**Sagebiel, Hubertus**
Küsterstraße 4
31319 Sehnde

**Hartmann, Marc**
Kessiehausen 1
31848 Bad Münder

**Hochschule Weserbergland**

Studiengang: Bachelor of Science - Wirtschaftsinformatik
Studiengruppe: WI 61/19
Dozent: Dominik Einfalt

**Hausarbeit im Modul Anwendungsentwicklung V**
(07.02.2022 bis 01.04.2022)

**Thema: Programmierarbeit Fahrplanverwaltungssystem**

**Ausbildungsbetrieb:**
Finanz Informatik GmbH & Co. KG
Laatzener Straße 5
30539 Hannover

**Sperrvermerk**

Die vorliegende Arbeit beinhaltet in
Informationen über betriebsbezogene Prozesse sowie vertrauliche Daten der
Ausbildungsstätte. Sie darf nicht ohne ausdrückliche Genehmigung der

**Finanz Informatik GmbH & Co. KG**
**Laatzener Straße 5**

**30539 Hannover**

veröffentlicht oder Dritten zugänglich gemacht werden.

**Inhaltsverzeichnis**

## Abkürzungsverzeichnis

## Abbildungsverzeichnis

# 1 Einleitung

In der Hochschule Weserbergland soll im Rahmen des Moduls „Anwendungsentwicklung V“ sollen Studierenden verschiedene Methoden und Werkzeuge zur Erstellung von komplexen und verteilten Anwendungen kennenlernen. Dabei gewinnen Sie die Fähigkeit einer komponentenbasierten Erstellung von Software mit Hilfe standardisierter Softwarearchitekturen und Spezifikationen.
Leistungsnachweis dieses Moduls ist, die Erstellung einer Webanwendung unter der Berücksichtigung und wesentlicher Verwendung der in der Veranstaltung behandelten Inhalte. Ziel dieser Anwendung war 14 Benutzeranwendungsfälle umzusetzen.

# 2 Beschreibung der Anwendung

## 2.1 Allgemeine Informationen

Bei der Anwendung handelt es sich um ein Fahrplanverwaltungssystem, welches sowohl von Kunden als auch von Mitarbeitern genutzt werden kann. Die Anwendung volviert sich grundsätzlich um 3 Objektarten. Diese sind die Haltestellen, die Buslinien und die Fahrpläne.
Aus Nutzersicht gibt es eine strenge Unterteilung zwischen Mitarbeitern und Kunden, welche sich in Form von zwei komplett getrennten Seiten wiederfindet. Diese Seiten sind lediglich über einen Knopf zum Wechseln auf die jeweils andere Seite Verbunden. Die beiden Seiten haben abgesehen von der Wechselfunktion noch die Gemeinsamkeit, dass beide die Möglichkeit bieten alle Buslinien, Haltestellen und Fahrpläne in einer Tabelle anzeigen zu lassen, um einen Überblick über die vorhandenen Daten zu ermöglichen.
Es ist mit der Anwendung möglich sich aus Kundensicht Informationen, wie Fahrpläne, Buslinien an Haltestellen und Haltestellen an Buslinien anzeigen zu lassen. Mitarbeiter hingegen haben umfangreiche Möglichkeiten die Daten des Systems zu verändern, wie zum Beispiel das Löschen, Anlegen und Ändern einzelner Objekte.
Die Oberfläche wurde nur auf 2 Seiten unterteilt, da dies die Komplexität für den Anwender reduziert und die jeweiligen Rollen alles was sie benötigen sofort sehen können. Des Weiteren wurde die Oberfläche mit Material UI Komponenten

und einem individuellen Tabellendesign entworfen, um eine angemessene Größe und Formatierung der Dargestellten Komponenten zu gewährleisten.

