 13 Admin Panel Kunden Seite

Admin kann die Kunden des Restaurants anzeigen, und wenn es notwendig ist, kann auch entfernen.

3.4 Deployment View

Technologien:

Front-End-Technologien:

- **Flutter**
- **HTML**
- **CSS**

Database:

- **MySQL**

Backend:

- **PhpMyAdmin**
- **Laravel**

In diesem Abschnitt wird auf die „geographische“ (räumliche) Verteilung beim Einsatz des Produkts eingegangen (auf welcher physikalischen Hardware?). Relevant sind dabei die Struktur der Hardware-Bestandteile (Computer, sonstige Geräten, CPUs, …), die erforderliche Netzwerk-Infrastruktur, die Topologie, verfügbare Kanäle, Partitionen und Prozesse.

[REQ3.4] Als Projektanforderung soll die Verteilung beim Einsatz des Produkts mit UML-Deployment Diagram dargestellt werden.

4 Teststrategie und Testplanung

4.1 Register Test:

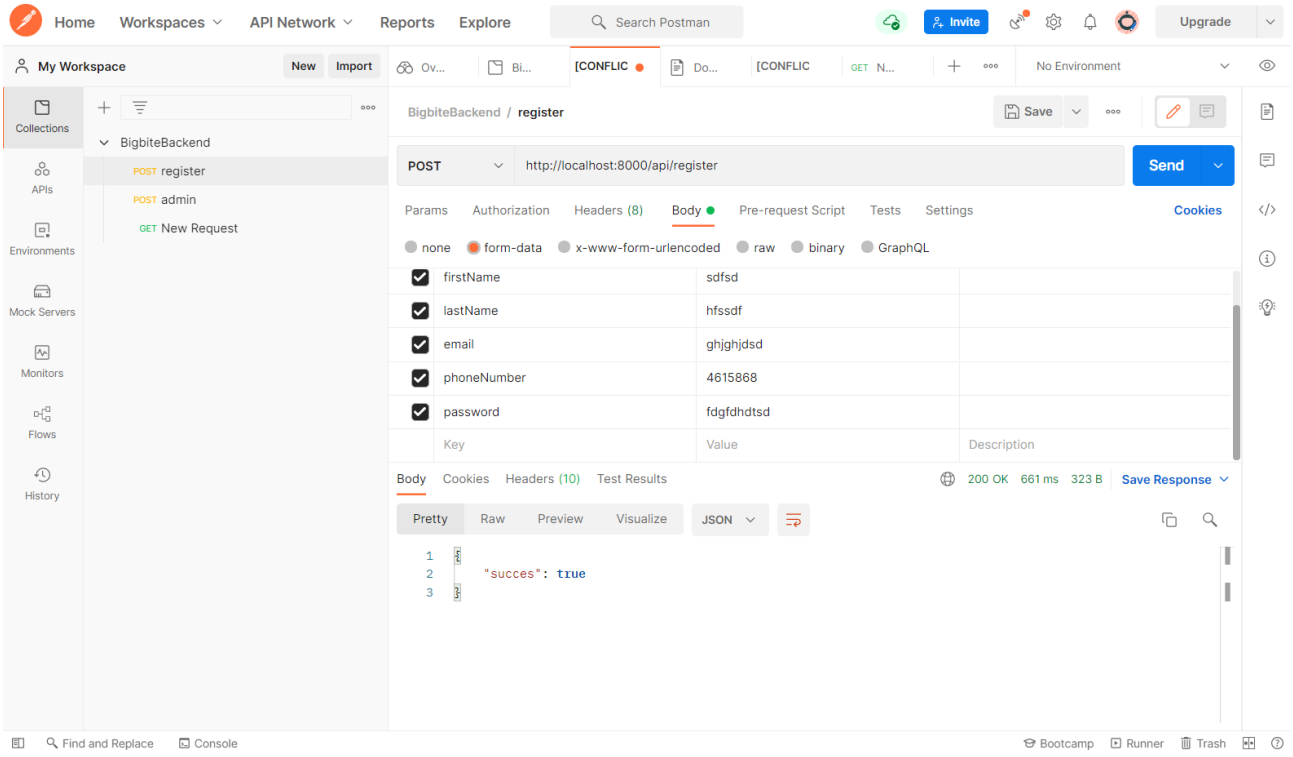


Abb. 14: Register Test durch Postman

Test Name	Customer Register Test
Testszenario	**Eine Post-Anfrage zum Routenpfad „/register“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**Kein Benutzer mit dem folgenden E-Mail schon in der Datenbank vorhanden ist: email@gmail.com**
Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Postman wird benötigt, um HTTP-Anfragen zu erstellen sowie deren Antworten zu lesen**
Testablauf	**Benutzer registriert sich mit seinem Email und Passwort**
Nachbedingung	**Server hat mit „true“ beantwortet.**
Testergebnis	**Erfolgreich**

