MyHike – Entwicklerhandbuch

Inhalt

[1. Einleitung: 3](#_Toc157087890)

[Projektübersicht: 3](#_Toc157087891)

[Zielgruppe: 3](#_Toc157087892)

[Die Zielgruppen für diese Webseite wären Wanderer mittleren Alters, Volksschullehrer, Touristen und ältere Personen. 3](#_Toc157087893)

[2. Installationsanleitung: 3](#_Toc157087894)

[Systemanforderungen: 3](#_Toc157087895)

[Installationsprozess: 3](#_Toc157087896)

[3. Grundlagen: 4](#_Toc157087897)

[Architektur: 4](#_Toc157087898)

[Technologiestack: 5](#_Toc157087899)

[Grundlegende Konzepte: 6](#_Toc157087900)

[4. Konfiguration: 6](#_Toc157087901)

[Konfigurationsdateien: 6](#_Toc157087902)

[Umgebungsvariablen: 6](#_Toc157087903)

[Konfigurationsoptionen: 7](#_Toc157087904)

[5. Nutzung der Software: 7](#_Toc157087905)

[Benutzerschnittstelle: 7](#_Toc157087906)

[Datenbank-Dokumentation: 13](#_Toc157087907)

[6. Entwicklungsrichtlinien: 13](#_Toc157087908)

[Coding Standards: 13](#_Toc157087909)

[Best Practices: 13](#_Toc157087910)

[Code Reviews: 14](#_Toc157087911)

[7. Fehlerbehebung und Debugging: 14](#_Toc157087912)

[Fehlermeldungen: 14](#_Toc157087913)

[Debugging-Tipps: 14](#_Toc157087914)

[8. Versionshinweise: 14](#_Toc157087915)

[Versionshinweise: 14](#_Toc157087916)

[9. Sicherheit: 15](#_Toc157087917)

[Sicherheitsrichtlinien: 15](#_Toc157087918)

[10. Ressourcen 15](#_Toc157087919)

[Hilfreiche Links: 15](#_Toc157087920)

[11. Kontaktinformationen: 15](#_Toc157087921)

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​​

# 1. Einleitung:

## Projektübersicht:

Im MyHike-Projekt werden wir uns auf die Erstellung einer Wander-Webanwendung konzentrieren, die völlig kostenlos Informationen zu mehreren Wanderwegen bereitstellt. MyHike wird sich darüber hinaus auf die Bereitstellung einer leicht navigierbaren Benutzeroberfläche konzentrieren und Benutzern durch die Bereitstellung zahlreicher Filteroptionen dabei helfen, ihre perfekte Wanderroute zu finden.

## Zielgruppe:

## Die Zielgruppen für diese Webseite wären Wanderer mittleren Alters, Volksschullehrer, Touristen und ältere Personen.

Für Wanderer ist es wichtig Strecken zu finden, die sie am meisten ansprechen. Sei es vom Schwierigkeitsgrad her oder von der Ausdauerkraft. Deshalb ist es für diese Gruppe sehr wichtig, dass es Wanderungen von Anfänger-Niveau bis Profi-Niveau gehen. Volksschullehrer hingegen achten dabei mehr auf die Sicherheit, Dauer und Schwierigkeitsgrad, um die Kinder zeitgerecht und unverletzt wieder zurückzubringen. Für Touristen ist es wichtig, dass sie die Landschaft auf dem Weg besichtigen können. Deshalb spielen Landschaft, Sehenswürdigkeiten oder eventuell auch Attraktionen die Hauptrolle. Ältere Personen, die sich nicht allzu sehr anstrengen möchten, aber doch in Bewegung bleiben und mit Enkelkindern Zeit verbringen möchten, suchen sich eher „leichtere“ und kürzere Strecken aus. Für diese Gruppe ist es aber wichtig, dass unter „Points of interest“ Gasthäuser oder Spielplätze zu finden sind.

# 2. Installationsanleitung:

## Systemanforderungen:

Da es sich bei der Software um eine Webapplikation handelt sollte sie auf jedem System verfügbar sein, welches eine funktionierende Internetanbindung zur Verfügung stellt und in der Lage ist einen Browser auszuführen.