## 2.2 Entitäten

```
 9
10  @Entity
11  @Table(name = "T_BUSLINIE")
12  public class Buslinie {
13
14      @Id
15      @Column(name = "BUSLINIE_ID")
16      @GeneratedValue(strategy = GenerationType.IDENTITY)
17      private Long buslinie_id;
18
19
20      @Column(name = "BUSLINIE_NAME")
21      private String buslinie_name;
22
```

Abbildung 11: Entitäten

Das in dieser Arbeit beschriebene Datenmodell wurde mithilfe von Entitätsklassen abgebildet. Die Persistenz dessen, wird durch den Einsatz von Spring Data sichergestellt.[1] Die obenstehende Abbildung visualisiert, wie die Entität Buslinie implementiert wurde. Im Folgenden wird diese Implementierung beispielhaft erläutert, da die Erläuterung aller Entitäten, den Rahmen dieser Arbeit sprengen würde.

Die Klasse Buslinie wurde mit @Entity annotiert, dies bewirkt, dass diese durch die JPA als Tabelle abgebildet wird. Mittels der Annotation @Table wird der Tabellenname angegeben.

Damit das Attribut buslinie_id durch die JPA als ID erkannt wird, wurde es mit der Annotation @Id versehen. Außerdem wurde das Attribut buslinie_id mit der Annotation @GeneratedValue versehen. Die Annotation @GeneratedValue bewirkt, dass die ID automatisch generiert wird. Die übrigen Attribute werden durch Spring automatisch als Spalten ab, mithilfe der Annotation @Column können weitere Eigenschaften zu den Attributen angegeben werden.

[1] Vgl. Hierzu und im Folgenden Spring (2022b), o.S.

## 2.3 Persistenz der Entitäten

Damit die Persistierung der Entitäten gesichert werden kann, wird für jede Entität ein eigenes JpaRepository-Interface implementiert. Das genannte Interface kann im Anschluss um diverse Funktionalitäten erweitert werden. Im Folgenden wird beispielhaft das Interface HaltestellenRepository erläutert.

```
10
11  @Repository
12  public interface HaltestellenRepository extends JpaRepository<Haltestelle, Long> {
13
14      @Query(value ="SELECT HALTESTELLE_ID, HALTESTELLE_NAME FROM"+
15      "T_Haltestelle WHERE HALTESTELLE_NAME=?1", nativeQuery = true)
16      Optional<Haltestelle> findHaltestelleByName(String name);
17
18      @Query(value ="SELECT HALTESTELLE_ID FROM t_HALTESTELLE as h"+
19      "WHERE h.HALTESTELLE_NAME=?1", nativeQuery = true)
20      String getIdForName(String haltestelleName);
21
22  }
23
```

Abbildung 22: Darstellung des HaltestellenRepository

Das verwendete JpaRepository-Interface bietet verschiedene Funktionalitäten an, da die Nennung aller den Rahmen dieser Arbeit sprengen würde, wird im Folgenden auf verwendete Funktionalitäten Bezug genommen.

Mit der Annotation @Query lassen sich personalisierte System Query Language(SQL)-Abfragen durchführen. In dem gegebenen Beispiel wurde noch der Flag nativeQuery = true hinzugefügt. Dieser sagt aus, dass native SQL-Queries erwartet werden, ansonsten würde der Input JPQL-Queries erwarten, welche eine andere Syntax haben.

Bei SQL-Anfragen, wie z.B der Methode findHaltestelleByName(), welche als Ergebnis die dazugehörige Entität zurückliefert.

## 2.4 Rest-API

Die Controller Klasse in Spring Boot bietet REST-Schnittstellen an, welche den externen Zugriff auf interne Ressourcen ermöglicht.[2] In diesem Kapitel wird ein Ausschnitt des Quellcodes gezeigt, um dabei die Umsetzung der Benutzeranwendungsfälle beispielhaft zu erläutern.

