Meeto: Collaboration and Social Networking

Distributed Systems 2014/2015

Deadline V2: December 5, 2014, 11:59pm

1 Objectives of the assignment

At the end of this practical project, you should have:

- Developed a Web interface for the Meeto application.
- Integrated the Web application with legacy code from your project's V1.
- Mastered Struts2, Java Servlets, JSPs, and JavaBeans.
- Followed the MVC pattern to build Web applications.
- Learned to use Websockets to asynchronously communicate with the client.
- Integrated your application with third-party REST service providers.

2 Overview

In the second stage of this project, you will create a new web frontend for your application. This will allow users to access your application from almost any internet-connected device on the planet, without the need to install specific client software. Since interoperability is a very important requirement, users on the web should have access to the same information as users on the desktop application. In order to achieve this, your web application will connected to the DataServer previously developed, using JavaRMI.

Users should have the same capabilities, regardless of the interface they use. So your web application should provide ways to plan meetings, manage agendas and register action items, key decisions. In order for the real-time chat to continue working, you will need to integrate WebSockets in your project. Users nowadays are demanding and tricks like meta-refresh and hidden iframes are no longer acceptable.

Finally, web applications no longer exist isolated from each other. By leveraging OAuth and REST APIs, you will integrate your application with a 3rd party service. In case of events, Google Calendar is an example of an important key player in event managing on the web.

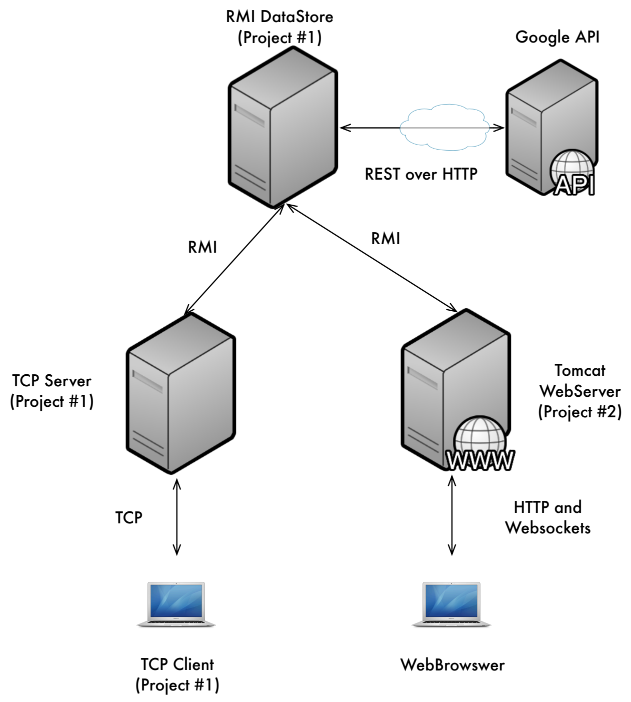

Fig. 1: The Webserver should integrate with the DataStore via RMI. The communication with the Browser should be done via HTTP and via WebSockets.

3 Architecture

Figure 1 shows the intended architecture for your project. You should implement a web application running on top of an HTTP Web Server (Apache Tomcat) that acts as a RMI client to the DataStore. Integrations with an external REST API is implemented at the RMI server or, alternatively, at the Webserver.

Clients will use browsers to connect to your web application, using HTTP to request webpages. You may improve the user experience of your application by carefully selecting requests to be performed via AJAX instead of refreshing the whole page.

Real-time communication to the browser should be implemented using websockets. This includes not only the chat, but also notifications like meeting invitations and action items.

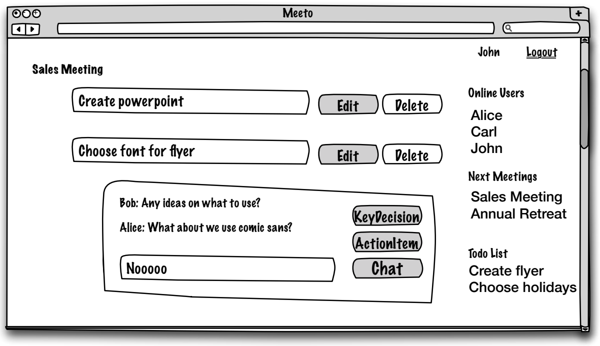

Fig. 2: Example of a possible web-interface

4 Web Interface

Using a MVC architecture, you should implement the following functional requirements using the Struts2 framework. Please refer to the slides used during the classes (Slides #15) and solve the Struts2 practical exercises before you start.

1. Register user
2. Securely login user
3. Create a new meeting
4. Invite users to meeting
5. Accept meeting invitations
6. See meeting overview
7. Add/modify/delete agenda item
8. Add Key decision to agenda item
9. Assign Action Item to user
10. (Groups of 3 only) Create Group
11. (Groups of 3 only) Add and remove user from groups
12. (Groups of 3 only) Invite group members to meetings
13. (Groups of 3 only) Assign Action Items through groups.

Figure 2 shows an example of a user interface for the basic features in this project. You should try to make your interface as usable as possible.

5 Real-time Notifications

In order for your application to feel responsive, you should leverage the power of WebSockets to immediately push information to the client. You should use it to improve experience in the following topics:

14. Chat inside each action item
15. Meeting invitation in real-time
16. Action Item assignment in real-time

6 Google Calendar Integration

Google Calendar provides a REST API for developers to integrate their own apps by reading, adding, editing and deleting events and calendars.

In order to guarantee the correct permissions and access to data, the Google API is protected using OAuth2, a standard for allowing 3rd party developers to use data with the user's consent. For this purpose you may use an API to facilitate the usage of tokens and token authentication. Note that you may not use Java APIs that perform all the HTTP requests for you.

For this part of the project, you should

17. Connect a user account with a google account
18. Login with a google account instead of regular credentials
19. When a new meeting is created, it should be added to all the users with a google account associated
20. Each time a user updates the agenda item, the new agenda should be reflected on google calendar
21. Each time a user accepts or declines the event in Google Calendar, that information should be reflected on your application.

- Introduction
- Web-based architecture
- Integration with Project 1
- WebSocket integration
- REST web service integration
- Users manual
- Tests specification

7 What you will learn

With this project you will acquire practical competences in the following topics:

- Introduction
- Web-based architecture
- Integration with Project 1
- WebSocket integration
- REST web service integration
- Users manual
- Tests specification

8 Submitting your assignment

Deliver everything in one ZIP file. This file should include a `readme` file with *all the necessary information* to execute and test the assignment without the presence of the students. Assignments that do not contain this README with all the necessary instructions will not be evaluated. Assignments that do not execute correctly will also not be evaluated.

Inside the ZIP file, there should also be a PDF containing your report. Do follow the suggested structure, as all sections will be evaluated.

You should submit the ZIP on the inforestudante platform until October 24, at 11:59pm: `http://inforestudante.uc.pt`