## Installationsprozess:

Sollte die **Applikation auf einem öffentlich zugänglichen Webserver gehostet** werden, so sollte natürlich kein Installationsprozess von Nöten sein, um die Applikation zu nutzen, in diesem Fall würde es genügen die DNS-Adresse der Applikation in einen Browser einzugeben oder auch die Webapplikation mit Hilfe einer Suchmaschine zu finden.

Nach **momentanem Stand** ist die Applikation aber noch nicht öffentlich gehostet. Aufgrund dessen müssen um die MyHike-Applikation zu Nutzen folgende Schritte getätigt werden:

* Clonen des Repository von github: <https://github.com/michaelreinprecht/HikingApp>
* Installation eines Tomcat Servers, z.B.: via IntelliJ (hierfür ist die IntelliJ Ultimate Edition notwendig, siehe auch Tomcat Installation unter [Hilfreiche Links](bookmark://_Hilfreiche_Links:))
* In IntelliJ unter dem „Run“ Menü -> „Edit Configurations“.
* Durch Drücken des „+“ Button eine neue Configuration erstellen, in der daraufhin erschienenen Liste „Tomcat Server“ auswählen.
* Daraufhin entweder unter „Application Server“ einen Tomcat Server (Version 10.1.5) auswählen. Falls kein korrekter Tomcat Server ersichtlich ist, kann unter Configure mithilfe der Tomcat Server Binaries ein neuer Server erstellt werden, diese Binaries sind im Github ebenfalls enthalten (apache-tomcat-10.1.5).
* An diesem Punkt wird folgende Warnung angezeigt werden:



Klicken Sie hier auf Fix und wählen daraufhin das „MyHike: war exploded“ Artifakt aus.

* Folgend auf „Apply“ und „Ok“ klicken.
* Spätestens wenn sie versuchen die Konfiguration zu starten sollte rechts unten eine Warnung erscheinen, die auffordert „Lombok Annotation Processing“ zu enablen. -> Enablen von Lombok Annotation Processing (wird in Teilen des Projektes verwendet, um Getter und Setter zu generieren).
* Als letzten Schritt muss sich über den Cisco Secure Client (AnyConnect VPN) mit dem Netzwerk der FHV-Dornbirn verbunden werden (insofern man sich nicht bereits in diesem Netzwerk befindet), da die PostgreSQL Datenbank des Projektes von der FHV-Dornbirn gehostet wird. Siehe AnyConnect VPN unter [Hilfreiche Links](bookmark://_Hilfreiche_Links:).

An diesem Punkt sollte die Applikation mit der oben erstellten Konfiguration problemlos ausgeführt werden können.

# 3. Grundlagen:

## Architektur:

Eine Übersicht über die Architektur der Software, die Architektur wurde von uns hier mithilfe eines C4-Modells dargestellt:

Ein Bild, das Text, Diagramm, Screenshot, Plan enthält.

Automatisch generierte Beschreibung

## Technologiestack:

Folgende Technologien, Frameworks und Bibliotheken wurden in dem Projekt verwendet:

* **Entwickeln, Kompilieren und Ausführen der Applikation:**
* IntelliJ Ultimate IDE
* JSP (+ HTML, CSS und kleinere Teile JavaScript)
* Tomcat Server
* AnyConnect
* Lombok Annotation Processing
* Maven Build-Tool (Alternative wäre z.B. Gradle)
* **Datenbank Technologien:**
* PostgreSQL
* Hibernate und JPA Datenbank Anbindung
* **Erweiterte Features:**
* Leaflet – OpenStreetMap
* Open Route Service
* **Testen der Applikation:**
* JUnit
* Selenium
* Mockito

## Grundlegende Konzepte:

**Domain Driven Design:**

Domain-Driven Design (DDD) ist ein konzeptioneller Ansatz für die Entwicklung von Software, der sich darauf konzentriert, das Verständnis der Geschäftswelt (Domain) direkt in den Softwaremodellen zu reflektieren. Wichtig sind dabei unter anderem die Verwendung einer für die Softwareentwicklung vereinheitlichten Sprache (Ubiquitous Language). Weiters wird dabei versucht den Kontext eines Systems in verschiedene festgelegte Subdomänen, namens „Bounded Contexts“ aufzuteilen.

