UNIVERSITE LIBRE DE BRUXELLES
Ecole Polytechnique

INFO-H511: Web Services

Project Assignment

1 Assignment

The project consists of conceiving, designing, and implementing a web service back-end (and, where needed, front-end). The project must be completed in groups of 2. If there is an odd number of persons following the course, one group of 3 persons will be allowed. (See Section 4).

The project consists of 3 phases:

1. In the first phase, you will propose a topic for your web service. You are free to propose any topic, provided that it meets the minimum requirements discussed below. In case you are not able to suggest your own topic (or in case that the proposed topics does not meet the minimum requirements), you can choose one of the topics included at the end of this document.
2. After being approved by the teaching staff, you will do the initial design of the web service in the second phase. In particular, you will choose what web services technology stack (RESTful or Big-WS*) you will follow, as well as the programming language that you will use, the programming framework (if any), and the libraries that you expect to use.
3. Finally, you will do the detailed design & implementation of the web service, and document your experience in a report.

Your topic proposal should detail:

1. the members of your project group;
2. a title for your web service;
3. the general objective of your web service;
4. a list of usage scenarios that illustrate the functionalities to be provided by the web service, as well as the kind of data that it manipulates.

At the very minimum, your proposal and web service must meet the following requirements:

- it must have a non-trivial data model (e.g., in a social news aggregator service, *posts*, *tags* and *users* altogether are considered to be non-trivial, but *posts* alone are considered to be trivial);
- it must support the addition and updating of at least part of the information items that it manages;
- it must support at least 3 representations of its data (e.g. JSON, XML, and RDF);
- it must use at least one other web service (e.g., by adding information to Google calendar, Twitter, Google maps, importing contacts from an online address book, . . .);

- it must take security into account (e.g., to ensure that only authenticated users can access the service, and that any given user can only access his own data). The security should be taken seriously, using established standards.
- whatever your choice of technology stack (e.g. REST instead of Big-WS*), you should implement the web service using the development guidelines/advice that was viewed in the theory lectures/seminars.

The above items are the minimum requirements of your web service; extra credit can be earned by providing extra functionalities (e.g., a web front-end, additional interactions between services, implementation deployed on Heroku or other hosting platform, ...).

You are free to use existing third party libraries during development.

2 Report

In addition to the code produced, you must submit a report of 15 pages *maximum* (excluding title page). This report must describe the architecture of your solution. It must describe the technologies you have selected to implement your service, as well as motivate the reasoning behind your choice of technologies, especially between SOAP and REST. It must also contain a high-level documentation of the API (underscoring the additional features). You should also document what you have learned as you have built the web service (what worked well? what didn't? what would you do differently next time?). Here is a suggested structure for the report.

For the second phase, you need in to hand a preliminary report that contains only sections 1 and 2.

1. Introduction and requirements.
   Briefly summarize the project goals. Introduce the topic of the web service you have chosen. Illustrate the requirements of the web service by means of the usage scenarios that were originally to be supported.
2. Analysis.
   (a) Requirements Analysis.
   Analyze the requirements; list for each requirement what technology(s) would be suitable; identify your preferred choice; motivate your analysis and your choice of preference.
   (b) Summary of chosen technologies and platforms. *Summarize your choice of technology stack (REST or Big-WS*)? Also explain what programming language. framework(s), and/or libraries you will be using, and why you have made this choice.*
3. Service design and implementation.
   (a) Functional Analysis.
   Analyze how the different usage scenarios from (1) can/should be supported by your chosen set of technologies.
   (b) Implementation & API
   Detail how you have actually implemented these functionalities, describe the associated API (if you use Big-WS you could describe the different services available and the kind of API followed; give/refer the WSDL file. if you use the REST approach you can give an overview of the resources; the supported operations; the possible formats used; etc.*

4. Observations on your development.
   What went well? What did not work as expected? What should have been done differently? What would you do differently if you had to re-do the project? How would you do it differently?

5. Conclusions.
   Reflect on the project. What have you learned?

3 Project defense and demonstration plan.

You will get the opportunity to demonstrate and present your web service and defend your choices of implementation during an oral presentation session. In preparation of this defense, you must submit, in addition to your code and report, a demonstration plan. This plan must be described in *a separate document* and detail the operations that you plan to follow during the defense to show that your web service is indeed functioning. In particular, the demonstration must illustrate the exchange of raw messages between a client and the web service. The demonstration must also include a showcase of the additional functionalities.

4 Modalities

- This assignment contributes 10/20 to the overall grade. A project implementing the minimum requirements and the report are quoted on 8/10 points in total. Additional functionality is quoted on the remaining 2/10 points.

- This assignment can be combined with that of another course, provided that the teaching staff of that course agrees. In that case, you must include in your project proposal: (1) a clear statement that the project is combined with the project of another course; (2) the name of this other course; (3) a copy of the assignment of the other project; and (4) a clear description of the work that is exclusive to your web service project (and which is hence not evaluated in the scope of the other project).

- The assignment should be solved in groups of 2 (if we have an odd number, one group of 3 students will be allowed). You are asked to send, per group, the names of the group members to Mr. Stefan Eppe (`stefan.eppe@ulb.ac.be`) by February 26 at the latest. If you cannot find a partner, please indicate so by sending an email to Mr. Eppe, who will hook you up with a partner.

- You need to create a git repository[1] in the `INFO-H-511` repository group at `http://wit-projects.ulb.ac.be/rhodecode` to submit your project proposal, interim report, code, demonstration plan, and final report. The username and password to login to this system correspond to your ULB/VUB NetID. The repository must be named

  ```
  project-<student1>-<student2>
  ```

  where `student1` and `student2` corresponds to your respective usernames (in alphabetical order, based on your last names). It is recommended that you create this repository *as soon as possible* and use it as a version control system to avoid last minute technical difficulties.

