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 5 verschiedene Webpages, die sich aus insgesamt 13 verschiedenen Modulen zusammensetzen. Dazu kommen noch 6 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.

**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 unterschiedliche Funktionen der Anwendung aufrufen: das Buchen und Erstellen von Flügen 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}.

**flightcreator.xhtml** stellt eine Eingabemaske zur Erstellung neuer Flüge innerhalb der Anwendung bereit. Der Seiteninhalt wird im Bereich content definiert und enthält ein Formular mit einem zweispaltigen Gitter <h:panelGrid>, in dem die relevanten Flugdaten wie Flugnummer, Abflug- und Zielort, Abflug- und Ankunftszeit, maximale Passagieranzahl sowie Preis eingegeben werden können. Nach erfolgreicher Erstellung wird dem Benutzer eine Bestätigung ausgegeben, sofern es sich nicht um den ersten Seitenaufruf handelt.

Die komplexeste Seite der Anwendung ist **booking.xhtml**, welches den mehrstufigen Buchungsprozess innerhalb der Anwendung abbildet. Sie besteht aus einem zweispaltigen Raster, das in der linken Spalte die Navigationskomponente (bookingNav.xhtml) und in der rechten Spalte dynamisch wechselnde Inhalte je nach Buchungsfortschritt darstellt. Die Darstellung der jeweiligen Buchungsschritte – Flugauswahl (flightdisplay.xhtml), Flugdatenanzeige (flightdetails.xhtml), Eingabe von Passagierinformationen (passengerinfo.xhtml) und Buchungsbestätigung (confirmation.xhtml) – wird anhand des Statuswerts bookingController.status gesteuert.

**bookingNav.xhtml** ist die Navigationsleiste, die während des Buchungsprozesses angezeigt wird. Sie enthält ein Formular mit vertikal angeordneten Schaltflächen, deren Aussehen und Interaktivität durch den aktuellen Statuswert (bookingController.status) gesteuert wird. Damit kann der Benutzer zu jedem Zeitpunkt zu einem früheren Buchungsschritt zurückkehren. Um eine logische Abfolge sicherzustellen, werden Schaltflächen für nachfolgende Schritte deaktiviert (disabled="true") und optisch ausgegraut (customButtonDisabled), solange der jeweilige Statuswert nicht erreicht wurde.

Das **flightdisplay.xhtml** dient der Anzeige verfügbarer Flüge, die noch Plätze frei haben. Sie enthält ein JSF-<h:dataTable>, um die Liste der Flüge (flightList) darzustellen. Für alle Benutzer steht eine Schaltfläche zum Auswählen eines Flugs zur Verfügung, mit der der ausgewählte Flug an den Controller übergeben wird. Zusätzlich wird – ausschließlich für eingeloggte Manager #{user.manager} – eine „Löschen“-Schaltfläche angezeigt, die es erlaubt, Flüge direkt aus der Liste zu entfernen.

**flightdetails.xhtml** zeigt eine detaillierte Liste von Informationen über den zuvor gewählten Flug, wie Abflug- und Zielflughafen, Abflugs- und Ankunftszeit, Preis und noch verfügbare Plätze. Die Daten dafür stammen aus dem aktuell gewählten Flugobjekt bookingController.selectedFlight.

**passengerinfo.xhtml** bildet den dritten Schritt im Buchungsprozess ab und dient der Eingabe der Passagierdaten. Nachdem der Benutzer die benötigen Informationen eingegeben hat, wird er durch betätigen der Schaltfläche „Zur Buchungsübersicht“ zum letzten Buchungsschritt weitergeleitet.

**confirmation.xhtml** schließt den Prozess ab und dient der abschließenden Bestätigung aller eingegebenen Daten. In einer übersichtlichen Tabelle <h:panelGrid> werden sowohl die Fluginformationen (Flugnummer, Abflug- und Ankunftsort/-zeit, Preis) als auch die zuvor eingegebenen Passagierdaten (Vorname, Nachname, Nationalität) dargestellt. Diese Werte stammen aus dem bookingController.

## 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).

Der **BookingController.java** steuert den Ablauf des gesamten Buchungsprozesses innerhalb der JSF-Anwendung und ist als session-bezogene Managed Bean (@SessionScoped) mit dem Namen "bookingController" deklariert. Sie verwaltet die aktuelle Buchungsphase über eine Statusvariable (status), die zwischen den vier Prozessschritten unterscheidet: Flugauswahl (0), Flugdatenansicht (1), Passagierdateneingabe (2) und Buchungsbestätigung (3). Darüber hinaus speichert der Controller das aktuell gewählte Flugobjekt (selectedFlight) sowie die vom Nutzer eingegebenen Passagierinformationen (passengerInfo). Über die Methode getFlightList() werden nur Flüge mit verfügbaren Sitzplätzen aus der zentral verwalteten FlightList gefiltert.

