Assignment for Development teams

Distributed and Outsourced Software Engineering

Assignment 3 (Devs): Architecture and API Design

Deadline: November 11th -11 pm (CET)

1. Install EiffelStudio

The back-end will be implemented in Eiffel and we recommend you use the EiffelStudio IDE as the main development environment. All teams should install EiffelStudio because even the front-end team will need to run the server app during their development. Use the following instructions for installation:

https://code.google.com/p/dose2014/wiki/Resources

Make sure you are using the exact version as stated in the PDF. Otherwise, you might not be able to compile the code of others. Send questions about the installation to the course's student email-list:

se-dose-students@lists.inf.ethz.ch

Setup the EiffelStudio IDE as soon as possible. We cannot guarantee support related to installing problems after November7th.

2. Checkout the DOSE project

Each group will be working on a different project setup but located in the same repository. We recommend you checkout the folder of your group.

We provide a directory structure for the project that looks as follows:

```
implementations/group1/          -- source code of group 1
      …
implementations/group8/          -- source code of group 8
```

We also provide a demo application that shows how to use the Eiffel Web Framework (EWF) together with AngularJS (front-end) in:

```
implementations/demo/            -- simple Todo app for demonstration
```

You can use the demo app as a starting point for your own application. If you choose to do so, you can copy-paste the entire app into your group's folder.

Task: Checkout your group's folder and the demo folder (or checkout the entire repository). Open the demo (see the README.md file) in EiffelStudio and compile and run the project. Explore the implementation of front- and back-end. *Make sure to adjust the path to the "database file" and the "www" folder before running the Eiffel app.*

Important: since several students working on the same project, you should take extra care to make sure your code compiles before committing it to the repository. Also, write a log message for each commit!

4. Architecture and API design

As part of this assignment, your group needs to decide on the basic architecture of the app. We recommend an approach were the back-end provides a REST API, exposing data in JSON format. The front-end, written in a rich-client library, such as AngularJS or EmberJS, consumes the back-end's API and takes care of rendering the UI and displaying the data.

Task: Decide on the basic architecture of your app. The back-end must use EWF and should use SQLITE as a database. You have free choice of the front-end technology but we can only provide support for AngularJS.

4.1 Back-end

During this assignment the back-end team must develop several APIs:

- **Database schema**: create the tables and their relationships. Fill the table with test data to start development.
- **Database to object mapper**: an API that allows to retrieve data from the DB in a format that's suitable for further processing in your EWF application.
- **REST API**: the URIs of your server (e.g. /api/todos) and the respective HTTP methods; develop the APIs of the controllers that will handle the requests.

Task: The above-mentioned APIs must all be implemented in code. You do not need to develop documentation as part of this assignment even though it is often helpful to document the behavior your APIs (e.g. input-output formats) for your team members.

4.2 Front-end

The front-end team needs to develop the following:

- **Front-end routing**: design the URIs used in the front-end routing.
- **Controllers**: implement the APIs of the controllers that handle your app's logic.
- **Page mock-ups**: develop first mockups of the different pages (partial views, belonging to different routes) that your app will use.

Task: The above-mentioned APIs must all be implemented in code. You do not need to develop documentation as part of this assignment even though it is often helpful to document the behavior your APIs (e.g. input-output formats) for your team members.