Tour Planner

Software Engineering 2 SS2023

Lorenz Duelli, Jasmin Duvivié

## Architecture

We implemented a layered architecture for the Tour Planner as instructed in the SWEN course. The structure encompasses these layers:

* Frontend Layer
* Business Logic Layer
* Database Layer

Each layer only invokes methods of either their own layer or the layers beneath them. The data is passed on from layer to layer as annotated entities. These entities are mapped to the database with the Hibernate Object-Relational-Mapper and the Jakarta Persistence API.

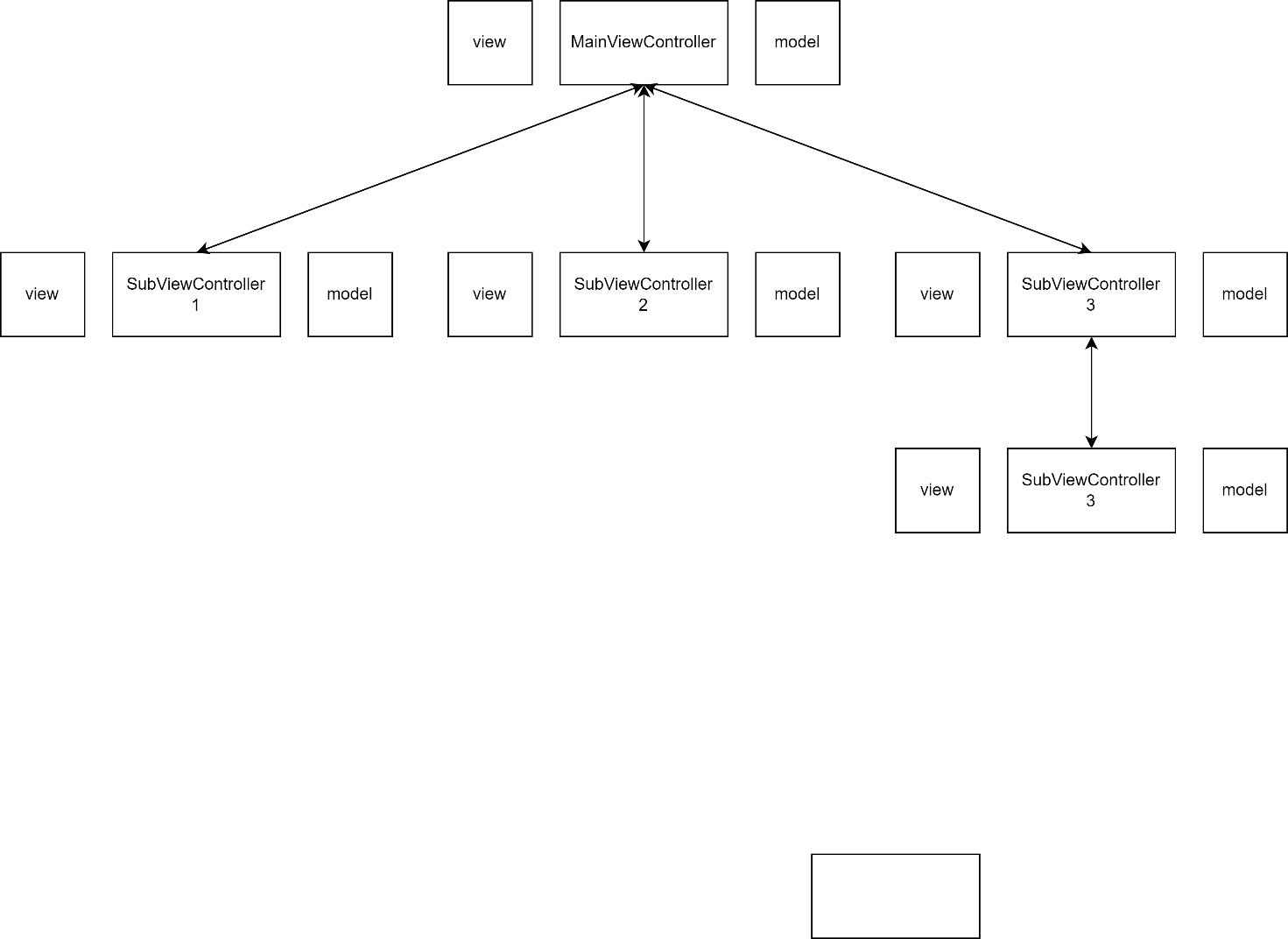


### Frontend Layer

The Frontend Layer is structured according to the MVVM-Principle:

* the user sees and interacts with the contents of the \*.fxml files (views)
* controllers (viewmodel) bind data between the views and their models
* the models (model) invoke the services of the business logic layer

Each controller binds between one \*.fxml and one model only. The MainViewController acts as a kind of parent, gets notified by the controllers when something has been updated and distributes the information to the other controllers, so that each controller only invokes their own model and is responsible only for their own tasks.



### Business Logic Layer

This layer carries out the functionality of the Tour Planner. In our application this layer is organized by services. These encompass requesting directions and images from the MapQuest API, handling image uploads, generating reports as well as importing and exporting tour data.

