# TourPlanner

## Git Link

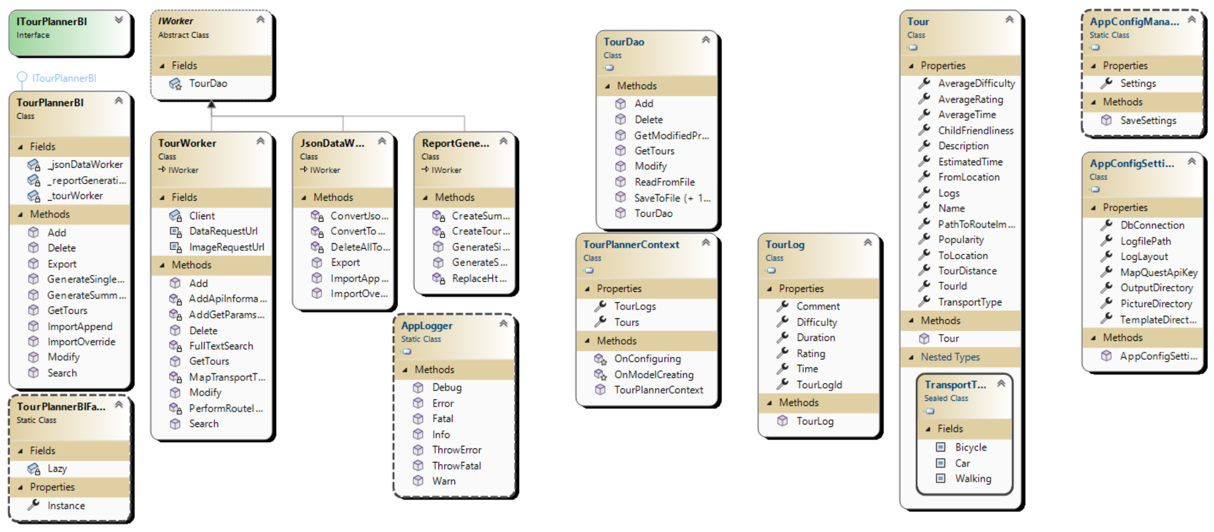
https://github.com/pinessap/TourPlanner

## Architecture Design

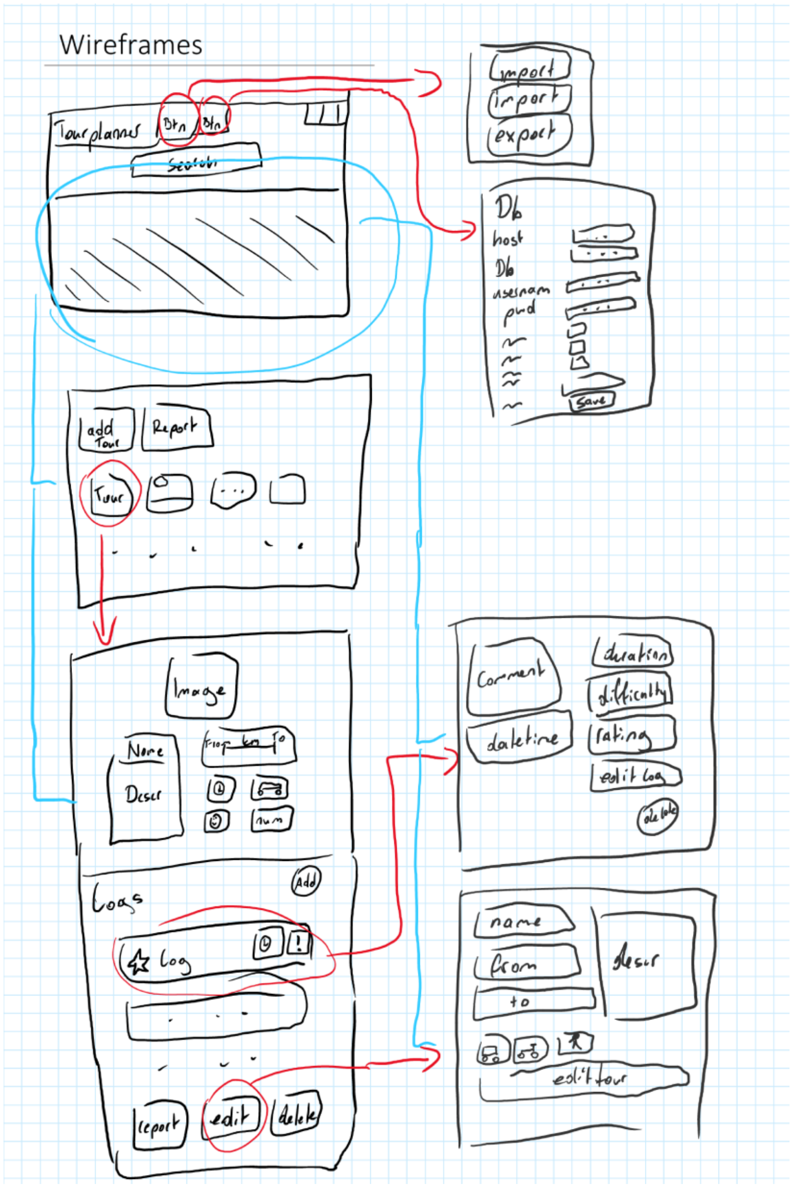
For the basic project architecture, a 3 layer model was chosen. PresentationLayer, BusinessLayer and DataAccessLayer are all represented via seperate projects in the application solution. In addition, the MVVM pattern was used to further seperate responsibilities. The PresentationLayer contains Views and ViewModels, and a seperate project models (since models are used throughout the entire application).