4.2 Duplicate Customer Test:

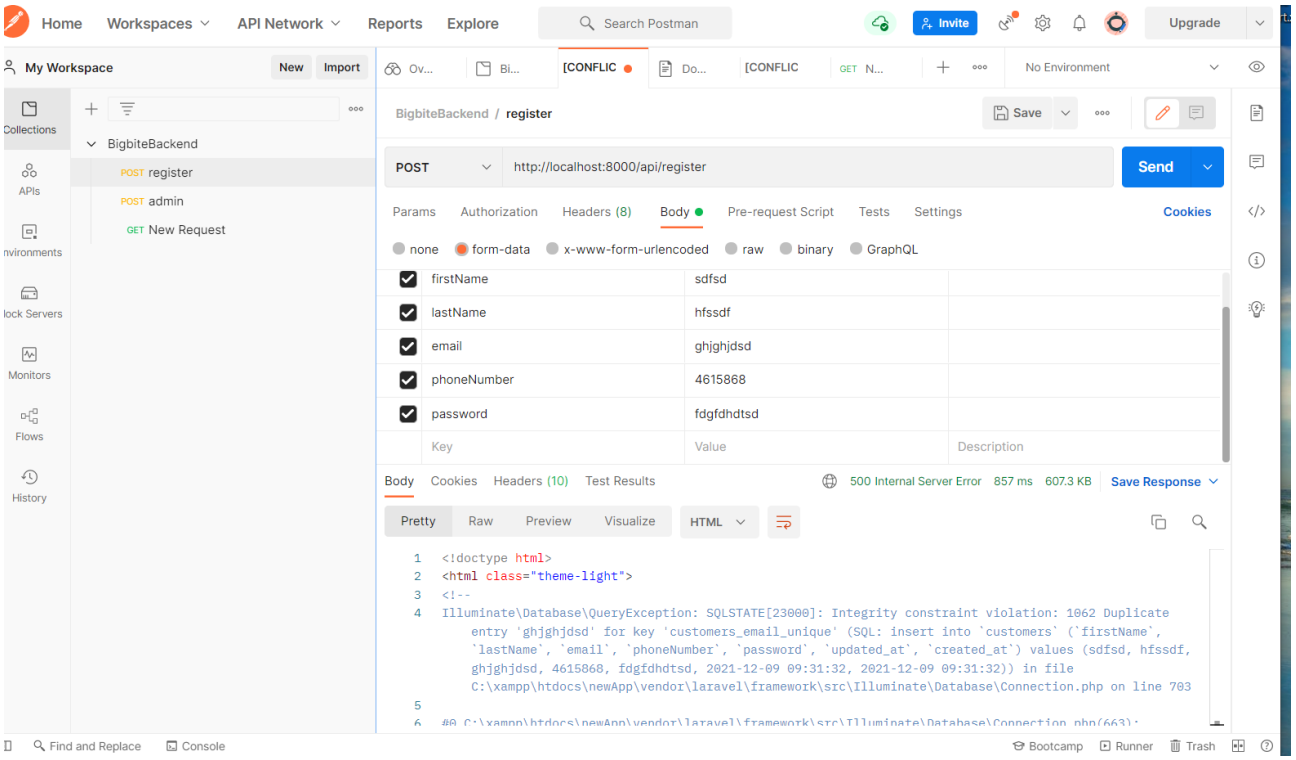


Abb. 15: Duplicating Test durch Postman

Test Name	Duplicate Customer Register Test
Testszenario	**Eine Post-Anfrage zum Routenpfad „/register“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**ein Benutzer mit dem folgenden E-Mail schon in der Datenbank vorhanden ist: email@gmail.com**
Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Postman wird benötigt, um HTTP-Anfragen zu erstellen sowie deren Antworten zu lesen**
Testablauf	**Benutzer registriert sich mit seinem Email und Passwort**

Nachbedingung	**Server hat mit „false“ beantwortet.**
Testergebnis	**Erfolgreich**