[2] Vgl. Hierzu und im Folgenden (Spring 2022b), o:.S

```java
@RestController
@RequestMapping(path = "bussupervisor")
@CrossOrigin(origins = "*")
public class BussupervisorController {

    private final BussupervisorService bussupervisorService;

    @Autowired // Dependency injection
    public BussupervisorController(BussupervisorService bussupervisorService) {
        this.bussupervisorService = bussupervisorService;
    }

    @GetMapping(value = "/getBuslinieWhoVisitsHaltestelle/{haltestelleName}")
    public ArrayList<Buslinie> getBuslinieWhoVisitsHaltestelle
    (@PathVariable("haltestelleName") String haltestelleName) {
        System.out.println(haltestelleName);
        return bussupervisorService.getBuslinieFromHaltestelle(haltestelleName);
    }

    @PostMapping(path ="/addHaltestelle")
    public void addHaltestelle(@RequestBody Haltestelle haltestelle) {
    bussupervisorService.addNewHaltestelle(haltestelle);
    }
```

Abbildung 33: BussupervisorController.java

Durch die Annotation @Autowired, wird die in Kapitel 4.1 beschriebene Dependency Injection erreicht.

Die Annotation @RestController übermittelt Spring, dass die Klasse BussupervisorController bei der Bearbeitung von Web-Anfragen berücksichtigt werden soll.

Die Annotation @RequestMapping(path =“bussupervisor“) ist für das Routing zuständig, sie ermöglicht es HTTP-Anfragen über den angegebenen Pfad, zu einer der Methoden abzubilden.

Im Moment unterstützt Spring fünf Annotationen für die Handhabung von eingehenden HTTP-Anfragen.[3] Darunterfallen: @GetMapping, @PostMapping, @PutMapping, @DeleteMapping und @PatchMapping. Wie die Namensgebung schon verrät, kümmert sich jedes Mapping und eine Form von Anfrage

[3] Vgl. Baeldung (2020), o.S.

