Ein Bild, das Text, Schrift, Logo, Grafiken enthält.

Automatisch generierte Beschreibung

Hausarbeit

Internationale Hochschule Duales Studium

Studiengang: Informatik

**Dokumentation der BUFIVAS Software**

Marc-Oliver Miekautsch

Matrikelnummer: 102206694

Winterbergstraße 56, 01237 Dresden

Betreuende Person: Henrik Paul

Abgabedatum: 31.09.2025

Inhaltsverzeichnis

[Abbildungsverzeichnis III](#_Toc194338864)

[1 Anwendungsbeschreibung 1](#_Toc194338865)

[1.1 JSF 1](#_Toc194338866)

[1.2 Java Beans/Klassen 3](#_Toc194338867)

[2 Motivation der verwendeten technischen Mittel 4](#_Toc194338868)

[2.1 Verwendete Tools 4](#_Toc194338869)

[2.2 Archtekturentscheidungen 4](#_Toc194338870)

[3 User Manual 5](#_Toc194338871)

[3.1 Flüge buchen 6](#_Toc194338872)

[3.2 Flüge erstellen 8](#_Toc194338873)

[3.3 Flüge löschen 8](#_Toc194338874)

[4 Installationshinweise 9](#_Toc194338875)

# Abbildungsverzeichnis

[Abbildung 1: Der Home Screen 5](#_Toc194338876)

[Abbildung 2: Booking-Portal, Detaillierte Flugansicht 6](#_Toc194338877)

[Abbildung 3: Booking-Portal, Flugübersicht 6](#_Toc194338878)

[Abbildung 4: Booking-Portal, Abschließende Prüfung 7](#_Toc194338879)

[Abbildung 5: Booking-Portal, Passagierdateneingabe 7](#_Toc194338880)

[Abbildung 6: FlightCreator 8](#_Toc194338881)

[Abbildung 7: Anmeldestatus 8](#_Toc194338882)

[Abbildung 8: Flugliste als Manager 9](#_Toc194338883)

# Anwendungsbeschreibung

Die Anwendung beschreibt ein Busfahrplaninformationsundverwaltungsanwendungssystem (BUFIVAS) und umfasst folgende Funktionen:

* Erstellen von Buslinien & Bearbeiten
* Erstellen von Haltestellen
* Löschen von angelegten Buslinien (wenn als Manager angemeldet)
* Planen von Busfahrten

Sie basiert auf JSF, wobei eine Website modular aus vielen Teilstücken zusammengesetzt ist, die beliebig getauscht werden können. Die Anwendung bietet 6 verschiedene Webpages, die sich aus insgesamt 15 verschiedenen Modulen zusammensetzen. Dazu kommen noch 9 Java Klassen, die für die Business- & Anzeigenlogik, sowie der Datenspeicherung nach dem JEE-Architekturmodell zuständig sind. Diese Teilstücke sollen nun nacheinander erklärt werden.

## JSF

Das allgemeine Layout setzt sich aus 4 Dateien zusammen; dem Header, Footer, der Styles CSS und dem Template, was diese 3 verbindet:

**template.xhtml** dient als zentrales Layout-Template der Anwendung und definiert den strukturellen Rahmen aller Seiten der Anwendung. Sie nutzt das JSF-Framework in Kombination mit Facelets, um eine klare Trennung von Layout und Inhalt zu gewährleisten. Dabei wird <ui:insert> für dynamisch einzufügende Seitentitel und Seiteninhalte sowie Includes für wiederverwendbare Komponenten wie Kopf- und Fußzeile genutzt, welche zu dem dynamisch ausgestauscht werden können. Außerdem werden ein PrimeFaces-Theme sowie eine benutzerdefinierte CSS-Datei eingebunden, um ein einheitliches visuelles Erscheinungsbild zu gewährleisten.

