Тестовое задание

Постановка задачи:

1) Реализовать на языке программирования Java и фреймворке Spring микросервисы для предоставления возможностей пользователям покупать транспортные билеты.
2) Задание состоит из основной задачи и дополнительных задач, призванных показать степень владения теми или иными технологиями. Выполнение дополнительных задач не обязательно, но будет плюсом.
3) Реализация может допускать различные вариации. Цель задания – показать уровень владения технологиями, а не скрупулёзное следование техническому заданию.
4) Необходимо реализовать только бэкенд часть (Java микросервисы). Фронтенд часть реализовывать не нужно.
5) Исходный код должен быть опубликован на Github/Gitlab/Bitbucket.
6) Сборка – Maven или Gradle.
7) **Срок выполнения**: две недели.

Основная задача:

1) Реализовать микросервис с REST интерфейсом, который будет обрабатывать заявки на покупку билетов.
2) Основные сущности:
    a. **Билет** – право проезда по определенному маршруту в определенную дату и место в транспортном средстве. Атрибуты: маршрут, дата/время, номер места, цена.
    b. **Маршрут** – определенный путь движения. Атрибуты: пункт отправления, пункт назначения, перевозчик, длительность в минутах.
    c. **Перевозчик** – компания, осуществляющая перевозки. Атрибуты: название, телефон.
    d. **Пользователь** – субъект, осуществляющий покупку билетов. Атрибуты: логин, пароль, ФИО.
3) Основные REST методы:
    a. Регистрация нового пользователя (все логины пользователей уникальны).
    b. Получение списка всех доступных для покупки билетов, с возможностью пагинации (страница/размер) и фильтрации по следующим атрибутам:
        i. Дата/время
        ii. Пункты отправления/назначения (вхождение по строке).
        iii. Название перевозчика (вхождение по строке).
    c. Покупка определенного билета. Уже купленный билет не должен быть повторно доступен для покупки.
    d. Получение списка всех купленных билетов для текущего пользователя.
4) Все данные должны храниться в СУБД PostgreSQL.
5) Не разрешается использовать Hibernate/Spring Data для доступа к БД. Возможные варианты – JDBC, JdbcTemplate, JooQ и т.п.
6) Входные данные REST методов должны валидироваться. В случае ошибки валидации должна возвращаться HTTP ошибка 400, а в теле ответа – сообщение об ошибке (нужно придумать любой формат тела ответа, отличный от стандартного в Spring Web).
7) Необходимо наличие Swagger документации через аннотации в коде.
8) По желанию: возможна реализация с помощью Spring WebFlux вместо Spring Web.

Не обязательно: написать Dockerfile для создания Docker образа микросервиса

Дополнительная задача 1

1) Реализовать аутентификацию пользователей с помощью JWT токенов (access и refresh токены по типу OAuth 2). Access токен должен быть ограничен по времени действия.

2) Добавить методы для аутентификации уже зарегистрированных пользователей и обновления access токена по refresh токену.
3) При вызове основных методов с неверным токеном должна возвращаться HTTP ошибка 401.

Дополнительная задача 2

1) Добавить для пользователей сущность «Роль», которая подразделяется на «Покупатель» и «Администратор».
2) Добавить новые REST методы для управления (добавление/изменение/удаление) сущностей **Билет**, **Маршрут** и **Перевозчик**. Данные методы должны быть доступны только для пользователей с ролью «Администратор».
3) Для покупателей вызовы этих методов должны приводить к HTTP ошибке 403.

Дополнительная задача 3 *

1) Добавить кеширование купленных билетов для каждого пользователя в БД Redis.
2) При запросе купленных билетов для текущего пользователя список билетов должен сначала извлекаться из Redis. Если данные в Redis отсутствуют, должен производиться запрос в БД PostgreSQL и обновление данных в Redis.
3) При покупке билета данные в Redis должны обновляться.

Дополнительная задача 4 **

1) Добавить новый микросервис, который будет принимать данные о купленных билетах из топика Kafka и сохранять в отдельную таблицу в БД (тот же PostgreSQL или любую другую на выбор, например, NoSQL БД).
2) При покупке билета в основном микросервисе данные билета дополнительно отправляются в этот топик Kafka.