## 2.5 Konfigurationen

bussupervisor > src > main > resources > application.properties

```
#configuration
spring.datasource.initialization-mode=always
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/bussupervisordb
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/api.html
spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

Abbildung 4: application.properties

In der Datei application.properties werden sämtliche Konfigurationseinstellungen durchgeführt.[4] In dieser Arbeit wurde sich dafür entschieden, sämtliche SQL-Statements zu loggen, dies diente vor allem Debugging-Zwecken. Außerdem werden in dieser Datei sämtliche Datenbankeinstellungen vorgenommen. Darunter zählen die Credentials der Datenbank und der Initialisierungsmodus. Bei dem Initialisierungsmodus handelt es sich darum, in welchem Stil die Datenbank, mit den Dateien data.sql und schema.sql initialisiert wird

# 3 Darstellung und Motivation der verwendeten technischen Architektur

## 3.1 Client Server Architektur

Bei der Implementierung einer funktionsfähigen Webanwendung muss zwanghaft die Client-Server Architektur bedacht werden, da wir uns in einem Netzwerk befinden, welches einen Server, sowie Client mit Informationen versorgen möchte.[5]

Da dieses Architekturmuster einen Teil des Leistungsnachweises darstellt, wird es im Folgenden kurz näher erläutert.

[4] Vgl. Hierzu und im Folgenden Spring (2022a), o.S.

[5] Vgl. Hierzu und im Folgenden von Thienen (2013) S.15ff.

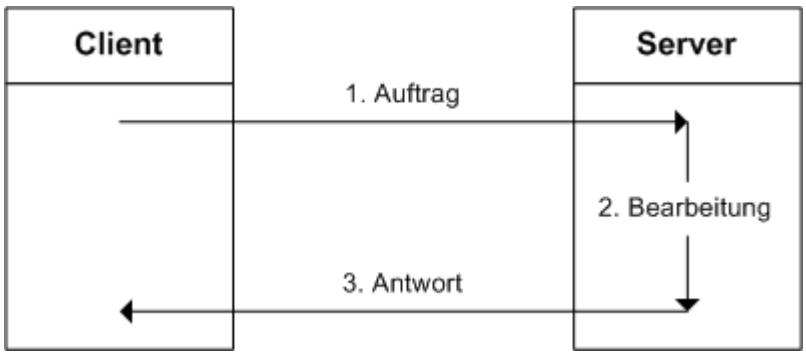


Abbildung 55: Vereinfachte Darstellung der Client-Server-Architektur[6]

Wie die obige Abbildung aufzeigt, handelt es sich bei der Client-Server-Architektur primär um zwei Akteure, dem Client und dem Server. Bei dem Client handelt es sich prinzipiell um den Browser, der Server wiederrum wird durch das Spring-Boot-Backend dargestellt.

In dieser Wechselwirkung gibt es drei Aktionen, den Auftrag, die Bearbeitung und die Antwort.

Im ersten Schritt wird dem Server, durch den Client, ein Auftrag zur Verarbeitung erteilt. Daraufhin nimmt der Server diesen entgegen und verarbeitet ihn. Zuletzt wird nach der Bearbeitung, durch den Server, eine Antwort erstellt und dem Client überliefert