4.3 Fake User Login Test:

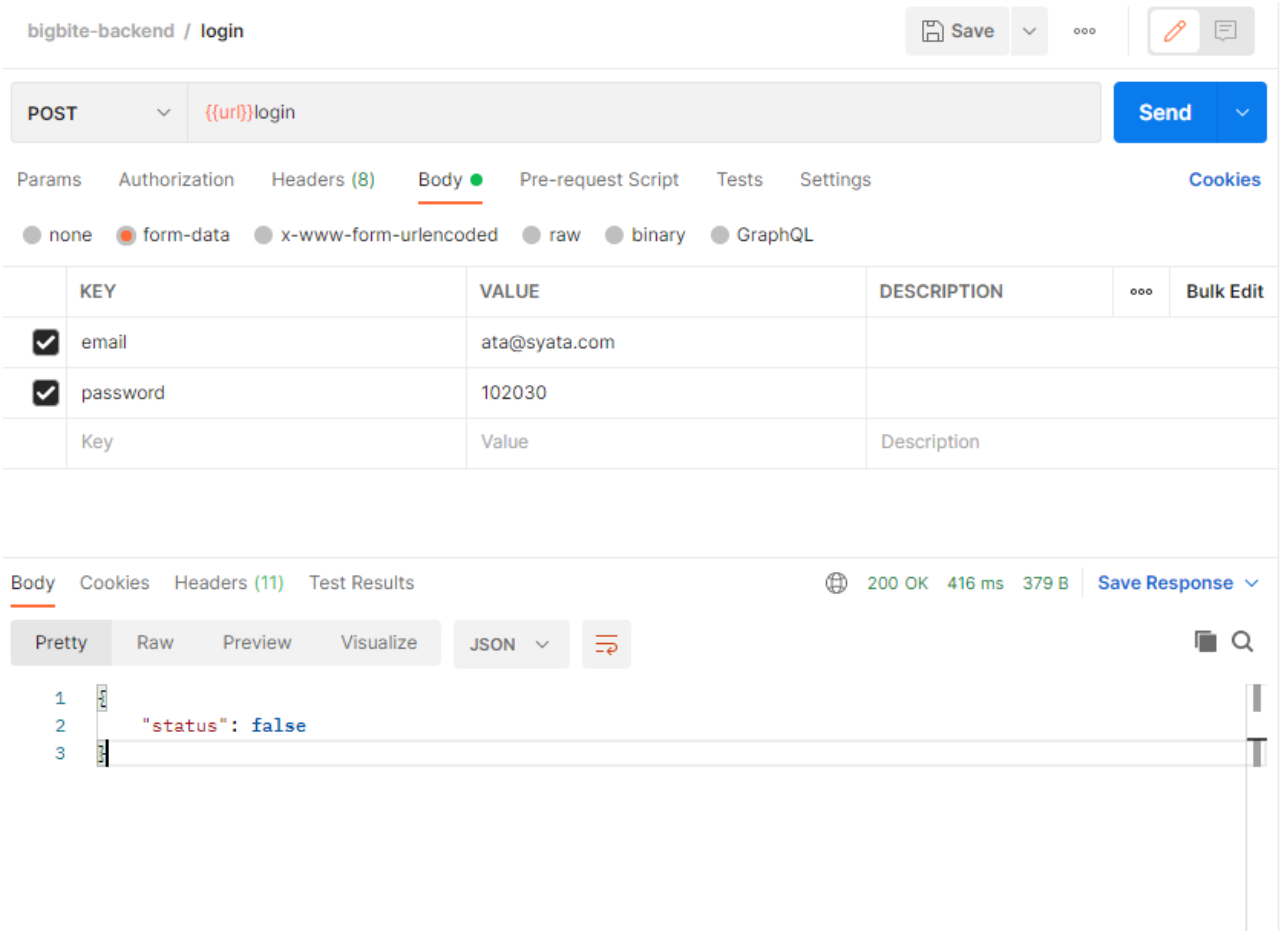


Abb. 16: Fake User Test durch Postman

Test Name	Fake User Login Test
Testszenario	**Eine Post-Anfrage zum Routenpfad „/login“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**Ein Benutzer mit den folgenden Attributen muss nicht schon in der Datenbank vorhanden und falsche Informationen damit wir sehen kann, wenn es falsche Info geht, gibt Testergebnis “false”. Falsche Information: E-Mail-Adresse: ata@syata.com**

	Kennwort: „102030“
Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Postman wird benötigt, um HTTP-Anfragen zu erstellen sowie deren Antworten zu lesen**
Testablauf	**Benutzer meldet seine Email und Password**
Nachbedingung	**Server hat mit „false“ beantwortet.**
Testergebnis	**Erfolgreich**

