FH University of Applied Sciences TECHNIKUM WIEN

Protocol

Software Engineering 2 Labor
BIF4-A1/A2 (Holzer Raoul)

Tour Planner

Aldin Zehinovic if22b130
Armin Dervisefendic if23b040
https://github.com/mide553/Tour-Planner.git

Project Overview

The Tour-Planner application is a WPF-based desktop application designed to help users plan and manage tours. It includes features such as creating, editing, and viewing tours, as well as logging tour details.

Project Structure

App.xaml / App.xaml.cs: Application entry point and global resources.
MainWindow.xaml / MainWindow.xaml.cs: Main application window.
Views/: Contains XAML views for the UI.
ViewModels/: Contains ViewModel classes for MVVM architecture.
Converters/: Value converters for data binding.
db/: Entity classes and database context for Entity Framework.
img/: Images used in the application.
bin/ and obj/: Build and temporary files.
Tour-Planner.sln: Solution file for the project.

Application Startup (App.xaml and App.xaml.cs):

The application is initialized using the Host class from the Microsoft.Extensions.Hosting namespace. Configuration is loaded from the appsettings.json file, ensuring that the application settings are properly loaded at startup. Dependency Injection (DI) is configured to register services such as TourPlannerDbContext, MainViewModel, and SearchViewModel. The application starts the host asynchronously, ensuring smooth startup and shutdown operations.

Main Window (MainWindow.xaml and MainWindow.xaml.cs):

The main window, MainWindow, is defined in XAML and its corresponding code-behind. The window's DataContext is set to an instance of MainViewModel, which is obtained through the DI container. The UI consists of a menu bar, a search bar, a data grid for displaying tours, buttons for CRUD (Create, Read, Update, Delete) operations, and a tab control to show detailed information for tours and logs.

View Models (MainViewModel.cs and SearchViewModel.cs):

MainViewModel is responsible for handling the primary application logic, such as loading tours and tour logs from the database. It also manages commands for adding, editing, and deleting tours and logs. Meanwhile, SearchViewModel handles the search functionality, binding the search text and command to the UI's search bar.

Database Context (TourPlannerDbContext.cs):

The TourPlannerDbContext class is the Entity Framework Core context that facilitates interaction with the database. It defines DbSet<Tour> and DbSet<TourLog>, which represent the tours and tour logs tables. The OnModelCreating method is used to configure entity mappings, ensuring the proper structure for the database.

Models (Tour.cs and TourLog.cs):

The Tour and TourLog classes serve as the data models for tours and tour logs, respectively. Both classes implement INotifyPropertyChanged to support data binding with the UI, allowing the UI to automatically update when the model data changes.

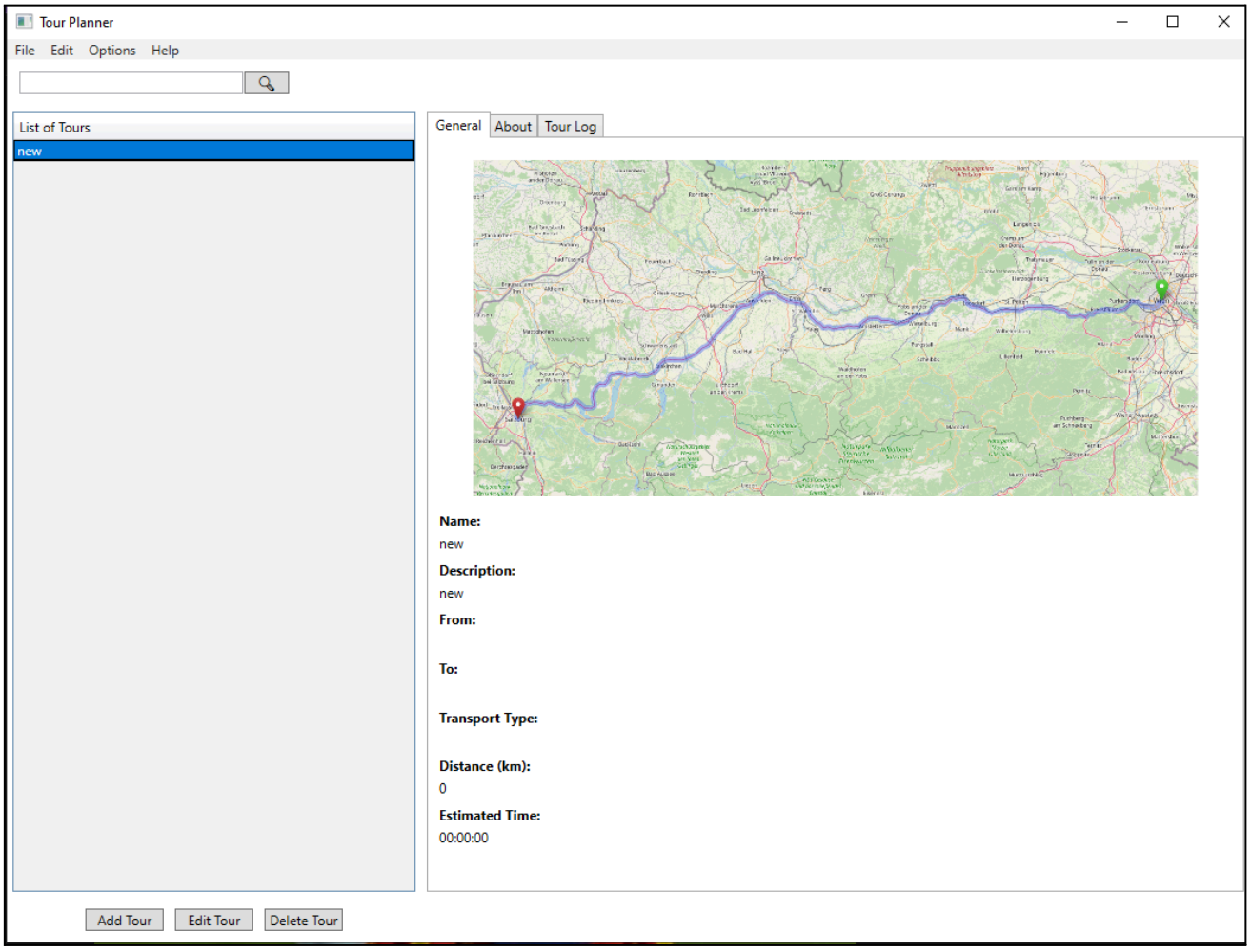
Converters (RelativeToAbsolutePathConverter.cs and StringToVisibilityConverter.cs):

