Введение

Существует Сервис, позволяющий выполнять операции и получать их результат. Взаимодействие с сервисом асинхронное, через шину событий. Для постановки задачи отправляем событие { command, taskId, payload }. Результат получаем в виде события от Сервиса { result, taskId }.

Сценарии бизнес-логики предполагают последовательный и условный вызов ряда операций Сервиса. Очевидно, при помощи императивного подхода, невозможно реализовать код бизнес-логики, удовлетворяющий Требованиям, описанным ниже.

Задача

Необходимо предложить и реализовать (опционально) способ работы с Сервисом, позволяющий описывать процессы бизнес-логики, максимально приближенно к императивному стилю, с удобной обработкой ошибок и легко тестировать такой код. Другие варианты (включая декларативный стиль описания), удовлетворяющие требованиям, приветствуются.

Требования

1. Решение должно быть горизонтально масштабируемым. Потенциально, обработчики разных шагов бизнес-логики могут выполняться в разных экземплярах приложения.
2. Решение должно быть отказоустойчивым. В случае непредвиденной ошибки (падение/рестарт приложения, обрыв связи, ...), процесс выполнения должен возобновляться с шага остановки.

Дополнительно

* Предполагается использование RabbitMQ в качестве среды для обмена данными. Мотивированные альтернативные предложения, также рассматриваются.
* Допустимо использовать внешние БД (SQL/NoSQL), при наличии мотивации.
* Для упрощения, в коде тестового задания допустимо использование стандартного механизма Event из Node.js.

Пример бизнес-логики

Сервис реализует функции checkApproved, approve, getBalance, feed, sign, send. Приложение использует эти функции.

Упрощенный псевдокод, описывающий бизнес-логику в императивном стиле:

|  |
| --- |
| businessCase1(address) {   isApproved = service.checkApproved('123')   if (!isApproved) {     service.approve('123')   }   balance = service.getBalance(address)   if (balance < 5) {     service.feed(address, 5 - balance)   }    signature = service.sign()    return service.send('123', address, signature) } |

Ожидаемый результат

* Предложение варианта реализации. Дизайн-документ с описанием предлагаемого решения, технических деталей реализации, анализом его плюсов и минусов, возможных альтернативных решений.
* Пример реализации решения на Typescript (опционально).