**„Employee Administration“ Handbook:**

Inhalt

[Technical details: 2](#_Toc206590814)

[Introduction: 8](#_Toc206590815)

[Shared views across all levels of staff: 8](#_Toc206590816)

[Time view: 8](#_Toc206590817)

[Vacation screen: 9](#_Toc206590818)

[Views for manager and admin: 10](#_Toc206590819)

[Request view: 10](#_Toc206590820)

[Employee overview view: 11](#_Toc206590821)

[Creating an employee: 11](#_Toc206590822)

[Updating an employee: 12](#_Toc206590823)

[Deleting an Employee: 13](#_Toc206590824)

[The Master Data button: 13](#_Toc206590825)

[Employee Form View: 13](#_Toc206590826)

[Under the “create new address” Button: 15](#_Toc206590827)

[Create new address view: 15](#_Toc206590828)

[Master Data view of employee: 15](#_Toc206590829)

# Technical details:

1. **General Information**

**1.1. Project name**

Employee Administration

**1.2. Version**

1.0.0

**1.3. Author**

Fabian Andiel

**1.4 Date**

19.08.2025

**1.5 Purpose of program**

Handling employee master data, creating, updating, deleting employees, vacation and vacation request management,…

1. **System requirements**

**2.1. Java-Version**

Java 17

**2.2. Build-Tool**

Maven

**2.3. Frameworks**

Hibernate

**2.4. Datenbank**

PostgresSQL

**2.5 Versionsverwaltung**

Git/GitHub

**3. Architecture& Project structure:**

**3.1 Architectural Overview:**

The archtitecture follows the layered architecture style, where the application is divided into separate layers, each responsible for a specific concern or functionality.

Presentation Layer with the gui package, application controller layer with the controller package, persistance layer with the dao package,……

**3.2 Project structure:**

**1. org.fabianandiel.constants**

contains as status , role names and strings used throught the whole system.

**2. org.fabianandiel.context**

App-wide contexts for the loggedIn User and the person to update. See also React Context.

Allows sharing of values between different parts of application without parameter drilling from one level to the next.

**3. org.fabianandiel.controller**

Communicates with the DAOs.

**4. org.fabianandiel.dao**

Communicates with the database using JPA-Hibernate.

Provides all CURD operations and special queries.

**5. org.fabianandiel.entities**

Contains JPA entity entities that are tied to database tables.

Plus the entites are also annotated with Java Bean Validation for further validation and don’t have any effect on JPA-Hibernate

**6. org.fabianandiel.gui**

Here the FXML-based views get connected to the application core logic:

The controllers in this package handle UI Logic (updating views, text fields,…) and core logic. They communicate with controller layer for data fetching and operations.

**7. org.fabianandiel.interfaces**

Contains basic interfaces for controller and dao.

**8. org.fabianandiel.services**

Contains helper classes and functions.

**9. org.fabianandiel.validation**

Contains validation helper object for the user`s login request and a validation service for the employee form view.

1. **Installation & Configuration:**

**3.1.: Install the following:**

* Java JDK 17
* Maven
* PostgreSQL
* pgAdmin 4
* Git
* JavaFX Runtime SDK

**Step 2: Clone or Download the Project**

git clone https://github.com/FabianAndiel1985/employee\_management.git

or use the zip option

**Step 3: Set up the PostgreSQL Database**

Either create the database by using the full database dump from documentation/Datenbank/ Database\_dumps

or

Let the Hibernate in the project do the work for you

**Step 4: Configure the Database Connection**

Set up the database configuration in the persistence.xml.

