

JobPortal (JP) Analysis

Lufthansa Industry Solutions

An analysis of the architecture and good practices used on the front-end job application project.

Timelapse: 26.10.2022 – 29.10.2022 (4 days).

Author: Gerald Nika.

Languages, technologies, and IDE-used:

- HTML5, CSS3, TypeScript, JavaScript;
- Angular CLI 14.2.6;
- Node 16.18.0, NPM 8.19.2;
- Bootstrap 5.2;
- RxJS, JSON, Firebase;
- Visual Studio Code 1.72.

Architecture used:

- ✓ Components;
- ✓ Nested Routes;
- ✓ Models (interfaces for the Firebase collections used);
- ✓ Guards for authentication and authorization (Role Based Access Control);
- ✓ Services (contains all logic for authentication and CRUD functionality operations).

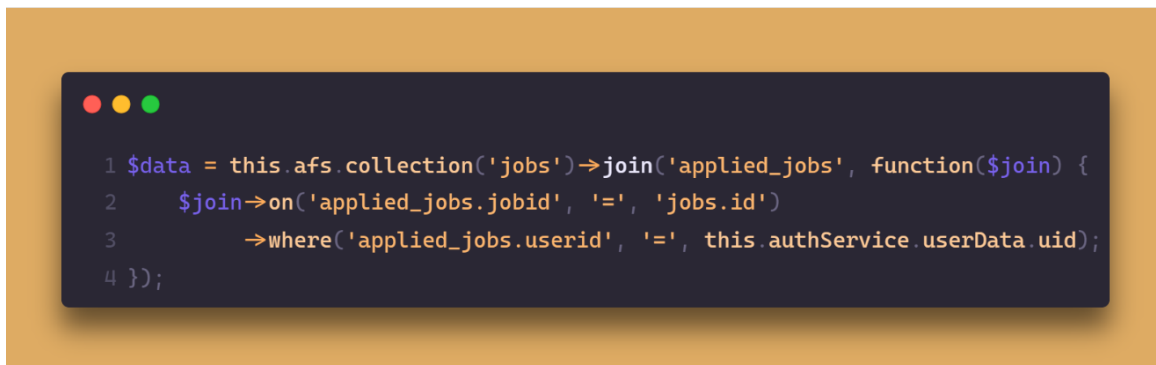
Good practices used:

- Delegating complex logic to services (The Single Responsibility Principle);
- Sequence of component members;
- Separating component class, CSS, and template;
- Prefixing component selectors;
- Use of Angular CLI for the development of components, routes, services, models, and so on;
- Use of environment variables;
- Maintain proper folder structure;
- Safe strings declaration;
- Descriptive file and class names;
- Comments for important lines of code;
- Use of interfaces.

Difficulties faced during the development process:

1. Since Firebase was a new platform for me, it took me some time to wrap my head around the way how its' logic is set up and how it uses NoSQL Database to build queries for storing and synchronizing data in real time. After going through some online forums (included in the references below), I finally got my hands on it and started developing the project.
2. The second difficulty I faced was when I was trying to develop the “Apply for the job” functionality. I tackled it in two ways.

The first way I created a separate collection, named it “applied_jobs” and it held two fields: “userid” and “jobid”. So every time an application was made, the ID of the user who applied and the job ID were stored. This way I thought it would be really practical to fetch firstly all job ID-s that were associated with the current user’s ID, then join with the query to get all jobs using these job ID-s. But, I could not achieve this approach, because it became really difficult to build complex compound queries using Firebase, even though I had managed to develop it using SQL and PHP. Check the code snippet below:



```
1 $data = this.afs.collection('jobs')->join('applied_jobs', function($join) {
2     $join->on('applied_jobs.jobid', '=', 'jobs.id')
3     ->where('applied_jobs.userid', '=', this.authService.userData.uid);
4 });
```

Credentials to test the application:

Seekers:

- 1) nikageri01@gmail.com
- 2) marianmara@gmail.com

Recruiters:

- 1) andignika@gmail.com
- 2) gerardotatzati@gmail.com

(All four user accounts' passwords are "123456").

References:

Angular + Firebase + Typescript — Step by step tutorial

<https://medium.com/factory-mind/angular-firebase-typescript-step-by-step-tutorial-2ef887fc7d71>

Firebase With Angular CRUD Operations

<https://medium.com/nerd-for-tech/firebase-with-angular-crud-operations-ba91ddf1dcc4>

Gerald Nika.