**MVC-Prinzip:**

Das MVC-Prinzip steht für Model-View-Controller und ist ein Architekturmuster, das in der Softwareentwicklung weit verbreitet ist. Es hilft, die Struktur von Anwendungen zu organisieren und die Trennung von Anwendungslogik, Benutzeroberfläche und Daten zu fördern. Es besteht aus den folgenden drei Hauptkomponenten:

* Model: Das Model repräsentiert die Anwendungsdaten und die Geschäftslogik. Es enthält die Datenstrukturen, die für die Anwendung benötigt werden, und definiert die Methoden und Regeln, die auf diese Daten angewendet werden. Typischerweise ist das Model unabhängig von der Benutzeroberfläche und dem Controller.
* View: Die View ist für die Darstellung der Daten verantwortlich, die vom Modell bereitgestellt werden. Sie präsentiert Informationen dem Benutzer, zeigt also lediglich die Daten an, die vom Modell bereitgestellt werden.
* Controller: Der Controller verarbeitet Benutzereingaben und aktualisiert das Model und die View entsprechend. Er ist für die Anwendungslogik verantwortlich und dient als Vermittler zwischen Model und View. Wenn ein Benutzer mit der Benutzeroberfläche interagiert, leitet der Controller die Anfragen an das Model weiter und aktualisiert dann die View basierend auf den Änderungen im Model.

# 4. Konfiguration:

## Konfigurationsdateien:

Die einzigen Konfigurationsdateien, welche zu diesem Projekt unter Umständen benötigt, werden könnten sind die Tomcat Binaries, welcher verwendet werden, um einen Tomcat Server zu erstellen/installieren. Diese Dateien sind in dem Github zu finden unter „apache-tomcat-10.1.5“. Wie diese Dateien verwendet werden, wird erklärt unter [Installationsprozess](bookmark://_Installationsprozess:).

## Umgebungsvariablen:

**Umgebungsvariable für Sichern von Passwörtern:**

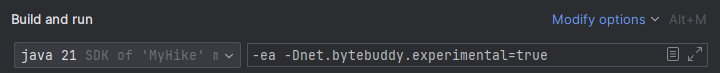
Wir haben momentan keine Umgebungsvariablen für das Abspeichern/Sichern von Passwörtern. Für die Entwicklung dieses Projektes (welches wohl nie veröffentlicht wird) hielten wir es nicht für notwendig das Datenbankpasswort in einer Umgebungsvariable zu sichern.

**Umgebungsvariable für Testing:**

Um das Mocking bei Selenium Tests richtig zu realisieren, war es notwendig in eine Umgebungsvariable “MOCK” zu erstellen, welche in einem ServletListener abgefragt werden kann. Ist die Umgebungsvariable “MOCK” gesetzt (wie in dem unten angehängten Bild), dann werden gewisse Teile der Datenbank standardmäßig gemockt (dies geschieht in dem ServletListener  - MockListener.java). Dies bewirkt, dass die Selenium Tests keine bestehenden Datenbankobjekte abändern und auch keine neuen hinzufügen können.

## Konfigurationsoptionen:

Für **das standardmäßige Ausführen** der Applikation sind keine besonderen Konfigurationsvariablen notwendig. Für **Testing** ist es jedoch nötig in der Run-Konfiguration innerhalb der VM-Options folgende Konfigurationsvariablen anzugeben:



Dies ist der Grund, da die von der Mockito-Library verwendete Version von ByteBuddy Java 21 nicht unterstützt. Warum dies der Fall ist, ist nicht ganz klar, da die Mockito-Library die ByteBuddy Version 1.14.5 verwendet, welche eigentlich Java 21 unterstützen sollte, jedoch führt die Verwendung zu einer Fehlermeldung. In der Zukunft unter Verwendung einer neueren Mockito-Version kann diese Option unter Umständen entfernt werden.

# 5. Nutzung der Software:

## Benutzerschnittstelle:

Eine Anleitung zur Benutzung der grafischen Benutzeroberfläche (falls vorhanden).

**Login.jsp:**

1.Importe:

Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

2. Navigationsleiste:

Bootstrap-basierte Navigationsleiste mit Links zu "Discover", "Create Hike" und "Your Hikes".