The BusinessLayer is reachable via the TourPlannerBlFactory class. The returned interface implementation contains all functions the BusinessLayer offers. A similar approach is used in the DataAccessLayer, where a single class is used to expose its functionality to the outside, while the interal project organization doesn't concern outsiders. All data has the same flow, going from PresentationLayer to BusinessLayer to DataAccessLayer and back up in reverse order. Therefore the PresentationLayer does not reference the DataAccessLayer and is completely independent from it.

Lastly, the Logging and ConfigurationManager classes exist as external projects, to enable full encapsulation and use in different projects. The singleton pattern and static classes are used to further simplify usage.

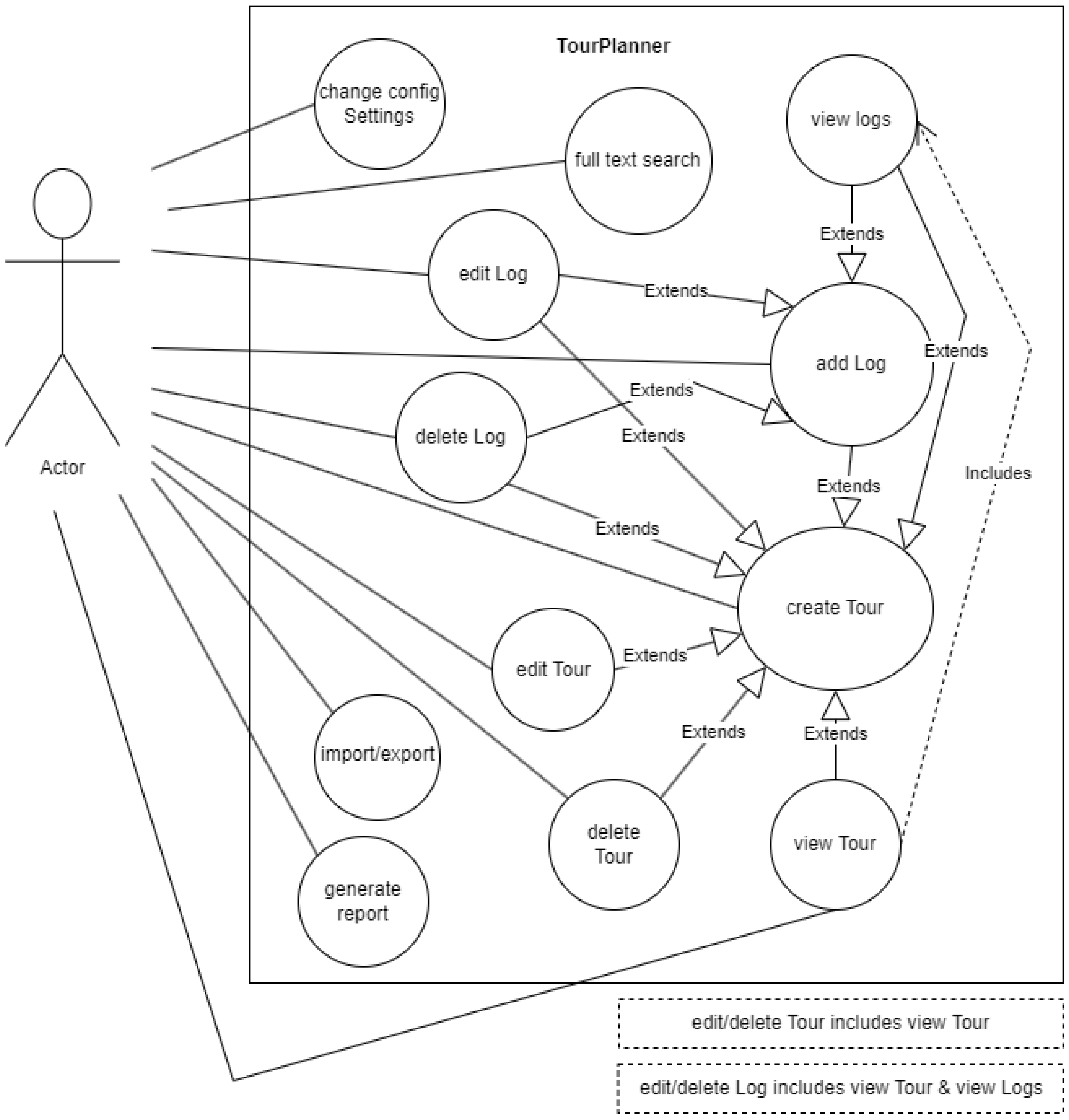


## Presentation Layer



Since we use the MVVM pattern, we have Views and ViewModels in our Presentation Layer. The View makes use of data binding to bind the properties of UI elements to the ViewModel. By using data binding, any changes in the ViewModel properties are automatically reflected in the corresponding UI elements, and vice versa.