**Flight.java** modelliert die grundlegenden Eigenschaften eines Flugs innerhalb der Anwendung. Sie enthält Informationen wie Flugnummer, Abflug- und Zielort, Abflug- und Ankunftszeit, maximale Passagieranzahl (Capacity) sowie den Flugpreis. Zusätzlich wird über SeatsLeft die aktuelle Verfügbarkeit von Sitzplätzen abgebildet, die bei der Instanziierung dem Wert der Gesamtkapazität entspricht.

**Passenger.java** dient ebenfalls als einfaches Datenmodell zur Erfassung von Passagierinformationen im Buchungsprozess. Die Klasse enthält die drei Attribute name, lastName und passport, welche den Vornamen, Nachnamen und die Nationalität (bzw. Passangabe) eines Fluggastes repräsentieren.

**FlightCreatorController.java** übernimmt die Eingabe- und Erstellungslogik für neue Flüge innerhalb des Flight Creators (flightcreator.xhtml). Es ist eine JSF-Managed Bean und hat eine @RequestScoped Lebensdauer. Wichtig hierbei ist die Methode createFlight(), die ein neues Flight-Objekt erzeugt und über die injizierte FlightList zur Gesamtliste hinzugefügt. Danach werden alle Eingabefelder durch die Methode clear() zurückgesetzt, und das firstLoad auf false gesetzt. Diese Flag wird genutzt, um bei der Anzeige der Seite zwischen Erstaufruf und erfolgreicher Flugerstellung zu unterscheiden.

Die **FlightList.java** verwaltet eine zentrale Liste aller verfügbaren Flüge innerhalb der Anwendung und stellt diese als @ApplicationScoped-Bean bereit. Flüge werden in einer ArrayList<Flight> gespeichert, die beim Start der Anwendung mit einigen Beispielflügen befüllt werden. Über die Methoden getFlightList(), addFlight() und removeFlight() kann auf die Liste zugegriffen und diese verändert werden. Neue Flüge können beispielsweise vom FlightCreatorController hinzugefügt werden, während Manager über den BookingController Flüge wieder entfernen können.

# 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 2024.3.5 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 als Anwendungsserver
* Primefaces 5.3 & Primefaces Themes 1.0.10 als Style-Bibliotheken

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

## Archtekturentscheidungen

Nachfolgend soll nun zunächst die Begründung für die einzelnen Lebenszyklen der Java-Beans dargelegt werden.

Der **BookingController** ist Session-Scoped, da sich der Buchungsprozess über mehrere Seiten hinweg streckt. Außerdem soll der Benutzer dazu in der Lage sein, zu vorherigen Schritten im Buchungsprozess zurückzuspringen, sodass es notwendig ist die eingegebenen Daten zwischenzuspeichern.

Der **FlightCreatorController** ist dagegen Request-Scoped, da der Prozess nur auf einer Seite stattfindet und alle Daten direkt verarbeitet werden und eine Speicherung somit nicht erforderlich ist.

Die **FlightList** ist Application-Scoped da diese als Database-Bean agiert und ihre Informationen und Funktionalität von mehreren Orten in der Anwendung her zugänglich sein müssen.

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.

Eine andere wichtige Architekturentscheidung ist die Modularität der Anwendung, zum einen durch das Template, als auch durch den Booking-Prozess. Die Herangehensweise soll nun begründet werden.

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.

Der Booking Service ist ebenfalls modular aufgebaut, da es gut sein kann, dass in Zukunft einzelne Schritte ausgetauscht oder ersetzt werden müssen. Das ist mit dieser Herangehensweise sehr einfach, da nur der Name der importierten Datei ausgetauscht werden muss. Ähnlich einfach ist die Erweiterung um zusätzliche Schritte, da nur neue States (z.B. 4) angelegt werden müssen.

# 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.

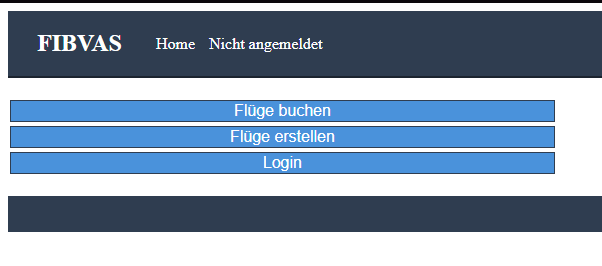


Abbildung 1: Der Home Screen

## Flüge buchen

Um eine neue Buchung zu starten, klicken sie auf „Flüge buchen“. Danach werden sie auf das Booking-Portal (siehe Abbildung 2) weitergeleitet. Tipp: Sie können jederzeit über die Navigationsleiste am linken Bildschirmrand zu jedem vorherigen Buchungsschritt zurückkehren.