Der Link "Login" ist aktiv, wenn der Benutzer sich auf der Login-Seite befindet.

3. Login-Formular

Ein Formular, in dem Benutzer ihren Benutzernamen und ihr Passwort eingeben können.

Anzeige einer Fehlermeldung, falls beim Anmeldevorgang ein Fehler auftritt.

 4. Registrierungslink

Ein Link, um zur Registrierungsseite zu navigieren, falls der Benutzer noch kein Konto hat.

 5. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**Registration.jsp**

1. Importe

Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

2. Navigationsleiste:

Bootstrap-basierte Navigationsleiste mit Links zu "Discover" und "Create Hike".

 3. Registrierungsformular:

Ein Formular, in dem Benutzer ein neues Konto erstellen können, indem sie einen Benutzernamen und ein Passwort angeben.

Bestätigungsfeld für das Passwort, um eine korrekte Eingabe sicherzustellen.

4. Bereich für Benachrichtigungen:

Anzeige von Erfolgs- oder Fehlermeldungen mithilfe eines benutzerdefinierten JSP-Tags (`<tags:multiAlert>`).

5. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**Hikelist.jsp**

1. Importe:

Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

Import von Klassen und Modellen für die Datenverarbeitung.

2. Navigationsleiste:

 Bootstrap-Navigationsleiste mit Links zu "Discover", "Create Hike" und "Your Hikes".

 "Your Hikes"-Link ist nur sichtbar, wenn ein Benutzer angemeldet ist.

3. Suchleiste:

 Ein Formular für die Hikesuche mit Eingabefeldern für Name oder Region.

 4. Filteroptionen:

Filteroptionen für maximale Dauer, Entfernung, Höhe, Ausdauer, Landschaft, Stärke und Schwierigkeitsbewertung.

Checkboxen für empfohlene Monate.

5. Hike-Anzeige:

Anzeige der gefilterten Hikes mit Bild, Name und Daten (Dauer, Entfernung, Höhe).

Falls keine Hikes gefunden werden, wird eine entsprechende Meldung angezeigt.

6. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**Edit.jsp**

1. Importe:

 Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

 Import von Klassen und Modellen für die Datenverarbeitung.

2. Navigationsleiste:

 Bootstrap-Navigationsleiste mit Links zu "Discover", "Create Hike" und "Your Hikes".

 "Your Hikes"-Link ist nur sichtbar, wenn ein Benutzer angemeldet ist.

3. Formular zur Bearbeitung:

Formular zum Bearbeiten von Hike-Details, einschließlich Name, Beschreibung, Dauer, Entfernung, Höhe und Bewertungen.

4. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**Discover.jsp**

1. Importe:

 Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

 Import von Klassen und Modellen für die Datenverarbeitung.

2. Navigationsleiste:

 Bootstrap-Navigationsleiste mit Links zu "Discover", "Create Hike" und "Your Hikes".

 "Your Hikes"-Link ist nur sichtbar, wenn ein Benutzer angemeldet ist.

3. Anzeige entdeckter Hikes:

Anzeige von Hikes mit Bild, Name und kurzer Beschreibung.

Links zu Detailseiten für jedes Hike.

4. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**Detail.jsp**

1. Importe:

Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

Import von Klassen und Modellen für die Datenverarbeitung.

2. Navigationsleiste:

Bootstrap-Navigationsleiste mit Links zu "Discover", "Create Hike" und "Your Hikes".

"Your Hikes"-Link ist nur sichtbar, wenn ein Benutzer angemeldet ist.

3. Hike-Details:

Anzeige detaillierter Informationen zu einem ausgewählten Hike, einschließlich Bild, Name, Dauer, Entfernung, Höhe und Bewertungen.

Beschreibung des Hikes.

 4. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**CreatedHikes.jsp**

1. Importe:

Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

Import von Klassen und Modellen für die Datenverarbeitung.

2. Navigationsleiste:

Bootstrap-Navigationsleiste mit Links zu "Discover", "Create Hike" und "Your Hikes" (aktiv).

"Your Hikes"-Link ist nur sichtbar, wenn ein Benutzer angemeldet ist.

3. Anzeige erstellter Hikes:

Anzeige der vom Benutzer erstellten Hikes mit Bild, Name und kurzer Beschreibung.