**header.xhtml** definiert den oberen Seitenbereich (Header) der Anwendung und wird in das Layout-Template template.xhtml eingebunden. Sie besteht aus einem dreispaltigen Gitter <h:panelGrid>, das den Titel der Anwendung ("BUFIVAS"), einen Link zur Startseite sowie einen Anzeigebereich für den aktuellen Anmeldestatus enthält. Abhängig davon, ob ein Nutzer eingeloggt ist #{user.loggedIn}, wird entweder der Text „Angemeldet als Manager“ oder „Nicht angemeldet“ angezeigt.

**footer.xhtml** definiert den Fußbereich der Anwendung und enthält lediglich den einen h:outputText mit dem Namen des Entwicklers, sowie Debug-Informationen, wenn Fehler auftreten.

**home.xhtml** stellt die Startseite der Anwendung dar und basiert auf dem zuvor definierten Layout in template.xhtml. Sie bindet sich über das <ui:composition>-Tag an das Template ein und befüllt den dort vorgesehenen content-Bereich. Innerhalb eines Formulars werden mehrere Schaltflächen h:commandButton bereitgestellt, die die unterschiedlichen Funktionen der Anwendung aufrufen: das Erstellen und Bearbeiten von Buslinien und Haltestellen, den Fahrtenplaner, sowie Login- und Logout-Optionen. Die Anzeige der Login- bzw. Logout-Schaltfläche erfolgt dabei dynamisch abhängig vom Anmeldestatus des Nutzers über die Bedingung #{user.loggedIn}.

**buslinecreator.xhtml** stellt eine Eingabemaske zur Erstellung neuer Buslinien innerhalb der Anwendung bereit. Der Seiteninhalt wird im Bereich content definiert und enthält ein Formular mit mehreren Eingabeelementen. Zunächst wird in einem zweispaltigen Gitter <h:panelGrid> die Liniennummer als Pflichtfeld abgefragt. Anschließend ermöglicht ein weiteres PanelGrid die Eingabe der Betriebszeiten (von/bis) im Format HH:mm, wobei ein f:convertDateTime-Tag zur Validierung und Umwandlung der Uhrzeitangaben verwendet wird. In einem weiteren Abschnitt kann über ein Dropdown-Menü die Starthaltestelle aus der bestehenden Haltestellenliste ausgewählt werden. Die Erstellung der Linie wird über einen Schaltknopf Linie erstellen ausgelöst. Nach erfolgreicher Anlage der Linie erscheint eine Bestätigungsmeldung, sofern es sich nicht um den ersten Seitenaufruf handelt.

**busstopcreator.xhtml** dient der Erweiterung des Haltestellenpools. Benutzer können hier neue Haltestellen mit einem eindeutigen Namen sowie den zugehörigen Standort anlegen. Die Eingaben erfolgen in einem zweispaltigen Formularraster, das eine übersichtliche Zuordnung von Beschriftung und Eingabefeld ermöglicht. Nach dem Absenden über den Button Haltestelle erstellen wird die neue Haltestelle in das System übernommen. Um dem Anwender Rückmeldung zu geben, blendet die Seite eine Bestätigungsmeldung ein, sobald die Anlage erfolgreich war.

Die Seite **buslineeditor.xhtml** dient der Bearbeitung von bestehenden Buslinien. Die Seite an sich ist lediglich ein Container, welcher abhängig vom internen Status des Controllers unterschiedliche Teilansichten einblendet. Im Ausgangszustand wird zunächst eine Auswahldarstellung eingebunden, über die der Benutzer eine Linie auswählen kann (buslineeditorselect.xhtml). Sobald eine Linie gewählt ist, wechselt die Seite in den Bearbeitungsmodus und zeigt die Bearbeitungsansicht buslineeditordata.xhtml, in der die Daten der Linie gepflegt werden können

**buslineeditorselect.xhtml** ist eine einfache Auswahldarstellung und enthält lediglich ein <h:selectOneMenu> zur Auswahl der zu editierende Linie, sowie einen Button zum Bestätigen.

