

Ещё один антипаттерн разделения монолита на микросервисы - "сущность-сервис"

Антипаттерн Entity Service заключается в том, что при разбиении монолита на микросервисы каждый микросервис становится просто тонкой оберткой вокруг отдельной таблицы базы данных или отдельного типа объекта. Это может привести к высокой степени связанности между сервисами и сложностями в реализации бизнес-логики.

Простой пример

Допустим, у нас есть интернет-магазин, и мы решаем перейти на микросервисную архитектуру. Один из простых и наивных способов декомпозиции — создать микросервисы вокруг каждой основной сущности (продукты, клиенты, заказы): ProductService , CustomerService , OrderService , и так далее.

Антипаттерн

- ProductService отвечает только за CRUD-операции с продуктами.
- CustomerService отвечает только за CRUD-операции с данными клиентов.
- OrderService отвечает только за CRUD-операции с заказами.

Проблемы

1. **Высокая связанность:** Если нам нужно реализовать функционал, который требует взаимодействия между продуктами, клиентами и заказами (например, оформление заказа), нам придется либо реализовать эту логику на клиенте (что плохо с точки зрения безопасности и повторного использования), либо создать еще один сервис, который будет координировать остальные.
2. **Дублирование кода:** Бизнес-логика, которая касается взаимодействия между сущностями, будет дублироваться в разных сервисах или клиентах.
3. **Сложность и затраты на поддержку:** Из-за высокой степени связанности, изменения в одном сервисе могут потребовать изменений в других, что делает систему хрупкой и трудно поддерживаемой.

Более подходящее решение

Вместо того чтобы создавать микросервисы вокруг отдельных сущностей, лучше сгруппировать их вокруг бизнес-функциональностей. Например, можно создать сервис `OrderManagement`, который будет отвечать за всю логику оформления заказов, включая проверку наличия товара на складе, расчет стоимости и обработку данных клиента. Этот сервис будет взаимодействовать с сервисами продуктов и клиентов `ProductService` и `CustomerService`, но бизнес-логика будет централизована.

Такой подход позволяет сократить связанность и упростить поддержку, делая каждый микросервис ответственным за конкретный набор функций, логически связанных между собой.

Ещё показатели, что вы (возможно) неправильно декомпозировали монолит на микросервисы

- Сервис А не может выполнить свою функцию без синхронной интеграции с сервисом Б - возможно вы их зря разделили или неправильно и можно объединить
- При внедрении новых доработок системы вы часто в одних и тех же задачах дорабатываете и сервис А и сервис Б - возможно вы их зря разделили или неправильно и можно объединить
- Вам нужна строгая согласованность между сервисами А, Б и В - возможно вы их зря разделили или неправильно и можно объединить