Links zu Detailseiten für jedes Hike.

4. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**Create.jsp**

1. Importe:

Import der erforderlichen Bibliotheken und Stylesheets (Bootstrap, Font Awesome).

Import von Klassen und Modellen für die Datenverarbeitung.

2. Navigationsleiste:

Bootstrap-Navigationsleiste mit Links zu "Discover", "Create Hike" (aktiv) und "Your Hikes".

"Your Hikes"-Link ist nur sichtbar, wenn ein Benutzer angemeldet ist.

3. Formular zum Erstellen eines Hikes:

Formular zum Eingeben von Hike-Details wie Name, Beschreibung, Dauer, Entfernung, Höhe und Bewertungen.

4. Bootstrap- und JavaScript-Importe:

Import erforderlicher Bootstrap- und JavaScript-Bibliotheken.

**CSS-Stylesheets:**

  - Stelle sicher, dass die benötigten CSS-Dateien für das Styling eingebunden sind.

  - Beispiel: `<link rel="stylesheet" type="text/css" href="pfad/zum/dein-styling.css">`

**Servlets:**

FilterHikesServlet: Verarbeitet die Anwendung von Filtern auf Wanderungen.

LoginServlet: Verarbeitet die Anmeldung von Benutzern.

LogoutServlet: Verarbeitet die Abmeldung von Benutzern.

RegistrationServlet: Verarbeitet die Registrierung neuer Benutzer.

CreateHikeServlet: Erstellt eine neue Wanderung im System.

EditHikeServlet: Verarbeitet die Bearbeitung von existierenden Wanderungen.

DeleteHikeServlet': Verarbeitet das Löschen von Wanderungen aus dem System.

AddCommentServlet: Verarbeitet das Hinzufügen von Kommentaren zu einer Wanderung; leitet bei Fehlern auf die Detailseite der Wanderung um und zeigt Fehlermeldungen an.

DeleteCommentServlet: Verarbeitet das Löschen von Kommentaren; entfernt den Kommentar aus der Liste der Wanderungskommentare, aktualisiert die Wanderung und leitet bei Erfolg auf die Detailseite um.

SoftDeleteHikeServlet: Setzt das "isDeleted"-Flag für eine Wanderung auf "true", um eine weiche Löschung zu simulieren; leitet bei Fehlern auf die Detailseite der Wanderung um und zeigt Fehlermeldungen an.

AddPOIServlet: Fügt einen neuen Punkt von Interesse (POI) zu einer Wanderung hinzu; verarbeitet die Authentifizierung, aktualisiert die Wanderung mit dem neuen POI und gibt Fehlermeldungen bei Bedarf an die Ajax-Anfrage zurück.

POIServletUtils: Enthält grundlegende Methoden, die in vielen POI-bezogenen Servlets verwendet werden, einschließlich der Authentifizierung für die Eigentümerschaft der Wanderung.

MockListener: Initialisiert Mock-Objekte für die Datenbank-Fassade und gibt festgelegte Testdaten zurück, wenn die Umgebungsvariable "MOCK" vorhanden ist.

ResourceServlet: Verwaltet die JPA-EntityManager-Instanz für die gesamte Anwendung, indem sie bei Initialisierung erstellt und bei Abschaltung geschlossen wird.

ServletUtils: Enthält allgemeine Methoden für Servlets, wie z.B. das Konvertieren eines hochgeladenen Bilds in Base64-Format.

**JavaScript für Funktionalitäten auf bestimmten Seiten:**

**create\_edit.js:**

Zeigt eine Karte an, auf der die Route für eine neue oder bearbeitete Wanderung erstellt wird.

Erlaubt das Festlegen von Start- und Zielmarkern auf der Karte.

**detail.js:**

Zeigt eine interaktive Karte mit der detaillierten Route der ausgewählten Wanderung an.

**detailMap.js:**

Ähnlich wie 'detail.jsp', jedoch spezialisiert auf die Kartenansicht.

**EditMap.js:**

Ähnlich wie 'create\_edit.jsp', jedoch spezialisiert auf die Kartenansicht beim Bearbeiten einer Wanderung.

Funktionen:

- Die Karte wird mit Leaflet erstellt.

