Tour Planner - Protocol

Application Architecture

The program is structured to have 5 key layers

- The Presentation Layer
    - Handles the views and their corresponding viewmodels
    - Viewmodels are databound to the views and only communicate using messages
- The Business Layer
    - Holds all non UI classes
    - Static BusinessManager Class is responsible for all Communication between layers
    - Handles most tasks
    - Communicates with the Data Access Layers
- The Data Access Files Layer
    - Is responsible for handling all file access
- The Data Access Database Layer
    - Is responsible for all communication with the database
- The Data Access API Layer
    - Is responsible for all API communication and getting the map address

The Tours are handled by the program using the TourList class. This class contains a single property, a list of tours. The TourList is what is accessed by the BusinessManager and handles everything from adding new tours to the list and editing or deleting an existing tour, to searching through all tours for any matches to a search string.

An almost identical class exists for the logs, which is what is stored in each tour. This separates the handling of the tours logs from the tour class itself.

The handling of the UI changes within the datagrid of the TourDisplay view, uses a dictionary of string lists to identify which tabs are needed for a specific selection. This makes changes to the number and composition of each view type simple as the dictionary is the only element in the code behind that needs changing.

Implemented Design Patterns

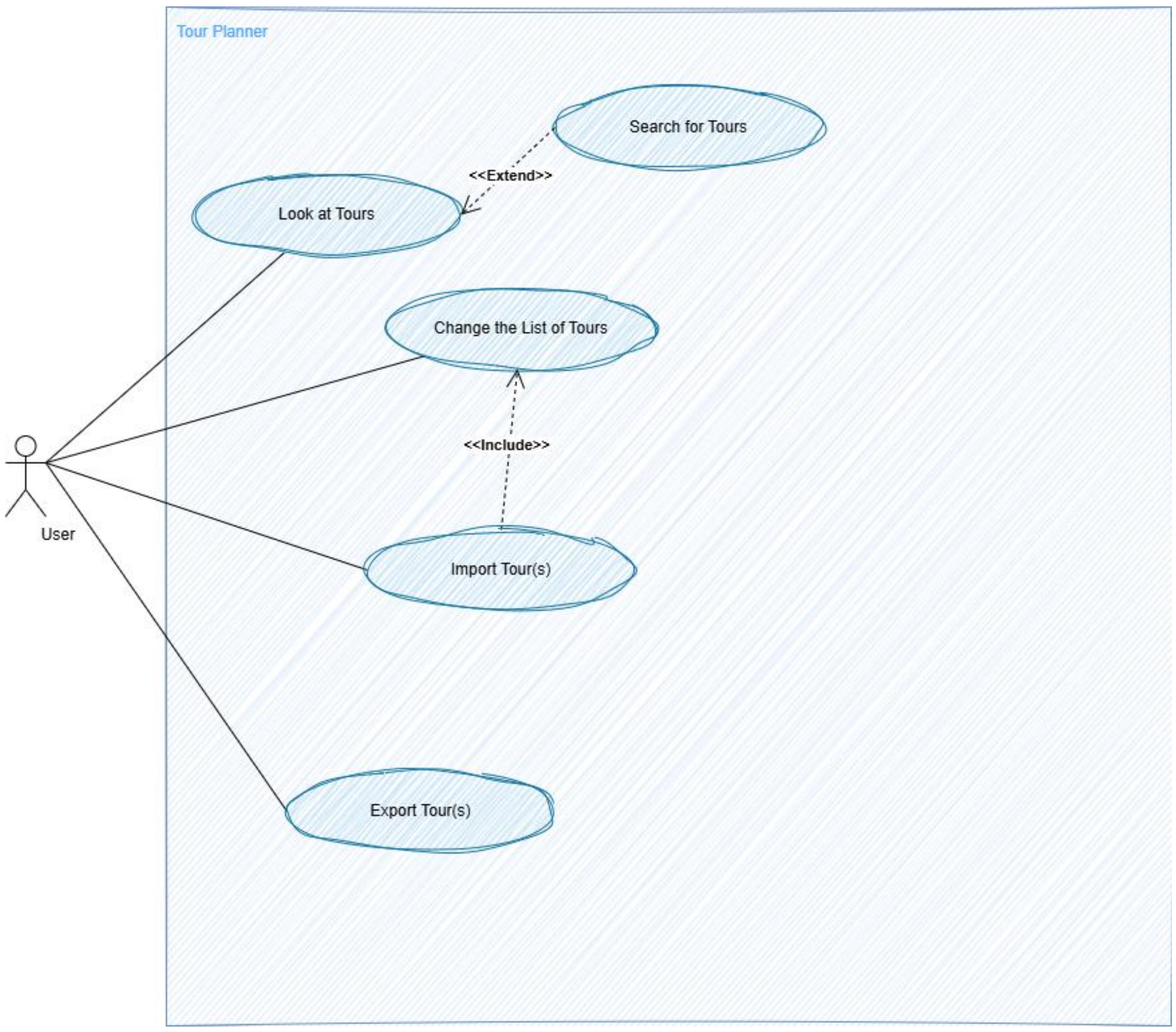
Command Pattern

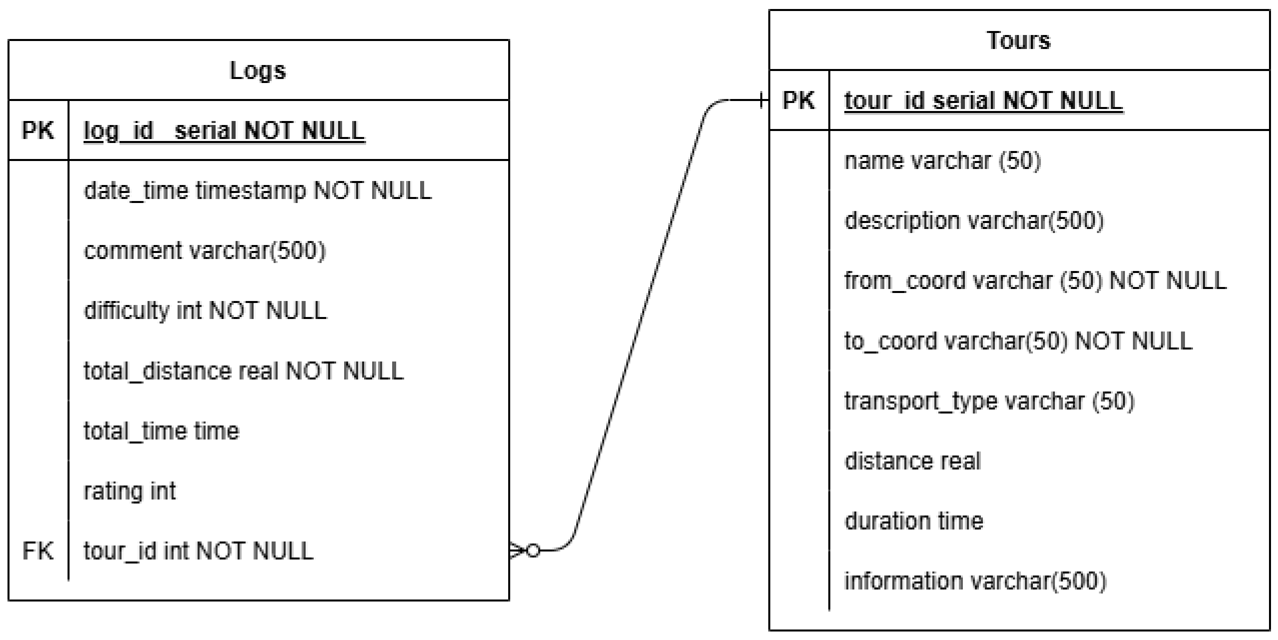
To bind the commands of various buttons to the viewmodels and adhering to MVVM principles, the RelayCommand of the mvvm.Light library was used to facilitate the implementation of a command pattern.

Observer Pattern

While no custom observer pattern was created, the mvvm.light messenger class was used to tell all viewmodels any time the currently selected tour or log was changed.

Use Cases & UX




Logs
PK log_id serial NOT NULL
date_time timestamp NOT NULL
comment varchar(500)
difficulty int NOT NULL
total_distance real NOT NULL
total_time time
rating int
FK tour_id int NOT NULL
Tours
PK tour_id serial NOT NULL
name varchar (50)
description varchar(500)