Der **buslineeditordata.xhtml** beinhaltet die Detailansicht des Linieneditors und zeigt die aktuell gewählte Buslinie mit ihren zugeordneten Haltestellen. In einer Tabelle werden die Haltestellen mit Namen und Standort aufgelistet, in der Reihenfolge, in der sie von der Linie angefahren werden. Zur Verdeutlichung werden in der ersten Spalte Grafiken angezeigt, abhängig davon, ob es sich um einen Start-, End- oder Zwischenstopp handelt. Manager-Benutzer haben zusätzlich die Möglichkeit, einzelne Haltestellen über eine Schaltfläche aus der Linie zu entfernen. Unterhalb der Tabelle befindet sich ein Formularbereich, mit dem weitere Haltestellen aus dem allgemeinen Haltestellenpool hinzugefügt werden können. Dabei wird neben der Auswahl der Haltestelle auch die Fahrzeit zum vorherigen Halt erfasst, eingeschränkt auf Werte zwischen 1 und 60 Minuten. Ein Button zum Zurücknavigieren erlaubt es, jederzeit in die Auswahlansicht zurückzukehren.

**planner.xhtml** ist die zweite Container-Seite und besteht aus zwei Komponenten: Im linken Bereich wird über plannerNav.xhtml eine Navigationsleiste eingebunden, über die der Benutzer durch die einzelnen Schritte wechseln kann. Der rechte Inhaltsbereich wird dynamisch abhängig vom Status des Controllers befüllt: im Ausgangszustand (Status 0) mit einer Übersicht der vorhandenen Linien (buslinedisplay.xhtml), anschließend (Status 1) mit der Eingabe und Anpassung eines konkreten Fahrplans (buslineschedule.xhtml) und schließlich (Status 2) mit der Ergebnisdarstellung der berechneten Fahrt (plannerResult.xhtml).

**buslinedisplay.xhtml** zeigt eine tabellarische Übersicht aller im System vorhandenen Buslinien. Jede Zeile der Tabelle enthält die Liniennummer sowie Aktionsschaltflächen. Über Auswählen kann eine Linie für die weitere Planung bestimmt werden, wodurch der Workflow im Fahrtenplaner fortgesetzt wird. Für Benutzer mit Manager-Rechten steht zusätzlich eine Löschen-Funktion zur Verfügung, mit der Linien aus dem System entfernt werden können.

**plannerNav.xhtml** stellt die Navigationsleiste für den Fahrtenplaner bereit. Sie enthält drei Schaltflächen, mit denen der Benutzer zwischen den verschiedenen Phasen der Planung wechseln kann: Linienübersicht, Fahrplan und Ergebnis. Die Buttons sind dabei dynamisch an den Status des Controllers gekoppelt. So stehen die Schaltflächen für Fahrplan und Ergebnis nur dann aktiv, wenn die vorherigen Schritte bereits durchlaufen wurden, andernfalls werden sie als deaktiviert dargestellt.

Die Seite **buslineschedule.xhtml** bildet den zweiten Schritt im Fahrtenplaner. Sie erlaubt es dem Benutzer, innerhalb der zuvor gewählten Linie einen Start- und einen Zielhalt festzulegen. Dafür stehen zwei <h:selectOneMenu> Felder zur Verfügung, die mit allen Haltestellen der Linie gefüllt sind und mit dem Bestätigungs-Button gespeichert werden können. Zusätzlich werden die Betriebszeiten der Linie angezeigt sowie eine tabellarische Übersicht der enthaltenen Haltestellen mit Namen, Standort und Fahrzeit zum nächsten Halt, sowie den Grafiken in der ersten Spalte, ähnlich wie bei buslineeditordata.xhtml. Für den Fall, das die Validierung der Eingaben fehlschlägt, enthält die Seite außerdem ein <h:outputScript> welches nur gerendert wird, wenn es etwas zum Anzeigen gibt.

Am Ende des Workflows steht die Seite **plannerResult.xhtml** und fasst die zuvor getroffene Auswahl zu einer vollständigen Fahrt zusammen. Sie zeigt an, in welcher Linie die Planung erfolgt ist, und berechnet die gesamte Fahrzeit für die gewählte Strecke. In einer Tabelle werden die tatsächlich genutzten Haltestellen in der korrekten Reihenfolge mit Symbolbild und Namen aufgeführt. Darüber hinaus gibt es noch den Button „Neue Fahrt planen“, womit der aktuelle Planungsprozess zurückgesetzt und eine neue Berechnung gestartet werden kann.