  Tip. If you attempt to push a large set of changes to a GIT repository with HTTP or HTTPS, you may get an error message such as `error: RPC failed; result=22,`

[1] `http://git-scm.com/documentation`

`HTTP code = 411.` This is caused by a GIT configuration default which limits certain HTTP operations to 1 megabyte. To change this limit run within your local repository

```
git config http.postBuffer *bytes*
```

where `*bytes*` is the maximum number of bytes permitted.

- Your project proposal must be pushed to your repository **no later than Monday 3 March 2014**. You will receive notification of approval or disapproval of your proposed topic by Monday 10 March 2014.
- The interim report (containing sections 1 and 2 of the suggested report structure outlined in Section 2) must be pushed to your repository **no later than Monday 31 March 2014**.
- Your code, demonstration plan, and report must be pushed to your repository **no later than Wednesday 14 May 2013**. The demonstration plan and report must be submitted in pdf or html format. You will be asked to demonstrate and explain your web service during the examination (date to be determined).

Important dates

- **February 26** group submission
- **March 3** project proposal submission
- **March 10** notification of acceptance for project proposals
- **March 31** interim report phase 2 due
- **May 14** project deadline

Suggested Topic 1: Social news aggregator

This web service will support the aggregation of RSS feeds and allow social interaction on news items, in the spirit of e.g., Google Reader `http://reader.google.com/`, FeedEachOther `http://feedeachother.com`, and Good Noows `http://goodnoows.com`.

The web service is intended to support the following scenarios:

- John logs into the service, and adds a few RSS feeds by providing their URL. He decides to subscribe to “Hacker News”, and “IEEE Spectrum”.
- John then sees the latest news from all his subscriptions, in their order of publication.
- Jane logs into the service. She opens the “XKCD What If?” feed - to which she had subscribed - and sees a list of the latest news from this feed.
- John opens a news on “Nanoscale Vacuums Speed Semiconductors”. He gets the content of the news and a link to the original source of that news.
- John decides to tag this news item as “Technology”. Later on, he decides to further add the “Electronics” tag.
- Jane then opens the list of articles tagged “Electronics” and sees every article that she had previously tagged as “Electronics”. In particular, John’s tagged news on “Nanoscale Vacuums Speed Semiconductors” does *not* show up, because his tags are distinct from Jane’s tags.
- John shares the news on “Nanoscale Vacuums Speed Semiconductors” with Jane.
- Jane opens the list of news shared by John and sees the news he just shared.

These scenarios are examples of how the web service is to be used; the web service itself will provide a reasonably complete API (e.g. a user will also be able to remove a feed from his subscriptions, and it will be possible to add more than 2 tags).

Suggested Topic 2: Flashcard memorisation

This web service will support flashcard encoding and memorisation for its users, in the spirit of e.g., Memrise `http://www.memrise.com` and iKnow `http://iknow.jp`.

The web service is intended to support the following scenarios:

- John logs into the service, and browses the existing set of flashcards, available to all users.
- John opens the flashcard deck "Dutch vocabulary" and is presented with a list of the cards. One of the card reads "Tree" on the front side, and "Boom" on the back side.
- John subscribes to the "Dutch vocabulary" deck, and starts a study session of 10 cards. During the session, the system first shows the card "Tree"/"Boom". Later on, the system displays "Tree" and let John pick the right choice between "Appel", "Wagen", "Boom", and "Huis". These choices correspond to the back of different flashcards from the same deck.
- After finishing the study session, John checks his statistics. He sees that his average of good responses for "Dutch vocabulary" is 8 out of 10, and that his average of good responses for "Advanced English" is 7.3 out of 10. He also sees a chart with the number of cards reviewed for "Advanced English"; that number has grown steadily over the past month.
- John shares the number of "Advanced English" cards he has studied on Twitter. His tweet reads: "I have studied 234 of the 500 cards of Advanced English! #study".
- Jane creates a new deck title "Countries and Capital Cities". After adding 15 cards, she decides to share her deck with all users.
- Jane adds a comment on the "Dutch vocabulary" deck, saying that there is a mistake in the spelling of the card "Aircraft"/"Vliegtugi".

These scenarios are examples of how the web service is to be used; the web service itself will provide a reasonably complete API (e.g. a user will also be able to remove a deck he created, and it will be possible to add more than one comment). The charting on the statistics page will be realized through Google Chart Tools.

Suggested Topic 3: Online poll service

This web service will support the creation and participation to online polls, in the spirit of e.g., Doodle (http://www.doodle.com) and Polls (http://polls.cc).

The web service is intended to support the following scenarios:

- John logs into the service, and creates a new poll *"What shall we eat tonight"*, listing as possible choices *Pasta*, *Pizza*, and *Hamburger*. He adds the email addresses of his housemates Jane and Adam so that they are automatically requested to fill in the poll.
- Jane receives an email containing the address of the poll; and fills in that she prefers pizza.
- Adam receives an email containing the address of the poll; and proposes *French fries* as an extra possibility. He also sees that Jane chose *Pizza.*
- John and Jane receive an email notifying them of the change to the poll proposed by Adam.
- Jane adds a comment to the poll that she definitely does not want to eat *French fries.*
- John fills in his own preference, and closes the poll. Adam and Jane are notified of the poll result. The majority voted for *Pizza.*
- Other users that did not receive the poll invitation email are not able to access the poll.
- After closing the poll, John gets a map of the Pizza places nearby.

These scenarios are examples of how the web service is to be used; the web service itself will provide a reasonably complete API (e.g. a user will also be able to cancel a poll, and it will be possible to change preferences).