from_coord varchar (50) NOT NULL
to_coord varchar(50) NOT NULL
transport_type varchar (50)
distance real
duration time
information varchar(500)

Libraries Used

GalaSoft.MvvmLight.Command

Used for RelayCommand for Command Databinding

GalaSoft.MvvmLight.Message

Used to communicate between ViewModels

Used to inform a View of any necessary UI changes

log4net

Used for program logging

Npgsql.EntityFramework.Core

O/R Mapping and connection to the PostgreSQL database

iText7

Used for generating .pdf reports

Development Timeline

10. – 23. 3. (~3h/day) (Intermediate Hand-In)

21.6. – 6.7. (~5h/day) (Final Hand-In)

Unique Feature

The ability to choose between using the database or a local file to display tours.

Along with the ability to export/import the entire tourlist to/from a json file. The exporting works regardless of if the database or file is chosen as the data source.

Unit Tests

The majority of the unit tests, test the methods in the tour and log classes.

Due to the use of container classes for the handling of both logs and tours, these list classes were also extensively tested.

It was also tested if the methods within tour and log could be correctly accessed through the list classes. This line of testing was especially important as while EditLog & EditLog_ThroughTourList passed their tests just fine, EditLog_EnsureCorrectTour which tested the editing of a log from a specific tour within a list of multiple tours, did not pass it’s test initially.

A few other minor errors were also found via unit tests, such as the difficulty calculation simply summing the difficulties of the tours instead of calculating the average.

The rest of the unit tests related to the database.

GitHub

https://github.com/LeafyCloud01/TourPlanner

Lessons Learned

Learned the intricacies of data binding including commands. (And added an explanation of all necessary steps for the various types of databinding to a personal c# programming tips file)

Learned about and how to use the extremely usefull mvvm.light messenger class and how to enable file importing and exporting using the openFileDialogue class.

If there is no obvious reason for an error persisting, try restarting visual studio.

O/R mappers are much cleaner to work with than direct SQL queries.

API services sometimes go down, and there’s absolutely nothing you can do about it.