# 4 Theoretische Erläuterung der verwendeten Technologien und API

## 4.1 Spring Boot

Spring Boot basiert vollständig auf dem Spring-Framework. Ziel bei der Entstehung, war es die Entwicklung von eigenständigen, lauffähigen Anwendungen ernorm zu erleichtern.[7] Spring Boot arbeitet unter dem Motto: „Covention over configuration“. Neben dem Motto sollen vereinzelte Annotation ausreichen, um extern konfigurierbare Artefakte mit allen dazugehörigen Abhängigkeiten zu erzeugen.

Neben der erleichterten Entwicklung ermöglicht Spring es, die technischen Schichten von einer Anwendung, also Persistenz und Webschicht, getrennt voneinander zu testen, was einen wesentlichen Bestandteil zu sauberem Code beiträgt.

---

[6] Vgl. Hierzu und im Folgenden Fettke (2016), o.S.

[7] Vgl. Hierzu und im Folgenden Reddy (2017) S.1-5.

Kernaspekt von Spring Boot ist die sogenannte Dependency Injection. Bei einer Dependency Injection handelt es sich um ein Entwurfsmuster der Programmierung, welche es ermöglicht Komponenten zur Laufzeit bereitzustellen. Spring erreicht dies, indem sie die Verwaltung der Abhängigkeiten über eine einfache Konfiguration anbietet.

Um den Lesefluss dieser Arbeit zu verbessern werden weitere theoretische Grundlagen in dem Kapitel 5 erläutert. Dieses Vorgehen soll ein verbessertes Verständnis, aufgrund der Verbindung von Theorie und Praxis, sicherstellen.

## 4.2 React

React wurde ursprünglich von einem Softwareingenieur bei Facebook im Jahr 2011 entwickelt. Ab dem Jahr 2013 stellte Facebook es als Open-Source-Projekt zur Verfügung, seit 2017 und Version 16.0.0. ist es unter einer MIT-Lizenz veröffentlicht.

React ist eine auf JavaScript basierende Softwarebibliothek, welche eine grundlegende Struktur für die Ausgabe von User-Interface-Komponenten von Webseiten bereitstellt.[8] In React können Komponenten als selbst definierte JSX-Tags repräsentiert werden, sie werden außerdem hierarchisch aufgebaut.[9]

React nutzt ein direktionierbares Datenfluss Konzept, welches dabei unterstützen soll, auch komplexe Anwendungen einfach und trotzdem performant aufzubauen.[10]

.

[8] Vgl. Meta Platforms, Inc. (2022a), o.S,

[9] Vgl. Meta Platforms, Inc. (2022b), o.S

[10] Vgl. Mateusz Grzesiukiewicz (2018), S.107-108

# 5 Darstellung des Datenmodells

## 5.1 Datenbankmodell

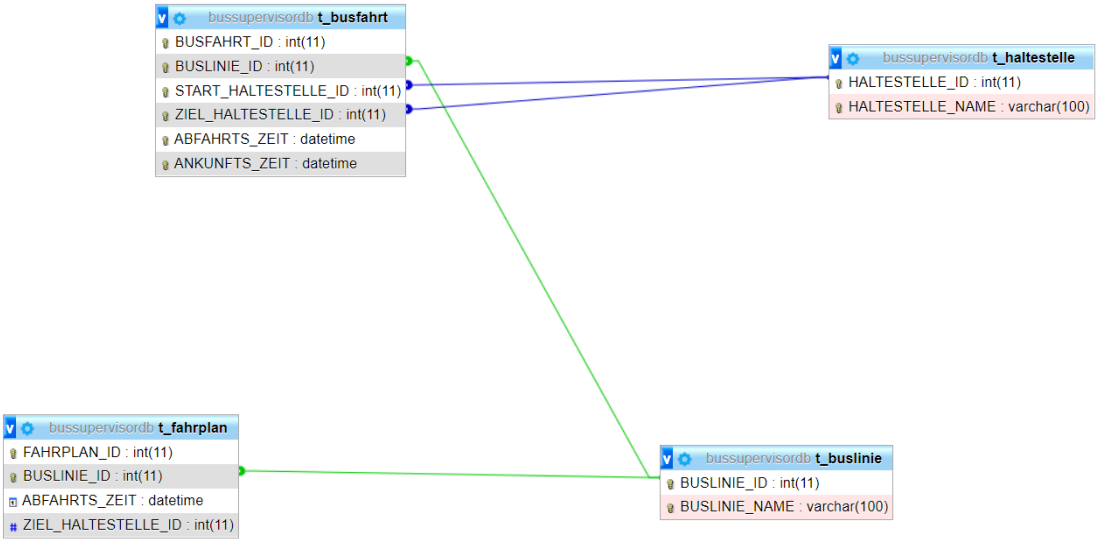


Abbildung 66: Eigene Darstellung

Wie in der obigen Abbildung zu erkennen ist wurde sich für vier Tabellen entschieden. Hierbei haben wir die DTO´s Buslinie, Haltestelle und Fahrplan. Um die dritte Normalform einhalten zu können, wurde sich für die Tabelle T_Buslinie entschieden. Diese löst die ansonsten, entstehende, n-m Beziehung, zwischen T_Haltestelle und T_BUSLINIE auf und verwandelt sie in eine 1-N-Beziehung.