## Java Beans/Klassen

**User.java** verwaltet den Anmelde- und Rollenstatus innerhalb der Anwendung und ist als JSF-Managed Bean mit dem Namen "user" sowie einer anwendungsweiten Lebensdauer (@ApplicationScoped) definiert. Die Klasse speichert zwei Zustände: ob ein Benutzer angemeldet ist (isLoggedIn) und ob es sich um einen Manager handelt (isManager).

**BusStop.java** repräsentiert eine einzelne Haltestelle im System und verwaltet zwei zentrale Eigenschaften: den Namen der Haltestelle sowie deren Standort. Beide Werte werden über den Konstruktor beim Erzeugen eines Objekts gesetzt und können über Getter- und Setter-Methoden gelesen oder verändert werden

**BusLine.java** bildet das Datenmodell für eine Buslinie. Sie verwaltet neben der eindeutigen Linien-ID auch die Betriebszeiten der Linie (Start- und Endzeit) sowie die Abfolge der zugehörigen Haltestellen. Diese werden in einer Liste von BusStop-Objekten gespeichert, ergänzt durch eine parallele Liste von Fahrzeiten in Minuten, die den Abstand zwischen aufeinanderfolgenden Halten abbilden. Im Konstruktor wird direkt die Starthaltestelle der Linie festgelegt, um sicherzustellen, dass jede Linie mindestens eine Haltestelle enthält. Über Methoden können weitere Haltestellen mit Fahrzeit hinzugefügt oder bestehende entfernt werden. Zusätzlich ermöglicht die Klasse den Zugriff auf die Fahrzeit zu einem Halt über dessen Index oder direkt über das BusStop-Objekt.

**BusLineList.java** dient als zentrale Sammlung aller verfügbaren Buslinien innerhalb der Anwendung. Sie ist mit @ApplicationScoped deklariert und steht damit über die gesamte Laufzeit der Anwendung hinweg zur Verfügung. Intern verwaltet sie eine Liste von BusLine-Objekten, die beim Start mit Beispieldaten befüllt wird. Außerdem enthält sie Funktionen zum Abfragen der Liste, sowie zum Löschen und Hinzufügen neuer Linien.

**BusStopList,java** stellt eine zentrale Sammlung aller verfügbaren Haltestellen im System bereit. Die Klasse ist als @ApplicationScoped implementiert und damit während der gesamten Laufzeit der Anwendung gültig. Beim Start wird die Liste mit einem festen Satz an Beispieldaten initialisiert, der typische Haltestellen aus verschiedenen Stadtteilen umfasst.

Der **BusLineCreatorController.java** steuert die Erstellung neuer Buslinien. Er ist lediglich @RequestScoped, der er nur innerhalb von buslineeditor.xhtml gebraucht wird und keine Daten über mehrere Seitenaufrufe hinweg gespeichert werden müssen. Der Controller hält Eingabewerte wie Liniennummer, Betriebszeiten sowie die erste Haltestelle zwischen, die der Benutzer über die Oberfläche angibt. Mit createBusLine() wird aus den eingegebenen Daten ein neues BusLine-Objekt erzeugt und in die zentrale BusLineList eingefügt.

Ähnlich dazu ist der **BusStopCreatorController.java**. Er übernimmt die Steuerung bei der Erstellung neuer Haltestellen. und stellt die Schnittstelle zwischen den Eingabefeldern auf der Benutzeroberfläche und der zentralen BusStopList dar. Der Controller verwaltet die Attribute Name und Standort, die beim Anlegen einer Haltestelle ausgefüllt werden. Mit der Methode createBusStop() wird aus diesen Angaben ein neues BusStop-Objekt erzeugt und in den Haltestellenpool aufgenommen. Über clear() können die Eingabefelder zurückgesetzt werden. Auch er ist nur @RequestScoped, da er nur innerhalb von busstopcreator.xhtml genutzt wird und die Daten nach dem Speichern nicht mehr gebraucht werden.

