Developers documentation Enigma Research 0.1

*Jan Kampherbeek, November 28, 2022

Enigma Research - introduction

Enigma Suite is a software suite,written in C#. The program is free and open source. This document provides information about the technical aspects for interested programmers.

Please read the User Manual for information about the functionality of Enigma Research.

License / open source

Enigma is Open Source. You can use it following the terms of the GNU General Public License (GPL).

The GPL allows you to use, change and redistribute this software only if your own software is open source. It does not have to be free, but the full source code should be available.

For more information see the file *gpl-3.0.txt* in the root of the source.

Enigma uses libraries from the Swiss Ephemeris (SE). For the SE additional conditions are in place. These conditions prohibit the use of the software unless it is open source and also free. If you want to charge money for a program using software from the SE, you need to buy a professional license from the SE. But doing so does not release from the conditions as set forth by the GPL: the code must be open source.

For more information see the file *se_license.htm* in the root of the source.

To use software from Enigma in your program, that program has to be open source. If you include the libraries from the SE, it also has to be free. Buying a license from the SE does not change the condition from the GPL that the software should remain open source. If you want to create software that is not open source you can use the libraries from the SE but you will need to buy a professional license, and you cannot use any code from Enigma.

Technical aspects

Technical environment

- Language: C# version 10.
- UI: WPF/XAML.
- .NET environment: .NET Core 6.
- Unit testing: NUnit.
- Mocking: Moq.
- Database: SQLite.
- IDE: Microsoft Visual Studio 2022.
- Dependency Injection: Microsoft Extensions Dependency Injection

Coding conventions

I will try to abide to the standards. For a definition check: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions

Using the Swiss Ephemeris

For astronomical calculations i will use the Swiss Ephemeris (SE). The SE consists of a set of data and a 64-bits dll: *swedll64.dll*.

To access the dll, the attribute [DllImport] is used. All imports from the dll are defined in facades. As an example for the definitions I used the file swissdelphi.pas that Pierre Fontaine and others created to access the same dll from Delphi.

Icons

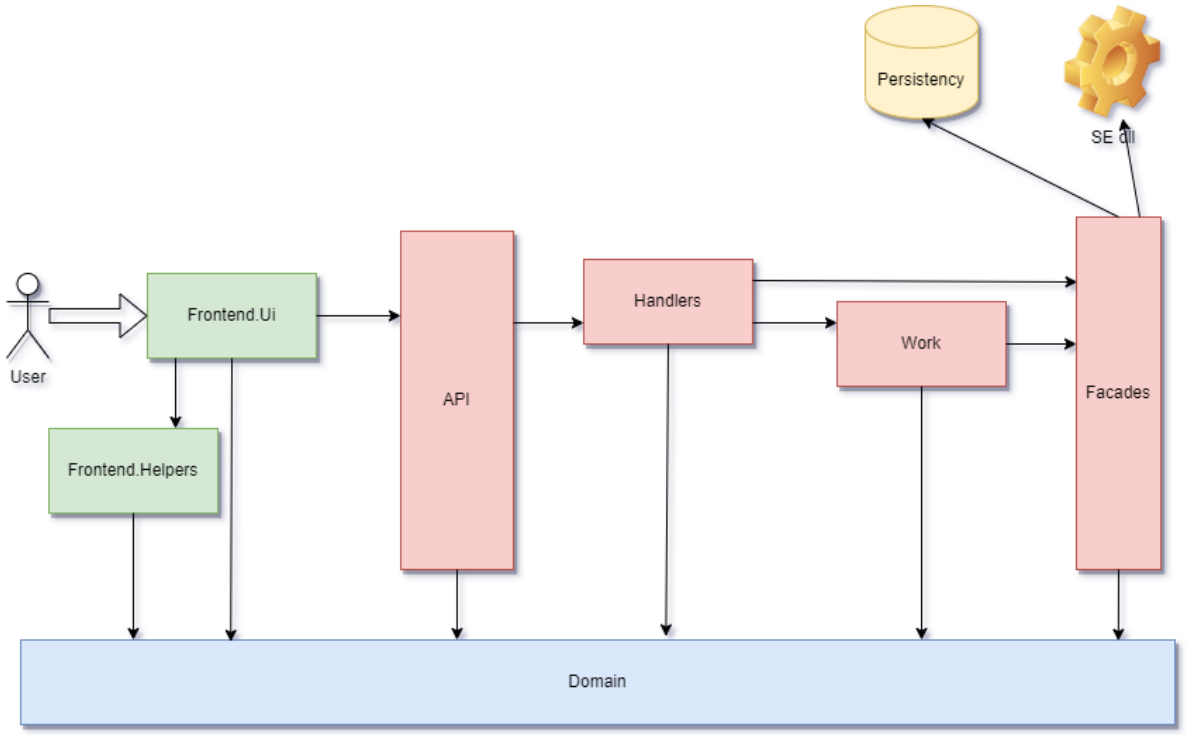
All icons in Enigma are from the icon set by Google, used for Material Design.

