Тестовое задание для стажировки

База данных

В качестве базы данных для приложения используется PostgreSQL. Развертывается локально на компьютере, в моем случае для подключения использовался url
jdbc:postgresql://localhost:5432/postgres

При подключении используется запись admin с паролем admin, для которой необходимо обозначить

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin;
```

в консоли pg.

В бд содержится одна таблица «socks», в которой 3 поля: color text not null, cottonpart integer not null, quantity bigint
Схема бд:

```
create table socks
(
    color      text    not null,
    cottonpart integer not null,
    quantity   bigint
);
```

В ней, при добавлении новых записей через веб приложение, будут храниться все возможные комбинации пар носков с полями color/cottonPart,
Каждый раз при попытке изменить количество пар носков тех или иных параметров, веб приложение проверяет, существует ли в бд запись с такими параметрами, если да, то изменяет их quantity, в противном случае добавляет новую запись. В случае когда при изменении quantity у записи оно становится не положительным, запись удаляется.

Веб приложение

Веб приложение написано на java с использованием фреймворка Spring boot.
Запускается через public class Application { }, в методе main(). Также в нем происходит инициализация объекта специального класса для работы с бд через внутреннюю статическую переменную INSTANCE для сохранения Connection connection.
Все возможные запросы описаны в public class Controller { }, в частности income(), outcome(), которые принимают на вход JSON объекты

и socks(), который принимает на вход url запрос и извлекает из него параметры. Все эти методы возвращают html код результата, а socks() также возвращает результат в теле этого результата. Также прописана обработка всех возможных ошибок с последующим возвращением html кода.
Каждый из вышеперечисленных методов обращается к базе данных через public class DataBaseController { }
Краткое описание методов:

1) public ResponseEntity<String> changeSocksCount
    Проверяет, существуют ли уже пары носков с параметрами на складе и соответсвенно изменяет их количество на складе, вне зависимости от поступления/отпуск. Удаляет запись при отрицательном quantity.
2) public ResponseEntity<String> countSocks
    Получает все записи из бд, подходящих по параметрам из запроса и считает сумму всех их quantity
3) private ResultSet getSocksTableByParams
    Возвращает таблицу записей из бд по параметрам для предыдущего метода
4) private Object sendStatement
    Отправляет запрос в бд
5) public static DataBaseController getInstance
    Возвращает статический объект этого класса

Класс параметров пар носков

В public class SocksData описаны все параметры и количество пар носков при поступлении/отпуске для сериализации из json в методах income/outcome в Controller. Причем в последнем варианте вызывается метод setOutcome(), который меняет значение quantity на отрицательное для сокращения кода при отправке запросов.

Тесты

Также описан класс Test в котором представлены метода для полного тестирования функционала отправки запросов, в том числе с json объектами.