- Koordinaten werden aus den Datenattributen der Karten-Elemente gelesen.

- Start- und Zielmarker werden hinzugefügt und mit benutzerdefinierten Icons versehen.

- Die Routenlinie wird auf der Karte gezeichnet.

**Systemanforderungen:**

Die Benutzeroberfläche dieser Software erfordert die Verwendung eines HTML5-kompatiblen Browsers und eine Internetverbindung für den Zugriff auf die Funktionen und Inhalte.

**Barrierefreiheit und Semantik:**

Aktuell entspricht die Barrierefreiheit der Software nicht den optimalen Standards. Es wurden vermehrt `<div>`-Elemente verwendet, die semantische Strukturen behindern. Um die Barrierefreiheit zu verbessern, sollen semantische HTML-Tags und ARIA-Labels (Accessible Rich Internet Applications) eingeführt werden. Dies gewährleistet eine bessere Interpretation durch Assistenztechnologien und verbessert die Nutzererfahrung für Menschen mit Behinderungen.

**Internationalisierung:**

Die Software ist derzeit ausschließlich in Englisch verfügbar. Zukünftig ist jedoch geplant, die Anwendung für Benutzer in verschiedenen Sprachen zugänglich zu machen. Entwickler sollten bei der Implementierung von neuen Funktionen sicherstellen, dass die Software leicht auf andere Sprachen erweiterbar ist und dass Benutzer die gewünschte Sprache auswählen können.

**Suchmaschinenoptimierung (SEO):**

Die Suchmaschinenoptimierung (SEO) ist bereits implementiert und erfüllt die aktuellen Anforderungen. Derzeit erlauben wir Suchmaschinen-Robotern den Zugriff auf unsere Seite, gewährleisten schnelle Ladezeiten und setzen auf die Generierung statischer HTML-Seiten durch unseren Server. Diese Maßnahmen zielen darauf ab, die Sichtbarkeit in Suchmaschinen zu maximieren und eine optimale Nutzererfahrung sicherzustellen. Es besteht jedoch die Möglichkeit, die SEO-Strategie in der Zukunft zu überprüfen und weiter zu verbessern, um sicherzustellen, dass sie den aktuellen Best Practices entspricht.

**Softwarefunktionalitäten:**

Die Benutzeroberfläche bietet verschiedene Funktionen, die je nach Benutzerstatus unterschiedlich zugänglich sind:

Öffentlich zugängliche Seiten:

   - Detailseite (Discover) für öffentliche Informationen.

   - Filterseite (Hikelist) für allgemeine Suchanfragen.

   - Hike Seite für die Anzeige Wanderroute.

   - Registrierungsseite (Registration) für neue Benutzer.

   - Anmeldeseite (Login) für bestehende Benutzer.

Zusätzliche Funktionen für eingeloggte Benutzer:

   - Erstellen einer neuen Wanderroute (Create Hike).

   - Löschen einer eigenen Wanderroute (Delete Hike), nur für den Benutzer oder den Administrator.

   - Bearbeiten einer eigenen Wanderroute (Modifizierung Hike), nur für den Benutzer oder den Administrator.

   - Ansicht "Meine Wanderrouten" (My Hike) für die Verwaltung eigener Wanderrouten.

## Datenbank-Dokumentation:

Bei der verwendeten Datenbank handelt es sich um eine PostgreSQL Datenbank. Das Datenbankschema sieht wie folgt aus:

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

# 6. Entwicklungsrichtlinien:

## Coding Standards:

Generell gilt natürlich, dass der **Code in Englisch** geschrieben werden muss! Natürlich sollten auch **sinnvolle** **Kommentare** hinzugefügt werden.

Für Coding Standards wurden gewöhnliche Java-Standards verwendet.

* Verwenden von aussagekräftigen und konsistenten Namen für Klassen, Methoden, Variablen und Konstanten.
* Klassen und Typen beginnen mit einem Großbuchstaben, Methoden und Variablen mit einem Kleinbuchstaben.