Example JDBC URL:

<property name="jakarta.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/employee-administration"/>  
<property name="jakartapersistence.jdbc.user" value="postgres"/>  
<property name="jakarta.persistence.jdbc.password" value="postgres"/>  
<property name="jakarta.persistence.jdbc.driver" value="org.postgresql.Driver"/>

Also make sure to have the PostgreSQL Driver in your POM.xml:

<dependency>  
 <groupId>org.postgresql</groupId>  
 <artifactId>postgresql</artifactId>  
 <version>42.7.7</version>  
</dependency>

**Step 5: Check if you have all needed JavaFX Dependencies:**

<dependencies>

<dependency>

<groupId>org.openjfx</groupId>

<artifactId>javafx-controls</artifactId>

<version>21.0.7</version>

</dependency>

<dependency>

<groupId>org.openjfx</groupId>

<artifactId>javafx-fxml</artifactId>

<version>21.0.5</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.openjfx</groupId>

<artifactId>javafx-maven-plugin</artifactId>

<version>0.0.8</version>

<configuration>

<mainClass>${main.class}</mainClass>

<launcher>start</launcher>

<jlinkImageName>app</jlinkImageName>

<jlinkZipName>app-zip</jlinkZipName>

</configuration>

<executions>

<execution>

<id>default-cli</id>

</execution>

<execution>

<id>debug</id>

<configuration>

<options>

<option>-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=localhost:8000</option>

</options>

</configuration>

</execution>

<execution>

<id>ide-debug</id>

<configuration>

<options>

<option>-agentlib:jdwp=transport=dt\_socket,server=n,address=${jpda.address}</option>

</options>

</configuration>

</execution>

<execution>

<id>ide-profile</id>

<configuration>

<options>

<option>${profiler.jvmargs.arg1}</option>

<option>${profiler.jvmargs.arg2}</option>

<option>${profiler.jvmargs.arg3}</option>

<option>${profiler.jvmargs.arg4}</option>

<option>${profiler.jvmargs.arg5}</option>

</options>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

**</build>**

**Step 6: Run the Application**

# Introduction:

Our program is aimed at all levels of employees of the company.

No matter what you role you currently have, you start the program with the Log In screen, where you entere your username and password provided by Mrs.Michaels provided to you on your first day.

Then you enter the main screen where you see your current status up on the left and are supposed to clock in for your current work day under the button “time”.

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

Automatisch generierte Beschreibung

# Shared views across all levels of staff:

## Time view:

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

There you find the hours you should have worked up to today for this month under **“target hours”,**

Under **“actual hours”** you find the hours you book for the current month up to today.

Under “**difference”** you have the difference between the hours you have booked and the hours you are suppoded to have booked.

Once you have clocked in and clocked out it is a done deal. Please contact Mrs. Michaels if you need any changes for your bookings.

## Vacation screen:

As an employee in the vacation screen you can apply for your vacation. Then your vacation will be set to pending until your supervisor accepts it or denies it.

In the case of acceptance the vacation gets the status accepted and you’re ready to go.

In the case of denial please talk to the supervisor – be sure we will find a solution.

If your supervisor doesn’t answer the vacation in time it will get the status expired. Please give him/her a little nudge. We are all humans.

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

Automatisch generierte Beschreibung

**Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung**

As a manager or admin you can instantly book your vacation with out any supervision.

# Views for manager and admin:

You will notice that you have a couple of extra menu points with the before mentioned roles.

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

Automatisch generierte Beschreibung

## Request view:

As a manager you can approve od disapprove the vacation requests of your current subsidiaries.

**Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung**

## Employee overview view:

Under **“all employees”** you find all list of a employees of the company including the inactive ones.

Under **“updateable employees”** you find as a manager your subordinates AND employees without a superior yet.

As an admin you find all managers and other admins.

Both as managers and admins you can delete, update, view the master data of any non-inactive employee under this managed secetion. The important prerequisite is that you have to mark the row with the employee first to delete, update or view the master data.

**Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung**

### Creating an employee:

The button create new employee leads you to the employee form. Here you can create a new employee:

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Beware that some fields might be blocked out according to your current authorization level.

### Updating an employee:

This also leads to the employee form view, where the fields will be filled with the values of the employee.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

### Deleting an Employee:

This will give the employee the status inactive and you can not update them any longer.

If you delete manager all his subordinates will lose him as a superior too.

### The Master Data button:

Leads you to the MasterData View of the selected employee.

## Employee Form View:

Here you can update or create a new employee within the restrictions of your authorization.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

### Under the “create new address” Button:

You can create a new address then go back to the employee form and select the new address in the dropdown.

## Create new address view:

Here you can add a new address.

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

Automatisch generierte Beschreibung

## Master Data view of employee:

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

Automatisch generierte Beschreibung