You can download the originals at https://fonts.google.com/icons

Projects

Enigma is build as a .NET solution that contains 8 projects. There is a separate project for unit testing, the other 7 projects contain the application. There is only limited communication between these 7 projects.

In the diagram you see three types of projects. In green projects for the user interface, in light red the core of the application and in blue a Domain project.



Frontend projects

There are two frontend projects, in the diagram recognizable by a green color.

Project Frontend.Ui

Frontend.Ui is activated as the application starts. It takes care of some initializing. Its main task is showing information to the user and receiving input from the user.

Frontend.UI can access *Frontend.Helpers*, *Domain* and *API*. No other project can access *Frontend.Ui*.

Project Frontend.Helpers

This project contains helper classes for handling the Frontend. *Frontend.Helpers* has access to *Domain*. It can only be accessed by *Frontend.UI*.

Core projects

The core projects handle the calculations, analyses etc. In the diagram they have a light red color.

Project API

The core projects are only accessible via the *API* project. Typically the classes in this project receive requests from the Frontend, perform some basic validation and pass the request to a Handler in the *Handlers* project. In most cases a response is returned to the Frontend. An API will never contain any business logic.

API can access the projects *Handlers* and *Domain*. It can only be accessed from *Frontend.Ui*.

Project Handlers

A Handler orchestrates the fullfilment of a request. Possibly it uses some basic business logic but in most cases it will rely on helper classes from the project *Work*. A handler is allowed to call other handlers. Sometimes it will simply pass through a request but it can also combine the results of several helper classes from the project *Work*.

Handlers can access *Work*, *Facades* and *Domain*. It can only be accessed by *API*.

Project Work

In *Work* you will find helper classes that can be used by handlers. Helper classes perform the real calculations, analyses etc. A helper class can use other helper classes but it cannot access a handler.

Work can access *Facades* and *Domain*. It can only be accessed by *Handlers*.

Project Facades

The project *Facades* contains classes that can access the outside world. A range of classes is used to access the dll from the Swiss Ephemeris. Other classes take care of persistency.

Facades can only access *Domain*, it can be accessed by *Handlers* and *Work*.

Domain project(s)

Currently there is only one domain project in the diagram it has a blue color.So all domain objects are accessible by all projects. A division into three projects: for the Frontend, the Core and a shared version is not implemented but probably will make sense in the future.

Project Domain

Domain contains all domain objects, including enums, DTO's and records. *Domain* cannot access other projects and is itself accessible by all projects.

Architecture

TODO

Astronomical aspects

The usual approach using the Swiss Ephemeris is followed but some specifics need to be mentioned.

School of Ram: hypothetical planets

The three hypothetical planets as proposed by the School of Ram, Persephone, Hermes and Demeter, are supported.

The calculations are based on the orbitual elements and calculated separately, without accessing the SE.

School of Ram: oblique longitude

The School of Ram supports a solution for the projection of the solar system bodies to the ecliptic. This solution ensures a proper placing of bodies in a house. However, the projection to the ecliptic is skewed. The solution is called 'true place' and also 'astrological place'. I prefer the more correct term 'oblique longitude'.

A dedicated calculation of this oblique longitude is implemented in Enigma. Some background information will become available. [TODO].

Generating documentation

Documentation is written in MarkDown (md) format. The result is exported to PDF for the user manual and to HTML (without styles) for the help files.

The user manual includes all the content of the helpfiles. The beginning and end of the texts that are used in the helpfiles are indicated with a standard xml-remark including the texts

`html-help-begin [name]` for the start of the part for the helpfile and

`html-help-end [name]` for the end of the part for the helpfile.

The name between the square brackets is replaced with the filename (without the .html part) of the help file.

The idea is to export the whole md-file to pdf for the user manual and also the whole file to html. The portions for the helpfiles can be copied to the help-files for Enigma.

The beginning and end of the helpfiles, including a reference to the styles used and a header in the format h1, should be added.