## Best Practices:

* Verwenden der Hot-Reload Funktion beschleunigt die Entwicklungsgeschwindigkeit stark.
* Gruppen von Servlets anhand ihres (Bounded) Kontext in Packages unterteilen.
* Vermeiden von Duplicate Code!
* Klare Trennung zwischen Datenbank und Anwendungslogik.
* **Laufendes** schreiben von Dokumentation und Tests(wurde bei uns leider etwas vernachlässigt)

## Code Reviews:

Code Reviews werden von einer Person durchgeführt, welcher an der Erstellung des Codes, welcher zu Review steht, nicht beteiligt war.

# 7. Fehlerbehebung und Debugging:

## Fehlermeldungen:

Dieser Punkt wird leider etwas kurzgehalten, da wir nicht von Anfang an wussten, dass eine Dokumentation notwendig ist und wir somit leider auch keine Fehlermeldungen dokumentiert haben.

Daher hier nur eine Erklärung für die aller häufigsten Fehler Szenarien:

* Keine Verbindung zu Datenbank möglich (Persistence Error) -> entweder Datenbank ist down oder man hat sich nicht mit Netzwerk der FHV-Dornbirn verbunden. Oder das persistence.xml File ist fehlerhaft.
* Hibernate Error (Could not retrieve Result Set) -> Tritt oft auf, wenn Stand der Datenbank nicht mit dem aktuellen Hibernate Setup der Applikation übereinstimmt.
* Artifact Exploded -> kann aus sehr vielen verschiedenen Gründen auftreten, einer der häufigsten Gründe ist jedoch, dass die Projektstruktur geändert wurde (z.B. ein Servlet in ein neues Package verschoben), in diesem Fall wird diese Verschiebung oft (entweder von IntelliJ oder Github) falsch registriert, es reicht meist das Projekt neu von Github zu clonen.

## Debugging-Tipps:

Das Debugging gestaltet sich recht einfach, da es möglich ist auch innerhalb der JSP Seiten Debug Points zu setzen.

Was etwas umständlicher ist, ist das Debuggen der kleinen JavaScript Teile von IntelliJ aus, hierbei bietet es sich an diese einfach mit dem Entwicklertool des Browsers zu debuggen.

Ob es sich um einen Datenbank Error handelt, ist klar, sobald die Datenbankmethoden nichts korrektes mehr zurückgeben. Wenn es sich um ein Problem mit Hibernate handelt, bietet es sich hier an zuerst festzulegen von welchem Model das Problem ausgeht und sich dieses Model dann nochmals genauer anzusehen.

# 8. Versionshinweise:

## Versionshinweise:

Verwendet wurde Java 21, die Versionen der verwendeten Libraries können innerhalb des pom.xml Files gefunden werden.

# 9. Sicherheit:

## Sicherheitsrichtlinien:

Sollte geplant werden die Software jemals zu veröffentlichen, so wäre es unabdinglich Umgebungsvariablen für die Datenbankverbindung und für den Testnutzer festzulegen.

# 10. Ressourcen

## Hilfreiche Links:

Verweise auf weiterführende Ressourcen, Dokumentationen und Supportkanäle:

Tomcat Installation in IntelliJ: <https://www.jetbrains.com/guide/java/tutorials/working-with-apache-tomcat/using-existing-application/>

AnyConnect VPN: <https://ilias.fhv.at/ilias.php?baseClass=ilSAHSPresentationGUI&ref_id=655762>

OpenRouteService: <https://openrouteservice.org/>

Mockito-Github (vielleicht für Updates checken): <https://github.com/mockito>

# 11. Kontaktinformationen:

Bei Problemen melden Sie sich bitte bei einer der folgenden E-Mail-Adressen:

* [michael.reinprecht@students.fhv.at](mailto:michael.reinprecht@students.fhv.at)
* [cindy.sebastian@students.fhv.at](mailto:cindy.sebastian@students.fhv.at)
* [furkan.kilic@students.fhv.at](mailto:furkan.kilic@students.fhv.at)
* [ismail.aschabow@students.fhv.at](mailto:ismail.aschabow@students.fhv.at)
* [kenan.ergüven@students.fhv.at](mailto:kenan.ergüven@students.fhv.at)
* [tugce.bilgic@students.fhv.at](mailto:tugce.bigic@students.fhv.at)
* [veronika.kircsev@students.fhv.at](mailto:veronika.kircsev@students.fhv.at)