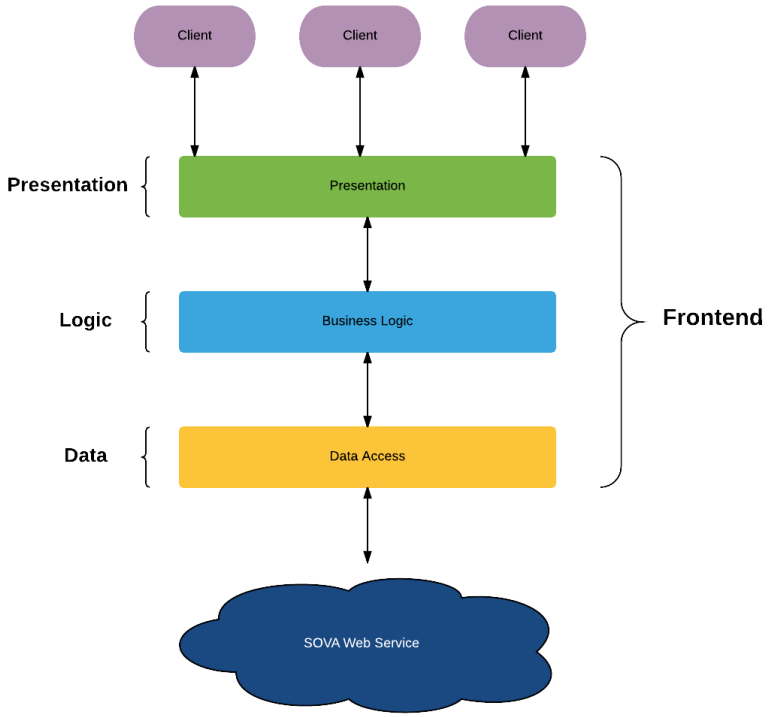
RAWDATA Portfolio-project Final Report

The Final Report of the RAWDATA Portfolio-project consist of two parts. Part 1 describes the last subproject, Portfolio-project 2. Part 2 is the reflective synopsis of the Project Portfolio. These two parts are described below. Also included below are instructions about form and submission of this report.

Part 1: RAWDATA Portfolio-project 3 description

(Please refer to the note: *RAWDATA Portfolio project* for a general introduction)

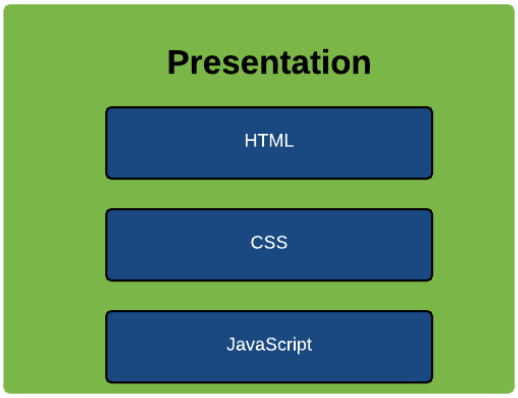
The goal of this portfolio subproject 3 is to provide a frontend to the SOVA (Stack Overflow viewer and annotator) application. The overall architecture of the frontend is sketched in the following figure:



The clients in the top of the above figure illustrate browsers used to access the application by users. The blue sky in the bottom is the backend developed in portfolio-projects 1 and 2. The three rectangles in the middle is the internal structure of the frontend to be developed in this project. Notice that the internal structure here is very similar to the internal structure of the backend from portfolio-project 2. The frontend is by itself a complex application and thus needs an architecture with decoupled parts organized into well-defined components having separated responsibilities. As in the portfolio-project 2, we want to favor maintainability, testability, extendibility, and scalability in the design.

Presentation

The presentation part of the frontend provides the responsive graphical user interface (GUI) to the SOVA application. It is a single page application (SPA) build with a modern web framework to deliver an intuitive and attractive experience for the users. The basis of the application is HTML5, CSS3 and JavaScript.

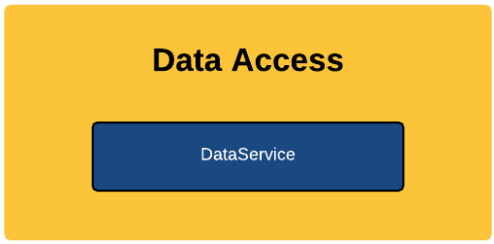


Bootstrap (or similar frame work) is applied to support the responsiveness and to promote an attractive UI and the JavaScript framework Knockout to connect the UI to the data models of the backend by use of the Model-View-ViewModel pattern, databinding and AJAX. Furthermore, RequireJS is used to support modularity of the application and jQuery is used for word clouds (and can also be used for AJAX).

Business Logic

The business logic in the frontend is primary presentation logic, and thus equal to the view model in the MVVM pattern. Even through there is only one page from a web user perspective, the application will consist of a number of different views, used to switch between different contents of the application. These views will properly have their own view models, that most likely will share some logic, which should be divided into separate modules.