Anders ist dagegen der **BusLineEditorController.java.** Er steuert den Bearbeitungsprozess bestehender Buslinien und ist als @SessionScoped implementiert, sodass die Daten während der gesamten Sitzung erhalten bleiben. Der Controller verwaltet die aktuell ausgewählte Linie und stellt Methoden bereit, um deren Haltestellenliste zu verändern, z.B. addBusStop() und removeBusStop(), mit welchen Haltestellen hinzugefügt oder entfernt werden. Dabei wird geprüft, ob ein Stopp bereits existiert, um doppelte Einträge zu verhindern; entsprechende Hinweise werden über Flash-Messages an die Oberfläche ausgegeben. Darüber hinaus ist der Controller dafür zuständig, den aktuellen Status der Oberfläche zu speichern, d.h. ob diese im Auswahl- oder Bearbeitungsmodus ist.

Am Ende gibt es noch den **PlannerController.java.** Er steuert den mehrstufigen Fahrtenplaner und als @SessionScoped hält er den Workflow-Status sowie die aktuelle Auswahl über die gesamte Sitzung hinweg. Der Controller verwaltet die gewählte Buslinie, die Start- und Zielposition innerhalb der Liniensequenz und berechnet auf Basis der hinterlegten Fahrzeiten die Gesamtfahrzeit sowie die Liste der tatsächlich befahrenen Haltestellen. Über setSelectedLine() wird nach der Linienwahl in den Planungsmodus gewechselt, confirmStops() validiert die Auswahl (Ziel muss hinter Start liegen), ermittelt die Teilstrecke und setzt den Status auf die Ergebnisansicht. Mit clear() setzt den Prozess zurück und mit deleteLine() können BusLinien aus dem System entfernt werden.

# Motivation der verwendeten technischen Mittel

Nachfolgend sollen die zur Entwicklung der Anwendung verwendeten Tools erläutert, sowie einige Architekturentscheidungen begründet werden.

## Verwendete Tools

Die Anwendung wurde in IntelliJ IDEA 2025.2.2 entwickelt und zuletzt am 25.03.2025 gebaut. Des Weiteren wurden verwendet:

* Maven als Paketmanager
* Git als Versionskontrollsoftware
* JDK 1.8 als Projekt-SDK
* JEE 7.0 als Web-Framework
* Glassfish 5.1 als Anwendungsserver

JEE wurde in der Version 7.0 verwendet, da diese standardmäßig JSF 2.2 beinhaltet.

## Archtekturentscheidungen

Der **PlannerController** ist Session-Scoped, da sich der Fahrtenplanungsprozess über mehrere Seiten hinweg erstreckt. Außerdem soll der Benutzer die Möglichkeit haben, zu vorherigen Schritten im Planungsprozess zurückzuspringen. Deshalb ist es notwendig, die bereits eingegebenen Daten wie ausgewählte Linie, Start- und Zielhaltestelle sowie die berechnete Fahrzeit zwischenzuspeichern.

Der **BusLineCreatorController** sowie der **BusStopCreatorController** sind hingegen Request-Scoped, da ihre Prozesse jeweils nur auf einer Seite stattfinden. Die eingegebenen Daten werden direkt verarbeitet, sodass eine Zwischenspeicherung über mehrere Requests nicht erforderlich ist.

Die **BusLineList** und **BusStopList** sind Application-Scoped, da sie als zentrale Datenpools fungieren und ihre Informationen und Funktionalität von verschiedenen Komponenten der Anwendung benötigt werden.

Die User-Bean ist Session-Scoped, da es theoretisch mehrere User geben kann die gleichzeitig das System bedienen, daher müssen sie getrennte User-Informationen haben.

Die Nutzung des Templates ermöglicht das schnelle Entwickeln und Integrieren von neuen Services für die Anwendung, da lediglich der mittlere Bereich „Content“ befüllt werden muss. Außerdem kann so der Header und Footer modular ausgetauscht werden, beispielsweise wenn ein anderer Header mit mehr Kontrollflächen als der aktuelle verwendet werden soll. Diese Änderung würde sich dann automatisch auf alle Seiten der Anwendung auswirken.