4.4 User Login Test:

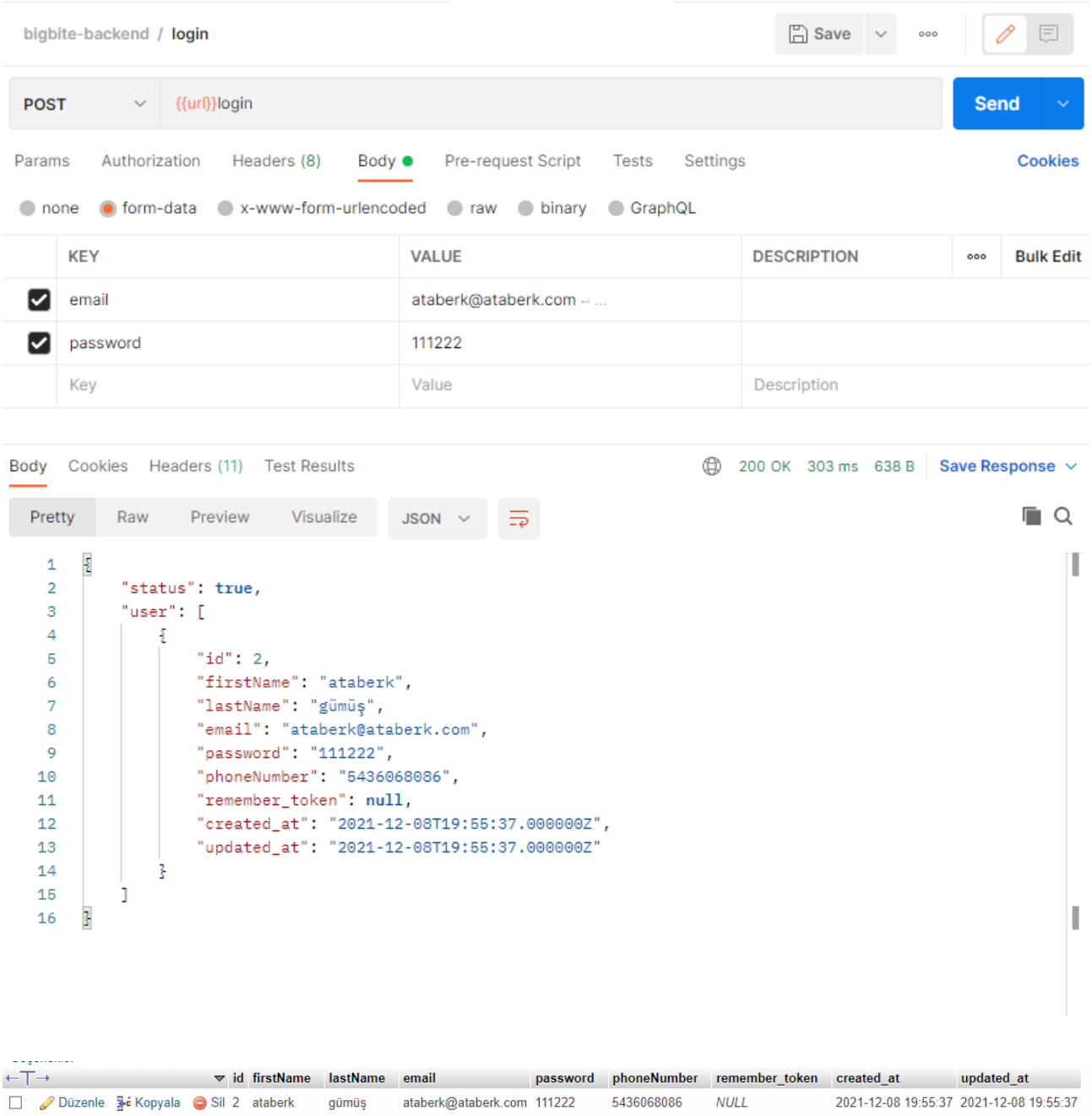


Abb. 17: User Login Test durch Postman

Test Name	User Login Test
Testszenario	**Eine Post-Anfrage zum Routenpfad „/login“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**Ein Benutzer mit den folgenden Attributen muss schon in der Datenbank vorhanden sein:**

	E-Mail-Adresse: ataberk@ataberk.com Kennwort: „111222“
Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Postman wird benötigt, um HTTP-Anfragen zu erstellen sowie deren Antworten zu lesen**
Testablauf	**Benutzer meldet seine Email und Password**
Nachbedingung	**Server hat mit „true“ beantwortet.**
Testergebnis	**Erfolgreich**

4.5 Food Delete Test

Test Name	Food delete test
Testszenario	**Eine Post-Anfrage zum Routenpfad „/deletefood“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**Admin muss mit seinem eigenen Mail und Passwort schon angemeldet sein**
Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Nur ein Browser benötigt**
Testablauf	**Admin klickt “Delete” Button**
Nachbedingung	**Food wird nicht mehr angezeigt**
Testergebnis	**Erfolgreich**

4.6 Food hinzufügen Test

Test Name	Food hinzufügen