The RelativeToAbsolutePathConverter is a utility that converts relative image paths to absolute file paths, enabling image loading from local directories. The StringToVisibilityConverter is used to convert strings into Visibility values, which are crucial for enabling or disabling UI elements such as buttons based on user interactions.

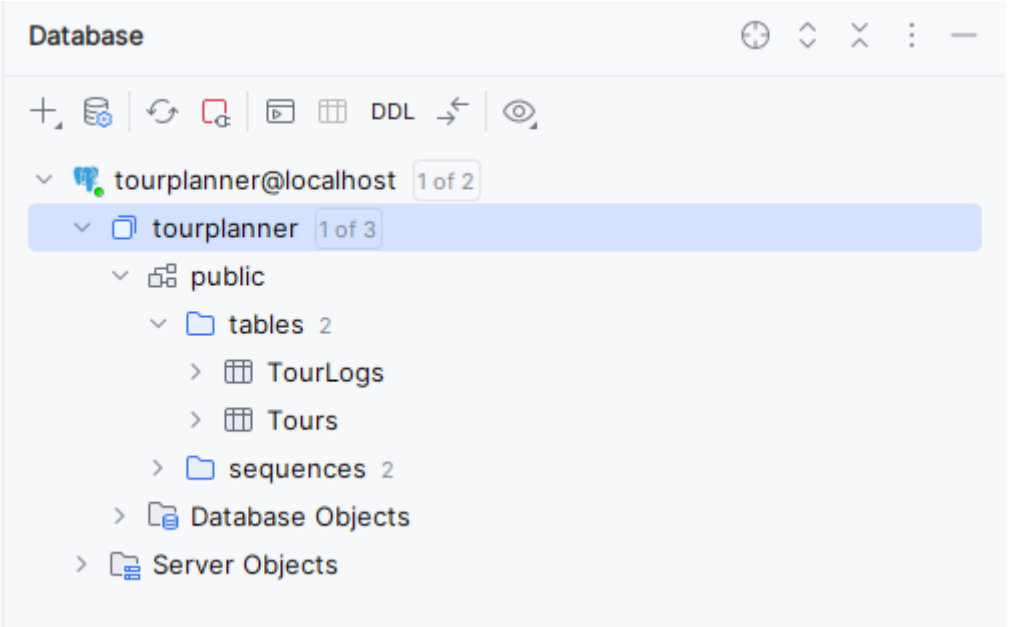
Views (AddTourWindow.xaml, AddTourWindow.xaml.cs, AddTourLogWindow.xaml, AddTourLogWindow.xaml.cs, SearchView.xaml, SearchView.xaml.cs):

The AddTourWindow and AddTourLogWindow are dialog windows that provide a user-friendly interface for adding or editing tours and tour logs. The SearchView is a user control that houses the search bar, allowing users to filter tours or logs based on search criteria.

Design


Tour Planner
File
Edit
Options
Help
List of Tours
new
General
About
Tour Log
Name:
new
Description:
new
From:
To:
Transport Type:
Distance (km):
0
Estimated Time:
00:00:00
Add Tour
Edit Tour
Delete Tour

Database Schema




```
CREATE TABLE IF NOT EXISTS "Tours" (
    "Id" SERIAL PRIMARY KEY,
    "Name" VARCHAR(255),
    "Description" TEXT,
    "ImagePath" VARCHAR(255),
    "From" VARCHAR(255),
    "To" VARCHAR(255),
    "TransportType" VARCHAR(100),
    "TourDistance" DOUBLE PRECISION,
    "EstimatedTime" INTERVAL,
    "RouteInformation" TEXT
);

CREATE TABLE "TourLogs" (
    "Id" SERIAL PRIMARY KEY,
    "Comment" TEXT,
    "Date" TIMESTAMP WITH TIME ZONE NOT NULL,
    "Difficulty" DOUBLE PRECISION,
    "Rating" DOUBLE PRECISION,
    "TotalDistance" DOUBLE PRECISION,
    "TotalTime" INTERVAL,
    "TourId" INT NOT NULL,
    FOREIGN KEY ("TourId") REFERENCES "Tours" ("Id")
);
```

How to Run the App

Prerequisites

1. **.NET 8.0**
2. **PostgreSQL Database**

Steps to Run the Application

1. **Start PostgreSQL Database with**
   Host=localhost;
   Port=5432;
   Database=tourplanner;
   Username=postgres;
   Password=password
2. **Run Queries to setup Tables**
3. **Pull the GitHub directory:** https://github.com/mide553/Tour-Planner.git
4. **Navigate to the Project's Solution**
5. **dotnet build**
6. **dotnet run**