**Комментарии по архитектурному решению**

Данное решение представляет собой систему бронирования, состоящую из следующих сервисов:

* User Service – сервис управления пользователями.
* Booking Service – сервис управления бронированиями.
* Payment Service – сервис управления платёжными аккаунтами и выполнения платежей.
* Notification Service – сервис оповещения пользователей.

Каждый сервис (например, Payment Service) состоит из двух частей:

* REST Endpoint (например, Payment Service REST Endpoint) – для выполнения синхронных операций (таких как создание платёжного аккаунта либо добавление средств на платёжный аккаунт).
* Background Processor - для асинхронного межсервисного взаимодействия через Kafka (например, Payment Service Background Processor).

Каждый сервис использует собственные таблицы в БД. Потенциально для каждого сервиса можно было развернуть собственную БД, но для упрощения все сервисы физически разделяют общую БД.

Background Processor в конкретном сервисе представляет собой набор Listeners (Kafka Consumers), каждый из которых обрабатывает сообщения определённого типа из соответствующего топика. Разные Listeners могут слушать сообщения из одной и той же очереди (что достигается посредством задания разных Consumer Groups). Оправка исходных сообщений (например, команд MakePayment) и ответов (событий PaymentSucceeded / PaymentFailed) происходит через один и тот же топик (payments), отдельные топики для ответов (replies) не используются в целях упрощения.

Скрипт создания топиков в лабораторной работе не используется. Для целей лабораторной работы подходит, что Kafka при попытке отправки сообщения в несуществующий топик создаёт его автоматически (хотя и с параметрами по умолчанию – без партиционирования и репликации).

Авторизации и аутентификации в данное работе нет. Сценарий начинается с создания пользователя в User Service через REST Endpoint. При этом в Payment Service отправляется событие UserCreated. Background Processor в Payment Service, получив это событие, создаёт соответствующий пользователю Payment Account. Для совершения бронирований на соответствующий Payment Account нужно положить деньги через REST метод DepositAmount. На каждую операцию с платёжным аккаунтом (добавление / списание средств) в Payment Service создаётся Transaction соответствующего типа (Deposit / Withdrawal).

Создание заказа производится через синхронный REST Endpoint в Booking Service. Бронирование создаётся всегда, то есть без учёта состояние платёжного аккаунта - в статусе Created. Ответ о создании бронирования пользователю возвращается сразу же. Далее созданные бронирование поступает на обработку в Booking Processor, который переводит его в статус Processing и отправляет команду Make Payment в Payment Service. Payment Processor, в зависимости от достаточности средств в платёжном аккаунте, отправляет событие либо PaymentSucceeded, либо PaymentFailed. Booking Processor в ответ меняет состояние бронирования либо на Completed, либо на Rejected, после чего отправляет одну из команд NotifyPaymentSuccess / NotifyPaymentFailure в NotificationService.

Обмен событиями и командами производится в JSON-формате. Сериализацию / десериализацию выполняет специально написанный generic сериализатор.