Testszenario	**Eine Post-Anfrage zum Routenpfad „/uploadfood“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**Admin muss mit seinem eigenen Mail und Passwort schon angemeldet sein**

Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Nur ein Browser benötigt**
Testablauf	**Admin muss notwendige Textareas erfüllen dann “Save” Button klicken**
Nachbedingung	**Neu hinzugefügtes Food wird gelistet**
Testergebnis	**Erfolgreich**

4.7 Category update Test

Test Name	Category update Test
Testszenario	**Eine Post-Anfrage zum Routenpfad „/categories.update“ wird erstellt, um die entsprechende API-Antwort zu überprüfen**
Vorbedingungen	**Admin muss mit seinem eigenen Mail und Passwort schon angemeldet sein**
Konfiguration des Tests	**1. Initialisierung der Server-Applikation 2. Nur ein Browser benötigt**
Testablauf	**Admin soll die bearbeitete Category speichern.**
Nachbedingung	**Categories werden wiederherstellt**
Testergebnis	**Erfolgreich**

Die oberste Priorität ist es, die definierten Use Cases des Produkts zu validieren. Hier soll es beschrieben werden wie das Produkt getestet wird.

[REQ4] Als Projektanforderung soll es für jeden Use Case mindestens einen Test Case definiert, implementiert und durchgeführt werden.