## 5.2 Klassendiagramm

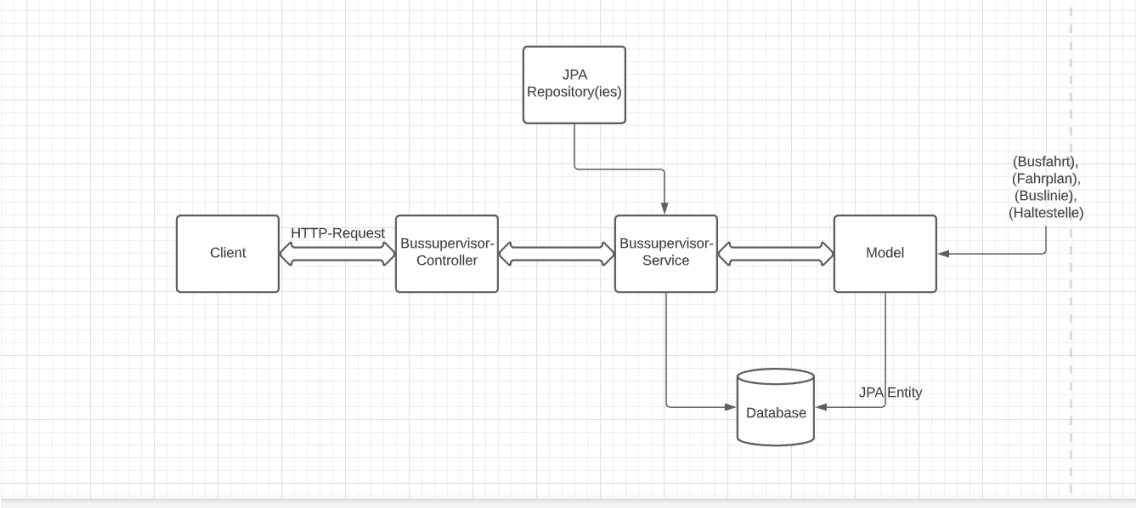


Abbildung 77: Klassendiagramm

Der Programmablauf startet damit, dass der Anwender durch das Klicken eines Buttons oder ähnliches, eine HTTP-Anfrage gegen den Rest-Controller sendet. Der Controller verarbeitet die Anfrage und ruft den bestimmten Anwendungsfall durch den BussupervisorService auf. Der BussupervisorService ruft das, zu der Enitity passende, JpaRepository auf. Je nach Anwendungsfall können mithilfe der entwickelten Anwendung Daten aus der Datenbank ausgelesen, bearbeitet, gelöscht oder neu angelegt werden.

## 6 Bedienungsanleitung der Weboberfläche

Die Weboberfläche basiert auf zwei Ansichten, der Kundenansicht und der Mitarbeiteransicht. Zwischen diesen kann gewechselt werden, in dem der Knopf, mit dem Namen der jeweils anderen Ansicht, oben rechts in der Ecke gedrückt wird.

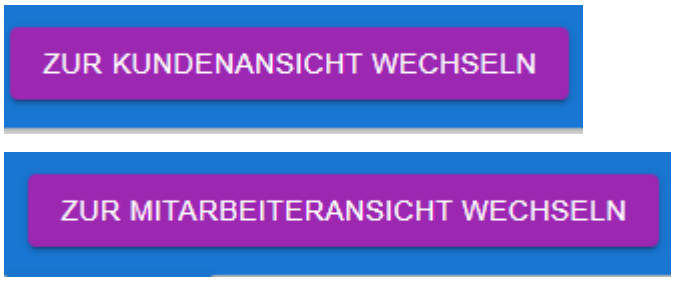


Abbildung 88: Knöpfe zum Ansichtswechsel

Unter der Kopfzeile der Anwendung befindet sich in der Mitarbeiteransicht eine Textanzeige, die dem Mitarbeiter anzeigt, ob die zuletzt ausgeführte Operation erfolgreich war, um dem Nutzer ein schnelles Feedback zur Verfügung zu stellen.

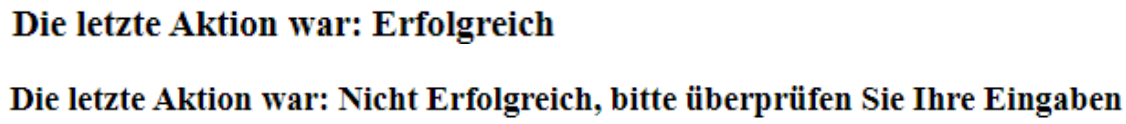


Abbildung 9: Feedbackanzeige

In der Kundensicht unter der Kopfzeile und in der Mitarbeiteransicht unter der Feedbackanzeige befinden sich Felder und Knöpfe, mit welchen die verschiedenen Funktionalitäten der Anwendung ausgeführt werden können. Bei der Kundenansicht umfasst dies die Möglichkeiten Buslinien an einer Haltestelle, Haltestellen einer Buslinie und den Fahrplan an einer Haltestelle, ab einer gewissen Uhrzeit abzufragen.


Abbildung 910: Eingabefelder und Knöpfe

Die Felder sind außerdem mit Platzhaltern und sprechenden Erklärungen versehen, um dem Nutzer besser darzustellen, was für eine Eingabe erwartet wird.

Ihre Funktionalitäten können genutzt werden, in dem die Felder mit den Platzhaltern gefüllt werden und danach auf den entsprechenden Knopf der Funktionalität gedrückt wird. Die Knöpfe sind durch ihr blaues Design klar vom Rest der Anwendung abgehoben.
Wenn Funktionalitäten zum Anzeigen von Werten genutzt werden, werden diese in Tabellenform unten auf der Seite angezeigt.

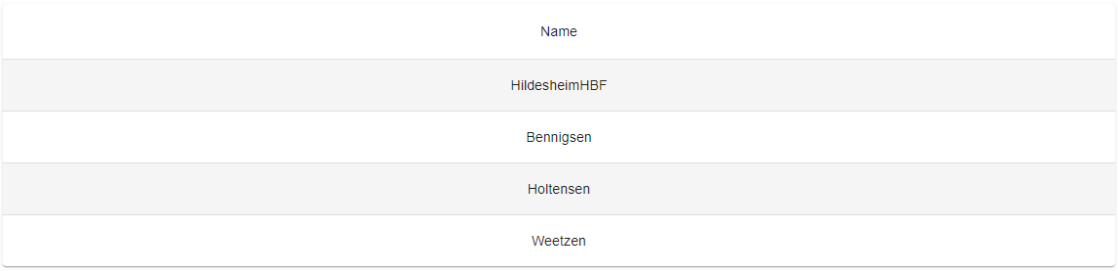


Abbildung 1011: Tabellarische Darstellung in der Kundenansicht

