Цель

Разработать систему приема платежей, включающую работу с бронированиями, оплатами и проверкой их статусов. Система должна быть безопасной, поддерживать разные платежные провайдеры и включать административный интерфейс.

Функциональные требования

Регистрация бронирования:

Создать публичное API для регистрации бронирования. API возвращает токен, необходимый для дальнейшей оплаты.

Оплата бронирования:

Создать публичное API для проведения оплаты на основании токена и данных карты. Поддерживать два замоканных провайдера: Провайдер для России. Провайдер для других стран. Провайдеры должны возвращать статусы оплаты (SUCCESS, FAILED, PENDING).

Управление статусами:

Бронирования и платежи должны иметь статусы:

Бронирование: NEW, PAID, CANCELED. Платеж: PENDING, SUCCESS, FAILED. Реализовать API для проверки текущего статуса бронирования и оплаты. Клиенты могут работать только со своими бронированиями. Отмена бронирования:

Клиенты могут отменять свои бронирования, если они еще не оплачены.

Административный интерфейс:

Создать административное API для: Просмотра всех бронирований. Просмотра всех платежей. Административное API должно быть доступно только пользователям с ролью "Администратор".

Автоматическая проверка статусов:

Реализовать фоновую задачу для автоматической проверки статусов неоплаченных платежей. Задача должна выполняться каждые 5 минут и обновлять статусы в базе данных на основании ответов замоканных провайдеров.

Технические требования

Технологии:

Язык программирования: Kotlin. Framework: Spring Boot. Безопасность: Spring Security. База данных: MongoDB. Фоновая задача: Spring Scheduling. Хранение данных:

Бронирования и платежи должны сохраняться в MongoDB. Настройка базы данных через docker-compose.

API:

Разработать RESTful API. Предоставить документацию через Swagger/OpenAPI. Разграничение доступа:

Клиенты могут работать только со своими бронированиями. Административное API доступно только пользователям с ролью "Администратор".

Платежные провайдеры:

Имитация двух провайдеров: Один для России (например, возвращает данные быстрее). Один для других стран (например, может эмулировать задержки). Возвращаемые статусы: SUCCESS, FAILED, PENDING. Периодическая проверка статусов:

Реализовать фоновую задачу с использованием @Scheduled. Задача должна проверять статусы только у платежей со статусом PENDING или аналогичным.

Запуск:

Предоставить инструкцию по запуску системы, включая команды для поднятия MongoDB через docker-compose.