# User Manual

Nach dem Start der Anwendung ist der Home Screen wie in Abbildung 1 zu sehen. Von hier aus kann auf alle Services der Anwendung zugegriffen werden.

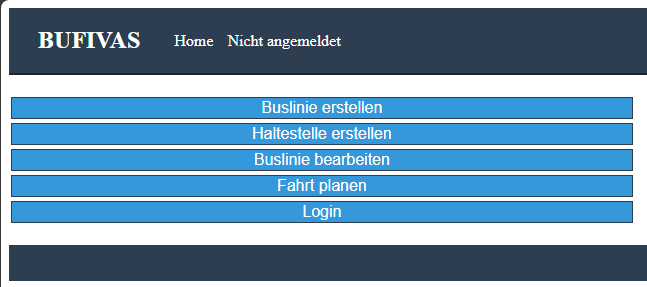


Abbildung 1: Der Home Screen

## Fahrten planen

Um eine neue Planung zu starten, klicken sie auf „Fahrt planen“. Danach werden sie auf das Planungsportal (siehe Abbildung 2) weitergeleitet. Tipp: Sie können jederzeit über die Navigationsleiste am linken Bildschirmrand zu jedem vorherigen Buchungsschritt zurückkehren.

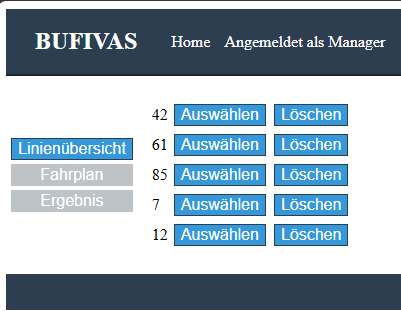


Abbildung 2: Planungs-Portal, Linienübersicht

In der Linienübersicht sind alle im System gespeicherten Linien aufgelistet. Klicken sie bei der gewünschten Linie auf „Auswählen“, um den Fahrplan der Linie aufzurufen, wie in Abbildung 3 zu sehen. Darüber hinaus können hier Manager Linien aus dem System entfernen („Löschen“).

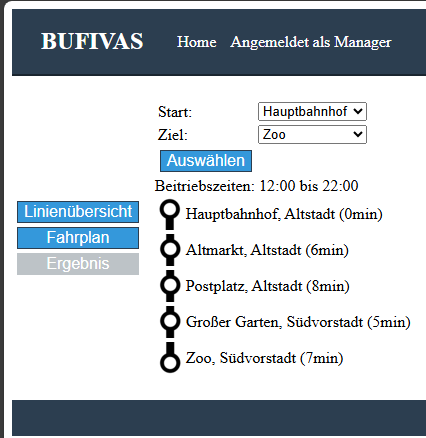


Abbildung 3: Planungs-Portal, Fahrplanansicht

Hier können sie ihre Start- und Zielhaltestelle wählen. Mit Klick auf „Auswählen“ schließen sie die Planung ab und gelangen auf die Übersichtsseite (siehe Abbildung 4).

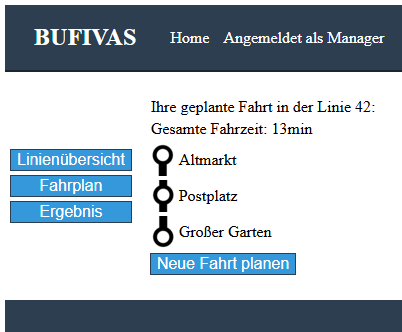


Abbildung 4: Planungs-Portal, Planungsübersicht

Mit „Neue Fahrt planen“ können sie den Buchungsprozess neu starten und eine neue Fahrt planen.

