Collaborative Code Editor

Yazan Abdullah

Contents

Chapter 1

Introduction

In this report, I will outline my solution to the final project in Atypon's Training.

Part I

Backend

Chapter 2

Outline

In this part, I will outline my implementation of the backend in the project.

- Docker Client/Docker Server: for code execution.
- MongoDB: for persisting data.
- Version Control Classes: set of classes and methods on top of the database to track changes in projects.
- Websockets/STOMP: for real-time data synchronization.
- Spring Security: for user authentication and authorization.
- Spring Web: for operations over HTTP.

Chapter 3

Code Execution: Docker Remote API

I used docker containers to safely fun users' code. I allowed for maximal flexibility, by cloning users' entire projects to the container, allowing them to structure their projects and run them like they would in a local environment.

To interact with docker, I used Docker remote rest API over the web. This is to allow me to move the docker server to a dedicated machine for performance and security purposes.

I designed a DockerClient class, that uses a WebClient object to issue commands and receive results from the docker server.

Each method in my docker client is responsible for an operation; it receives a configuration object holding the different parameters needed for the method.

Configuration objects can be huge in terms of the number of parameters, this is why I used the builder design pattern with them. I provided a directive that will further ease preparing the configuration objects.

Chapter 4

Real-Time Collaboration: Websockets & Conflict Resolution

hello

Chapter 5

Database: MongoDB

Chapter 6

Version Control

1. Layer on top of the database 2. stable/active project 3. merge conflicts: needleman-wucnsch

Chapter 7

Spring Security

Part II

Frontend

Chapter 8

React with Typescript

To implement the front end, I chose to use React.js with Typescript.

This helped me make the website more interactive, and allowed me to reuse components.

Typescript helped me find bugs by strong typechecking.

To style the website, I used Material UI component library.

For the editor, I used monaco editor.

I used Axios and Tanstack for http requests.

I used react-stomp-hooks to interactive with websockts.

Chapter 9

Styling the Frontend: MUI

Chapter 10

Code Editor: Monaco Editor