Zum Anzeigen aller Buslinien und Haltestelle sind sowohl in der Kunden- als auch in der Mitarbeiteransicht Knöpfe vorhanden, welche die gewünschten Werte in tabellarischer Form unten auf der Seite ausgeben. Die Seite der Mitarbeiter bietet außerdem die Möglichkeit alle FahrplanIds anzuzeigen. Es ist jedoch aus Überschaubarkeitsgründen nicht möglich mehr als eine Tabelle gleichzeitig darzustellen.

## 7 Installationshinweise

Bei der Installation der Anwendung ist zunächst die Bussupervisor.zip Datei in einem beliebigen Ordner zu entpacken. Ist die Datei entpackt sollte im Entpackten Ordner in der höchsten Strukturebene Npm install in CMD durchgeführt werden, ist dies fertig sollte mvn clean install ebenfalls in CMD ausgeführt werden.
Ist dies geschehen, sollte vor Ausführen der Anwendung MySQL und Apache gestartet werden. In PhPMyAdmin muss dann eine Datenbank mit dem Namen „bussupervisordb“ erstellt werden. Alternativ kann auch ein beliebiger Name gewählt werden, der jedoch in der Datei application.properties in Zeile 4 eingetragen werden.
Danach ist die Anwendung startbereit.
Zum Starten der Anwendung muss die starting-script Datei, welche in der höchsten Strukturebene des Anwendungsordners liegt, ausgeführt werden. Sie sollte das Backend starten und die Weboberfläche der Anwendung öffnen.

## Literatur


Baeldung (2020):
Spring @RequestMapping New Shortcut Annotations, https://www.baeldung.com/spring-new-requestmapping-shortcuts

Stand: 01.04.2022

Fettke, Peter. (2016):

Client-Server-Architektur,

Stand: 01.04.2022

Grzesiukiewicz, Mateusz. (2018):
Hands-On Design Patterns with React Native 1.Aufl., Packt Publishing

K, Siva Prasad Reddy. (2017):
Beginning Spring Boot 2: Applications und Microservices with the Spring Framework
1.Aufl., Apress

Meta Platforms, Inc. (2022a):

React A JavaScript library for building user interfaces, https://reactjs.org/,

Stand: 01.04.2022

Meta Platforms, Inc. (2022b):

Composition vs Inheritance, https://reactjs.org/docs/composition-vs-inheritance.html,

Stand: 01.04.2022

Spring (2022a):

Common Application Properties, https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html

Stand: 01.04.2022

Spring (2022b):

Spring Boot Reference Documentation, https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

Stand: 01.04.2022

Wolfhard, von Thienen. (2013):
Client/Server Technologie und Realisierung im Unternehmen
1.Aufl., Vieweg Verlag

## Anhangsverzeichnis

# VI Anhang