In our Presentation Layer we have on MainViewWindow. It includes a ContentControl Element that switches between different Views. By updating the value of the CurrentView property in our MainViewWindow ViewModel, we can dynamically change the content displayed within the ContentControl. We make use of messages to switch views, since in our Application often one Usercontrol Component (and therefore the included View in our MainViewWindow) has to tell our MainViewWindow ViewModel to change the CurrentView Property to another View. Sometimes we also pass along parameters, like a Tour Instance of our Tour Object (e.g. when switching to the TourDetail Window, which displays all Information of a Selected Tour, we pass that Selected Tour along to the next ViewModel).



## Database

To start the database, make sure you follow these steps:

1. Clone the project.
2. Download binaries as zip from https://www.enterprisedb.com/download-postgresql-binaries
3. Unzip and Copy their content to the "postgre" folder.
4. You should now have 3 batch-files and a folder named "pgsql" inside the postgre folder.
5. Now you should be able to start the database by running the "start_db.cmd" file
6. After successfully starting the database, you can import all tables by running the "import_db.cmd" file while the database is running.

## Testing

The tests were written at the end of the project. This presented challenges, since we realized that we did not create a well-testable application. Therefore we had to resort to testing less critical parts and classes. The only classes that we could test without having to drastically change our architecture were the Tour, Configuration & data conversion classes. The tests we chose aim to make sure the model and especially its calculated properties work as expected, and all data is presented in a correct manner.

## Unique Feature

The unique feature we chose to implement was something that helped us during development a lot: A simple way to change the App.config file in the UI. Pressing the "Settings" button at the top of the Application window opens a dialog box that enables users to change different config settings and have them saved in the .config file(s). One thing to note here, the changed settings are reflected in the actual build .config file(s), so the App.config file located in the Configuration project will not be affected and remain unchanged!

## Lessons Learned

The first major lesson learned concern unit tests. Implementing them only after being done with the rest of the project was a mistake. We should have implemented unit tests continuously during development. This would have enabled us to create a testable project from the start, and not force us to restructure our architecture drastically.

Another big lesson concerning the Presentation Layer and Views was the use of reusable UI Components. We only had the realization that we could reuse certain Components towards the end of the project. It turns out that thinking about which Elements and Components will be used several times beforehand could save time and also prevent a lot of duplicate code. Since we only came to that realization towards the end, we did not manage to make use of the possibility of defining reusable UI Components, however we did manage to use one for the Display of Error Alerts.

Furthermore we learned that it's also worth thinking about the Design of your Application and the Presentation Layer beforehand to prevent having to change the Design and therefore Code later on.

## Time spent:

| date | time (h) | comment |
|---|---|---|
| 11/03/2023 | 3 | project and repo setup |
| 17/03/2023 | 7 | database setup |
| 14/05/2023 | 2 | tourLog model |
| 21/05/2023 | 1 | added data modifying |
| 21/05/2023 | 1 | added full text search |
| 22/05/2023 | 6 | UI modifications, added custom title bar |
| 24/05/2023 | 4 | json import/export |
| 28/05/2023 | 10 | single report generation |
| 29/05/2023 | 1 | calculated tour values |
| 29/05/2023 | 4 | summarized reports |
| 09/06/2023 | 5 | added config file |
| 10/06/2023 | 3 | reworked config system |
| 11/06/2023 | 5 | added logging |
| 11/06/2023 | 2 | added log messages |
| 12/06/2023 | 3 | improved error handling |
| 12/06/2023 | 2 | refactor report generation |
| 12/06/2023 | 2 | refactor file handling |
| 13/06/2023 | 3 | added tourdetails window |
| 14/06/2023 | 6 | modified views & viewmodels, added tourdetails view |
| 15/06/2023 | 6 | added addtour view |
| 16/06/2023 | 8 | modified tourdetails view, added addlog & edittour view |
| 18/06/2023 | 6 | modified edittour view; added editlog view, delete log & search |
| 19/06/2023 | 7 | added import/export and generate sum/single report to frontend |
| 20/06/2023 | 11 | added error alerts in frontend, UI modifications |
| 20/06/2023 | 8 | Added API calls |
| 21/06/2023 | 4 | Refactor BusinessLayer |
| 21/06/2023 | 4 | Added unit-tests |
| 21/06/2023 | 13 | UI modifications, added unique feauture (config settings), maps, User validation to frontend |
| 22/06/2023 | 3 | added documentation |
|  | **140** | **Total Time** |