Тестовое задание для кандидатов на позицию Java разработчика

****Срок выполнения:** 2-3 дня**

****Система сборки:** Gradle**

****База данных:** PostgreSQL**

****Фреймворк:** Spring Boot**

****Архитектура:** Микросервисная архитектура**

****Формат сдачи:** Ссылка на репозиторий GitHub с исходным кодом и инструкцией по запуску приложения.**

Задание

1. **Создание микросервиса "Пользователи"**

Описание:
Создайте микросервис, который будет отвечать за управление пользователями. Микросервис должен предоставлять REST API для выполнения CRUD операций над пользователями.

Требования:

- **Модель данных:**
  - `User`:
    - `id` (Long) - уникальный идентификатор пользователя.
    - `username` (String) - имя пользователя (уникальное).
    - `email` (String) - email пользователя (уникальное).
    - `password` (String) - пароль пользователя (хранить в зашифрованном виде).
    - `createdAt` (LocalDateTime) - дата и время создания пользователя.
    - `updatedAt` (LocalDateTime) - дата и время последнего обновления пользователя.

- **REST API:**
  - `GET /api/users` - получение списка всех пользователей.
  - `GET /api/users/{id}` - получение пользователя по ID.
  - `POST /api/users` - создание нового пользователя.
  - `PUT /api/users/{id}` - обновление пользователя по ID.
  - `DELETE /api/users/{id}` - удаление пользователя по ID.

- **Валидация:**
  - Имя пользователя должно быть уникальным.
  - Email должен быть уникальным и валидным.

  - Пароль должен быть зашифрован перед сохранением в базе данных.

- **Тестирование:**
  - Напишите unit-тесты для сервисного слоя.
  - Напишите интеграционные тесты для REST API.

2. **Создание микросервиса "Роли"**

Описание:
Создайте микросервис, который будет отвечать за управление ролями пользователей. Микросервис должен предоставлять REST API для выполнения CRUD операций над ролями.

Требования:

- **Модель данных:**
  - `Role`:
    - `id` (Long) - уникальный идентификатор роли.
    - `name` (String) - название роли (уникальное).
    - `description` (String) - описание роли.

- **REST API:**
  - `GET /api/roles` - получение списка всех ролей.
  - `GET /api/roles/{id}` - получение роли по ID.
  - `POST /api/roles` - создание новой роли.
  - `PUT /api/roles/{id}` - обновление роли по ID.
  - `DELETE /api/roles/{id}` - удаление роли по ID.

- **Валидация:**
  - Название роли должно быть уникальным.

- **Тестирование:**
  - Напишите unit-тесты для сервисного слоя.
  - Напишите интеграционные тесты для REST API.

3. **Связь между микросервисами "Пользователи" и "Роли"**

Описание:
Реализуйте связь между микросервисами "Пользователи" и "Роли". Каждый пользователь может иметь несколько ролей, и каждая роль может быть назначена нескольким пользователям.

Требования:

- **Модель данных:**
  - `UserRole`:
    - `userId` (Long) - идентификатор пользователя.
    - `roleId` (Long) - идентификатор роли.

- **REST API:**

  - `GET /api/users/{userId}/roles` - получение списка ролей пользователя.
  - `POST /api/users/{userId}/roles/{roleId}` - назначение роли пользователю.
  - `DELETE /api/users/{userId}/roles/{roleId}` - удаление роли у пользователя.

- **Тестирование:**
  - Напишите unit-тесты для сервисного слоя.
  - Напишите интеграционные тесты для REST API.

4. **Архитектурный подход**

Описание:
Опишите архитектурный подход, который вы выбрали для реализации микросервисов. Объясните, почему вы выбрали именно этот подход, и какие преимущества он дает.

Требования:

- **Микросервисная архитектура:**
  - Опишите, как вы разделили приложение на микросервисы.
  - Объясните, как вы организовали взаимодействие между микросервисами.
  - Расскажите о выбранном способе хранения данных (например, отдельные базы данных для каждого микросервиса).

- **Слои приложения:**
  - Опишите структуру слоев вашего приложения (например, контроллеры, сервисы, репозитории).
  - Объясните, как вы организовали взаимодействие между слоями.

- **Тестирование:**
  - Опишите, как вы организовали тестирование вашего приложения.
  - Объясните, какие виды тестов вы использовали (unit-тесты, интеграционные тесты) и почему.

Дополнительные требования

- **Логирование:**
  - Реализуйте логирование запросов и ответов в REST API.
  - Используйте библиотеку для логирования (например, SLF4J).

- **Безопасность:**
  - Реализуйте базовую аутентификацию для доступа к REST API.
  - Используйте Spring Security для защиты API.

- **Документация:**
  - Добавьте документацию к API с использованием Swagger.

Критерии оценки

- **Качество кода:** Чистота, читаемость, соблюдение стандартов кодирования.
- **Архитектура:** Выбор архитектурного подхода, его обоснование и реализация.
- **Тестирование:** Полнота и качество тестов.
- **Безопасность:** Реализация механизмов безопасности.
- **Документация:** Полнота и понятность документации.

Срок выполнения: 2-3 дня

Формат сдачи: Ссылка на репозиторий GitHub с исходным кодом и инструкцией по запуску приложения.

Удачи!