## Linien & Haltestellen anlegen

Auf der Home-Page gelangen sie über die Schaltfläche „Buslinie erstellen“ zum Erstellungsformular, mit dem sich neue Buslinien zur Datenbank hinzufügen lassen:

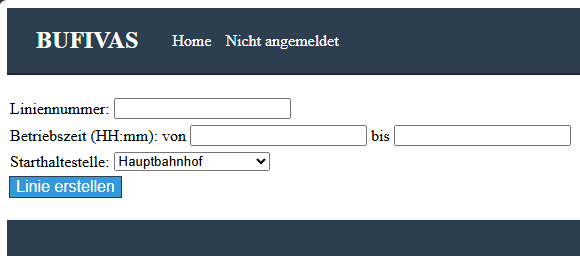


Abbildung 6: Erstellungsformular für Linien

Nachdem sie alle Informationen eingegeben haben, klicken sie auf „Linie erstellen“ um die Daten zu speichern und Linie im System anzulegen.

Das Erstellen einer neuen Haltestelle funktioniert nach dem gleichen Prinzip, geben sie die Informationen ein und Klicken sie auf „Haltestelle erstellen“

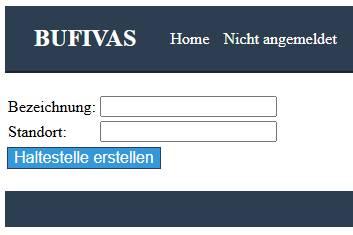


Abbildung 8: Erstellungsformular für Haltestellen

## Linien bearbeiten

Klicken sie zum Bearbeiten einer bereits angelegten Linie auf der Home-Page (siehe Abbildung 1) auf „Buslinie bearbeiten“. So gelangen sie auf die Auswahlseite des Editors (siehe Abbildung 7). Wählen sie in der Liste die Linie aus, die sie bearbeiten möchten und klicken dann auf „Auswählen“, um auf die Bearbeitungsansicht zu gelangen (siehe Abbildung 8).

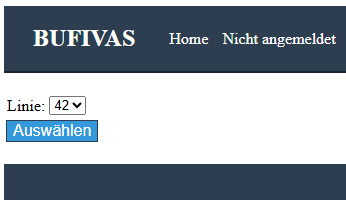


Abbildung 7: Linieneditor (Auswahlansicht)

Hier können sie neue Haltestellen zu der gewählten Linie hinzufügen, oder wieder entfernen, wenn sie als Manager angemeldet sind. Zum Hinzufügen einer Haltestelle müssen sie deren Namen, sowie die Fahrzeit von der letzten Haltestelle aus angeben. Klicken sie dann auf „Hinzufügen“, um die Haltestelle zu der Linie hinzuzufügen.

# Installationshinweise

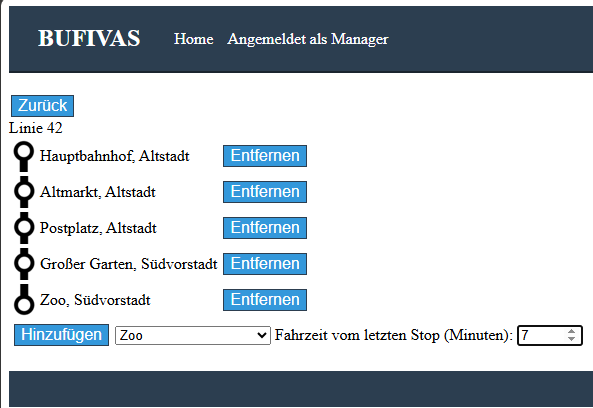


Abbildung 8: Linieneditor (Bearbeitungsansicht)

Um die Anwendung zu nutzen, wird Glassfish 5 benötigt. Wenn Glassfish installiert ist, legen sie die .war Datei in die <GLASSFISHROOT>/domains/<IHRE DOMAIN>/autodeploy/. Starten sie die Glassfish Server mit ihrer Domain und die Anwendung wird automatisch deployed. Ist das getan, können sie über [http://localhost:8080/BUFIVAS-1.0-SNAPSHOT/](http://localhost:8080/FIBVAS-1.0-SNAPSHOT/) auf die Anwendung zugreifen.