Recruiting Platform Prototype 1 Report

Liao Hu, Omar Thenmalai

Department of Electrical Engineering

Cooper Union for the Advancement of Science and Art

New York, NY, United States

Prof. Ethan Lusterman

03/01/2020

Contents

1 Introduction

In this project we will create a online recruiting platform that allows companies to post open job positions and applicants to browse and apply for positions. The first prototype contains an in-memory back-end system of the platform. To build this system, we chose to use the Spring framework along with Maven as the project management tool. With no user interface implemented, we demonstrate the functionality using curl command line tool, which tests the API of our locally running back-end system.

2 Project Description

In our system, there are three abstract classes and four modules. The three classes: User, Job, and Application, represent three different types of objects that appear in our system. The four modules: model, service, store, and handler, constructed the web service of the back-end system.

The User class manages all user accounts, both applicants and companies, as well as an admin account, which will be implemented. Each User object has six attributes: firstName, lastName, accountType, email, password, and a automatically generated userId.

The Job class represents all open and closed jobs. Each Job object has seven attributes: salary, company, title, location, experienceLevel, jobStatus, and an automatically generated jobId.

The Application class, similar to the Job class, represents all applications. Each application object has four attributes: userId, jobId, applicationStatus, and an automatically generated applicationId. There are three statuses for every application: pending, rejected, and accepted.

The model module defines the three classes above; the service module handles the business logic for each request, broken up into UserService, JobService, and ApplicationService. The store module will deal with connections to the various databases, broken up into a UserStore, JobStore, and ApplicationStore. Each store will connect to its own database, and handle queries and writes. The handler module handles and delegates all network requests with designated responses. Each handler, the JobHandler, UserHandler, and ApplicationHandler, represent a different URI path, and handle specific requests along those paths.

2.1 General Structure

For this project, we propose the general structure as shown in Figure 1. User's operations are divided into two categories: operations from an applicant and operations from a company. Based on a given user's account type, that user may only perform specific operation with respect to Jobs and Applications. For example, an applicant user may only view jobs and create new applications. By contrast, a company user may only view applications and create new jobs.

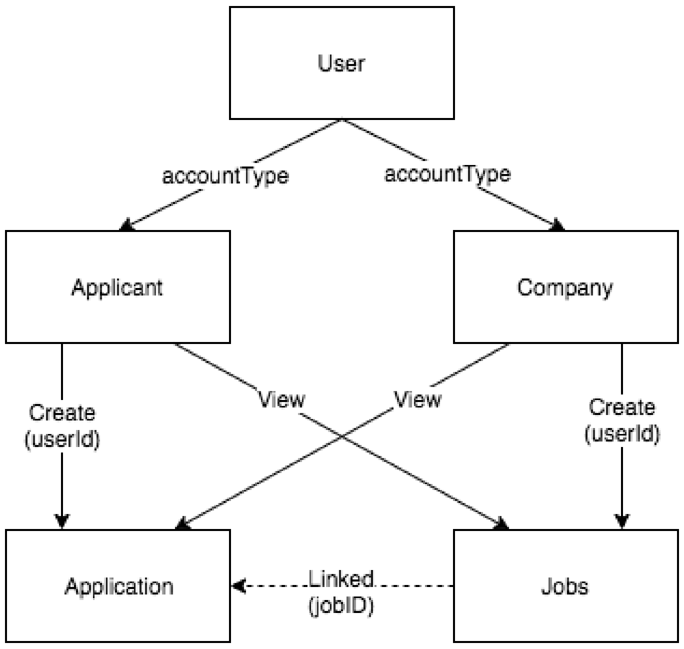

Figure 1: Project Structure

2.2 General Pipeline

To accomplish the restriction above, we proposed a general pipeline for every operation as shown in Figure 2. Every request will be first checked by a Account Type Filter to examine whether the user has the authority to perform that request. Next, a qualified request will go through a Parameter Validation Filter to check whether each parameter in the request is valid or not. Once these two filters have been passed through, the request is serviced by the business logic in the Service layer. The result of a given request is then passed back to the Handler layer in the form of an internally-defined status code. This status code is interpreted by the given handler, which then returns the required response.

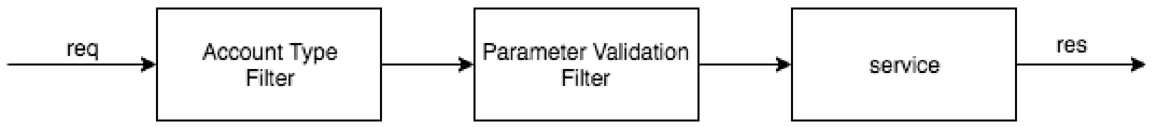

Figure 2: Project Pipeline

3 Team Dynamic

Aside from the hiccup earlier this week, things have been going well teamwork-wise. We encountered some issues early on setting up the github repository and familiarizing ourselves with Spring, but they were resolved. Both of us had similar ideas, with regards to project structure, making easy for us to create the initial structure and begin working. The work was evenly divided, and we were able to work individually once the overall structure was determined. As for the challenges, since we both think similarly, it has been difficult to reflect on our design decisions. An example where this hurt us so far has been in the decision to treat both companies and applicants as the same, rather than creating two separate classes and handlers. While our structure has been sufficient for the requirements of this step, we are discussing refactoring to allow for certain fields or features that may be exclusive to a given type of user.

4 Future Improvement

At this stage of the project, we haven't implement necessary security features. For example, user's password and other important information are not encrypted. The user model need to be reconstructed as discussed in the section 3 above, as we plan to not only add more account types such as administrator but also implement exclusive functions for different types of user.

Furthermore, we plan to make our code more "Spring" style, as it has an unique working pipeline called "Bean" that we haven't use at all.