Data Access



The data access part of the frontend should provide an interface to the backend. Instead of connecting directly with the backend web service in the presentation logic an abstraction, a data service, is created to define the internal interface to be used by the business and presentation parts. This way, changes to the backend interface can often be isolated to this part of the frontend.

A. Application design

Sketch a design of the application frontend. You should extend the backend web service such that it also handles the web application, i.e. the same web server serve both the backend by use of the path /api/… and the front end by use of the root / path. Since this is a single page

application you only need the default index.html page plus all the included CSS and JavaScripts.

A.1. Document the architecture of your "frontend" system.
A.2. Use one of the following responsive multi-device layout patterns and argue for your choice.
   a. Mostly Fluid
   b. Column Drop
   c. Layout Shifter
A.3. Show sketches of the different contents (pages) of your application. Show how Bootstrap grid system is used.

B. The Data Access

Create one or more data services that wrap the communication to the backend web service.

B.1. The service(s) must be implemented as JavaScript module using the revealing module pattern.
B.2. The service(s) exposes a set of functions to get access to the data from the web service by use of asynchrony. [Hint: use callbacks]

C. Business Logic

The logic of the frontend is primary related to the view models used to support the MVVM pattern. You must use knockout.js to implement the MVVM.

C.1. A view model must be associated to each view.
C.2. Knockout components must be used to compose the frontend application.
C.3. RequireJS must be used to support modularity, i.e. divide the solution into isolated parts, such as knockout components and services, and load the parts via RequireJS.
C.4. Databinding should be used to relate the view models to UI components.
C.5. The navigation must be controlled with databinding.

D. Presentation

The presentation must provide the responsive UI to the SOVA application. Use the mobile-first principle, i.e. start with the smallest devices and add features. You must implement the UI with Bootstrap. The presentation must be developed such that it appears coherent with respect to the layout of forms, lists, tables, etc.

D.1. The presentation must be developed such that SOVA becomes a SPA.
D.2. The presentation must use navigation (navbar) to switch content.
D.3. Content with listed information must be provided with pagination when needed.
D.4. Forms must use the form-group/form-control style (Bootstrap or like) and should provide examples on validation.
D.5. The frontend must include visualizations, e.g. word clouds.
D.6. [OPTIONAL] Theme your presentation layout to create your own expression.
D.7. [OPTIONAL] Add advanced controls, e.g. dropdowns, modals, alerts.

E. Mobile [OPTIONAL]

Transform your frontend application into a mobile app. Use e.g. Xamarin[1], PhoneGab[2] or similar service to make hybrid apps for IPhone and Android phones.

F. Deployment [OPTIONAL]

Deploy your application. We provide a server, rawdata-app.ruc.dk, to which you can deploy your application. You can find information on how to do this here: https://flexlab.ruc.dk/blog/rawdata-app-ruc-dk-publish-your-content/ (be aware that you may need to run another version of dotnet than the one we have used in the course due to limitations on the rawdata-app.ruc.dk server).

Part 2: Reflective Synopsis

The reflective synopsis is supposed to summarize what is covered in the four portfolio subprojects, to relate practical experience with relevant theory and to discuss and present reflections about these aspects and about the combined project as a whole. In addition, the synopsis should include one or more topic-chapters that each cover a relevant topic, describe in brief relevant theory / background for the topic and explain how it relates to the portfolio project. Finally, the synopsis should provide an overall conclusion and comments on future work.

Reflective Synopsis, content (suggested)

1. Introduction
2. Portfolio project summary
3. One or more topic-chapters that each
    - cover a relevant topic,
    - describe in brief relevant theory / background,
    - explain how it relates to the portfolio project,
    - describe what you have done in the project that relates to this topic
4. Portfolio project discussion
    - discuss and present reflections that relate to the four portfolio subprojects and to the combined project as a whole.
5. Portfolio project conclusion
    - concluding remarks and future work

Each group can choose to either include a single topic chapter as a joint contribution from the group or to include one topic chapter for each individual group member. In size the topic chapter(s) should be around 0.5 – 2 normal pages per group member (e.g. a 3-person group can include either one 1.5 – 6 pages chapter or three 0.5 – 2 pages chapters).

The Final Project Report

You are supposed to work in groups and submit your final Portfolio project report (in pdf format) through **eksamen.ruc.dk**. In addition, as far as part 1 concerns, your final database should be running on the course server **rawdata.ruc.dk** and your Visual Studio solution, including projects for each part(layer) must be committed to **GitHub** (or similar resource) with the Section3 tag. Be sure to include an URL to your source repository as well as the connect information to your MySQL database in the report.

[1] https://www.xamarin.com
[2] http://phonegap.com/

The Final Report should include a front page with **Title** and **full names** of **all members.** Part 1 of your report should in size be around 5-10 normal-pages[3] excluding appendices, while part 2 is expected to be around 6-15, 8-20 and 10-25 pages for respectively 2-3, 4 and 5-persons groups. Spend the number of pages that you find reasonable – don't stretch your writings to reach the maximum. The submission deadline for the Final Report as well as the portfolio 3 product (database & source code) is 17/12-2019 at 12:00.

[3] A normal-page corresponds to 2400 characters (including spaces). Images and figures are not counted.