### Database Layer

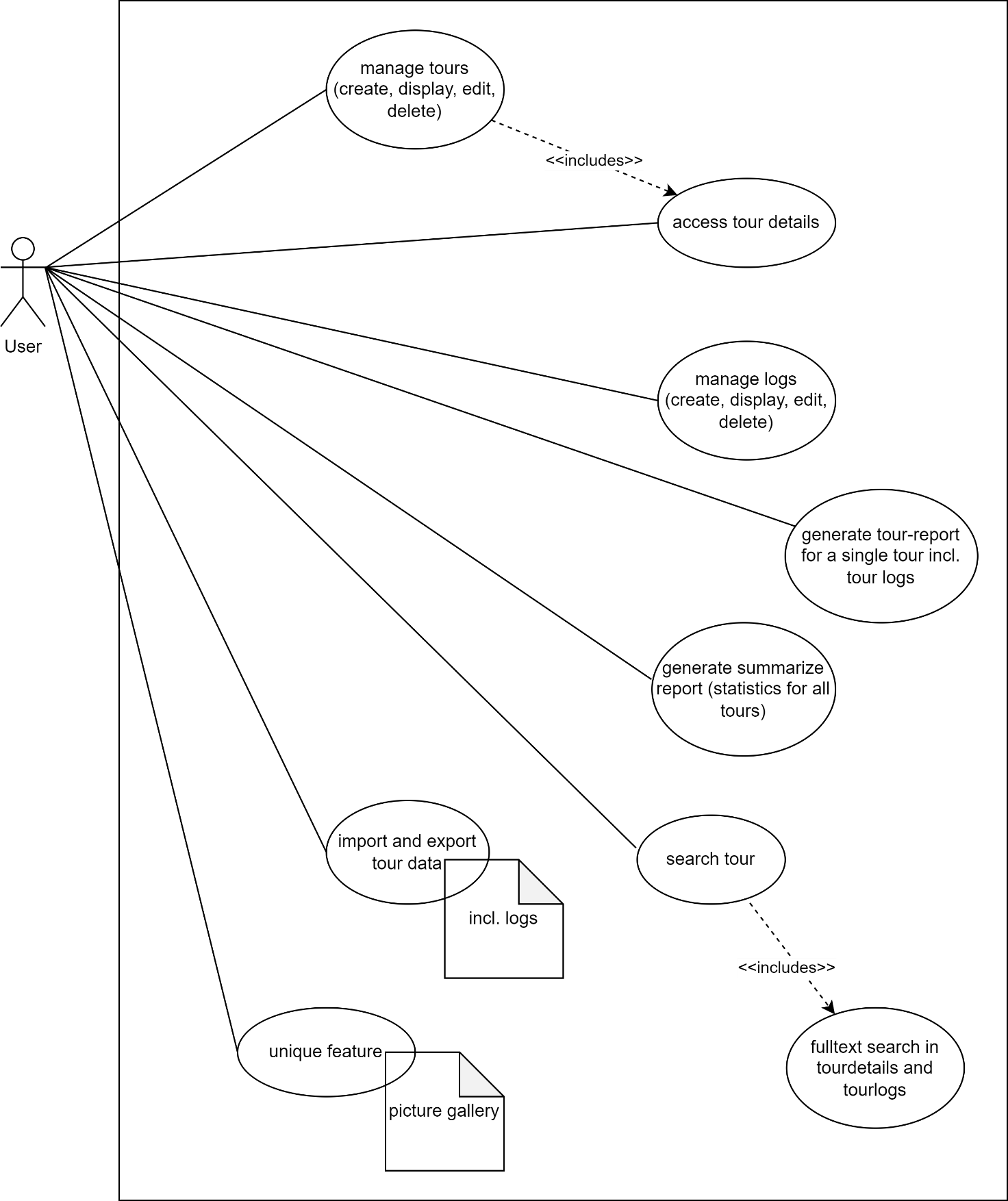
This layer is responsible for interacting with the database through DAOs (Data Access Objects).

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

Automatisch generierte Beschreibung

todo diagrams of the architecture

## Use Cases



## User Experience

We designed our application in a simplistic and classic looking way for the following reasons:

* we do not want the user to be overwhelmed by too much information. Thus, our information architecture is simple. Rather than bombarding the user with words, we use simple images such as s little disk icon for save and a little trash can for delete.

Ein Bild, das Zahl, Rechteck, Stecker enthält.

Automatisch generierte Beschreibung

* we do not want the user to be overwhelmed by flashy colours and effects. Thus, our design does not go far beyond simple and plain javafx.

Ein Bild, das Text, Karte, Screenshot, Software enthält.

Automatisch generierte Beschreibung

* None of us are particularly good designers so we could not be bothered to wrap our heads around fancy gimmicks and tricks that go beyond a spinner.

Ein Bild, das Screenshot, Kreis, Text, Design enthält.

Automatisch generierte Beschreibung

## Design Patterns

The following design patterns have been implemented:

* Factory Pattern (Controller Factory and Logger Factory)
* Publisher Subscriber Pattern (when a controller makes modifications which affect the state of the system, it publishes its changes. The MainViewController subscribes to these changes and invokes every other controller so that they carry out their necessary actions.

## Unit Tests

We mostly test the models and services, where applicable. The controllers simply bind between the \*fxml. files and the models and invoke methods of the models, so there is no need to test the controllers. The models have been tested where necessary i.e., where they carry out calculations to display the data which they receive from the business logic as they need it for the frontend. Most of the times however, the models only invoke services, in which case in makes sense to test the services but not the models. Thus, the services and models are most heavily tested.

## Unique Feature

As a unique feature we have implemented the functionality to upload pictures, which are associated with a tour. We know this kind of feature from e.g., Google Maps, where it helps people to get a concrete idea about the places they are going. Similarly, it is possible in our application to upload as many pictures as one likes to a specific tour to further support the usefulness of the Tour Planner.

Ein Bild, das Himmel, Text, Screenshot, Baum enthält.

Automatisch generierte Beschreibung

## Tracked Time

todo enter time

## Link to git

<https://github.com/smoothjass/swen2-project>

The repository is private, however the lecturer has been invited to collaborate and anyone interested is welcome to email us a request to:

[if21b145@technikum-wien.at](mailto:if21b145@technikum-wien.at)

[if21b144@technikum-wien.at](mailto:if21b144@technikum-wien.at)