Приложение «word_counter»
Инструкция по запуску

1 Системные требования

Для работы приложения необходимы:

- Java 11+
- Tomcat 9+
- Maven 3+

2 Установка и запуск бэкэнда

Для установки и запуска приложения необходимо:

1. Скачать и распаковать архив на диск;
2. Запустить консоль Windows в папке с приложением;
3. Запустить «ex-1.0.0.jar« из папки target, для чего в консоли (Windows) набрать:
   java –jar target\ex-1.0.0.jar

   Должна появиться фирменная надпись “Spring” и пройти загрузка приложения. После старта должна выйти надпись «…Started WordCounterApplicationTests in … seconds (JVM running for …)»
4. В браузере, в строке адреса ввести:
   localhost:8080/swagger-ui.html
   Будет осуществлен переход к Swagger UI где можно протестировать API.

3 Остановка бэкэнда

Для остановки приложения можно воспользоваться любым менеджером процессов.

В Windows можно использовать стандартный Диспетчер задач, вкладка «Процессы». Там необходимо отключить процессы java.

4 Запуск фронтэнда

Для запуска фронтэнда необходимо открыть браузером файл index.html из папки ***X*** \ *exercise**src**main**resources**static*, где ***X*** – адрес папки с проектом на диске.

Желательно использовать Chome или Firefox последних версий, т.к. используются такие возможности HTML5, как input data.

5* Сопроводительное письмо. Описание приложения

Используется следующий технологический стек:

- Каркас приложения (MVC), разработка, безопасность, работа с базой данных - Spring Boot 2.3.3 (devtools, web, security, data JPA);
- Сборка приложения – Maven;
- База данных – H2, в режиме in-memory *(исключительно в целях упрощения передачи тестового задания*);
- ORM, Validator – Hibernate;
- Контейнер сервлетов, CDI – Tomcat
- Система контроля версий – Git
- Фронт – HTML5, CSS3, JS (ECMAScript6).

Пару слов о проекте:

Приложение состоит из клиентской части, реализованной на JavaScript, и серверной части, реализованной на Java 11 + Spring Boot 2.3.3.

Серверная часть представляет собой открытый RESTful API.

API описан на Swagger'е.

База данных хранится в ОЗУ и очищается при остановке приложения. Такой выбор был сделан для упрощения разворачивания приложения при передаче.

Для работы с БД используется ORM-подход и Hibernate + Validator, для удобства работы, безопасности и расширяемости приложения в будущем (например, для перехода на БД Postgres).

Так как откликался я все-таки на позицию Junior *Backend* ☺, основной упор был сделан на серверную обработку данных: после каждого изменения поля входных данных фронт запрашивает с сервера полностью всю таблицу и все точки для графиков.

Это не совсем рационально с точки зрения реального приложения, т.к. фильтрацией данных в таблице и перестроением графика логично заниматься на фронте. Однако я реализовал весь этот функционал в бэкенде, фронт только получает и обновляет данные.

Что не удалось: стилизация и кроссбраузерность отдельно не прорабатывалась ☺ JS скрипты также были по кусочкам надерганы со всех уголков интернета. График «мертвый», не функционирует. Но бэкэнд точки для построения графиков выдает. А вообще я не очень в JS и верстку, разобраться с графиком не хватило времени из-за завала по основной работе ((По той же причине не стал писать юнит-тесты.

С уважением и надеждой на обратную связь, Дмитрий!