Auf der Flugübersichtsseite sind alle Flüge aufgelistet, für die noch Buchungen möglich sind. Wenn sie ihren Wunschflug gefunden haben, klicken sie neben der Flugnummer auf „Auswählen“, um die detaillierte Übersicht des Fluges aufzurufen, wie in Abbildung 3 zu sehen.



Abbildung : Booking-Portal, Detaillierte Flugansicht

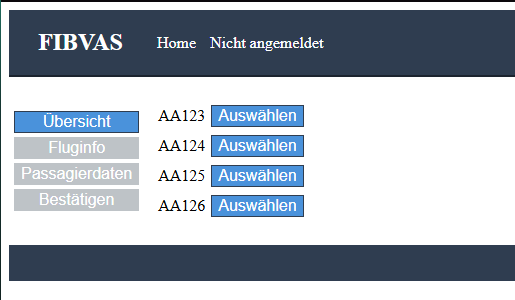


Abbildung : Booking-Portal, Flugübersicht

Wenn sie sich sicher sind, dass sie diesen Flug buchen möchten, klicken sie auf „Buchung starten, um zur Eingabemaske für ihre Informationen zu gelangen (siehe Abbildung 4).

Wenn sie ihre Daten eingegeben haben, klicken sie auf „Zur Buchungsübersicht“ um ihre Buchung nochmals zu prüfen. Sind alle Daten korrekt, klicken sie auf „Buchung abschließen“. Achtung: Danach kann ihre Buchung nicht mehr rückgängig gemacht werden!



Abbildung : Booking-Portal, Abschließende Prüfung

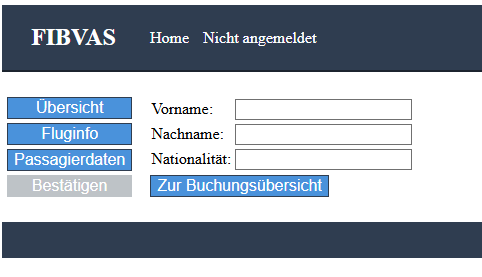


Abbildung : Booking-Portal, Passagierdateneingabe

Nach erfolgreicher Buchung können sie eine neue Buchung durch klicken auf „Neue Buchung“

## Flüge erstellen

Auf der Home-Page gelangen sie über die Schaltfläche „Flüge erstellen“ zum FlightCreator, mit dem sich neue Flüge zur Datenbank hinzufügen lassen. Die Eingabemaske ist in Abbildung 6 zu sehen.



Abbildung : FlightCreator

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

## Flüge löschen

Wenn sie einen Flug aus der Datenbank löschen wollen, müssen sie als Manager angemeldet sein. Klicken sie dazu auf der Home-Page (siehe Abbildung 1) auf „Login“ danach wird ihnen am oberen Bildschirmrand angezeigt das sie angemeldet sind, wie in Abbildung 7 dargestellt.

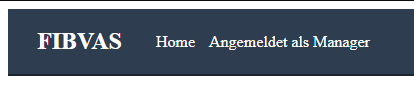


Abbildung : Anmeldestatus

Danach gehen sie auf „Flüge erstellen“ um zur Flugübersicht zu gelangen. Sind sie als Manager angemeldet, so erscheint neben „Auswählen“ eine weitere Schaltfläche „Löschen“ (siehe Abbildung 8). Klicken sie auf diese, um den Fug zu löschen.

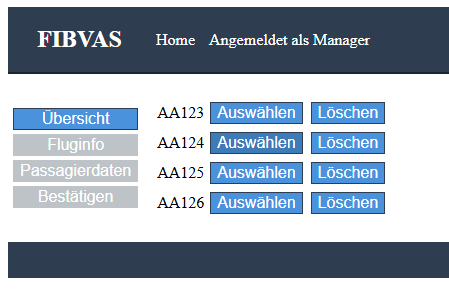


Abbildung : Flugliste als Manager

# Installationshinweise

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 geschehen, können sie über [http://localhost:8080/BUFIVAS-1.0-SNAPSHOT/](http://localhost:8080/FIBVAS-1.0-SNAPSHOT/) auf die Anwendung zugreifen.