SolarLab>_

Банально про архитектуру и микросервисы

Алексей Андреев https://t.me/AlekseyAndreev1984

Содержание

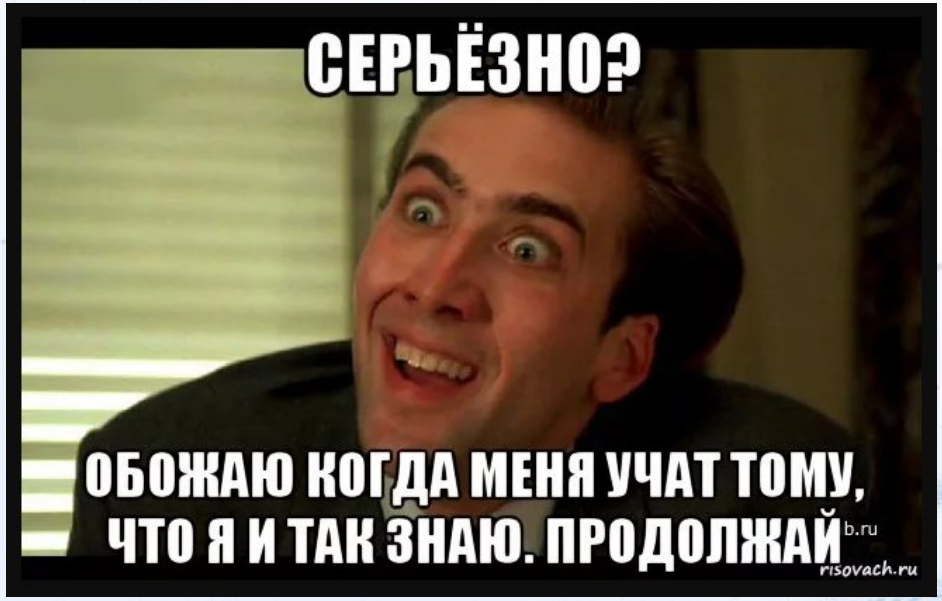
SolarLab>_

Почему банально

— Почему банально

Надеюсь в конце доклада вы все скажите:

Я и так всё это знал. Умел. И у меня есть своё мнение на этот счёт. Я и так всё знаю!



Почему банально

А вдруг?

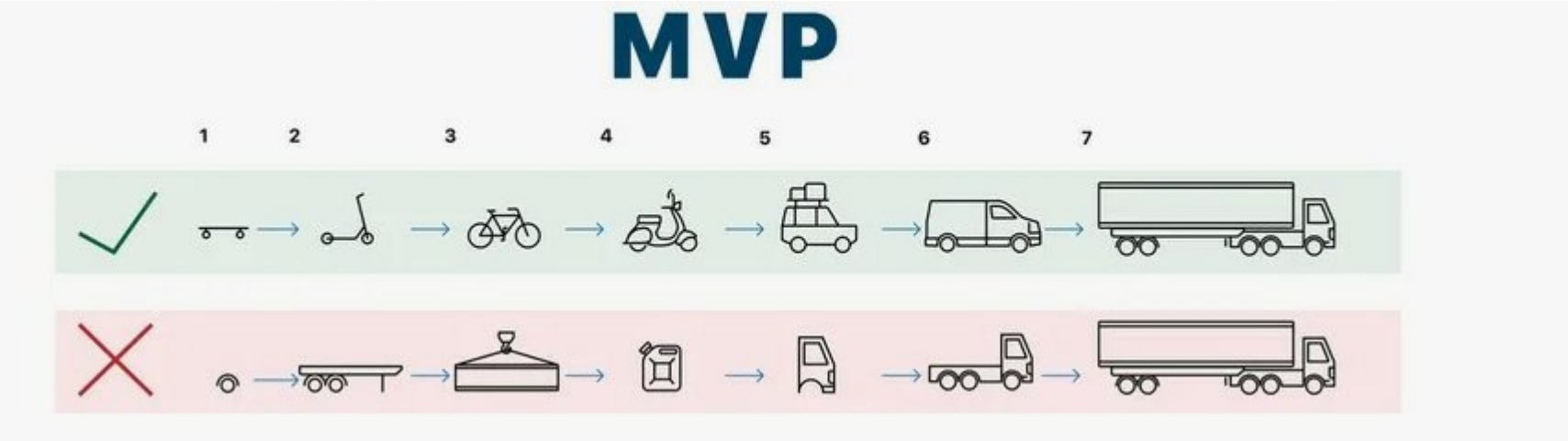
SolarLab>_

История из жизни - как мы делали MVP

История из жизни - как мы делали MVP

А давайте сделаем MVP

Что такое MVP?



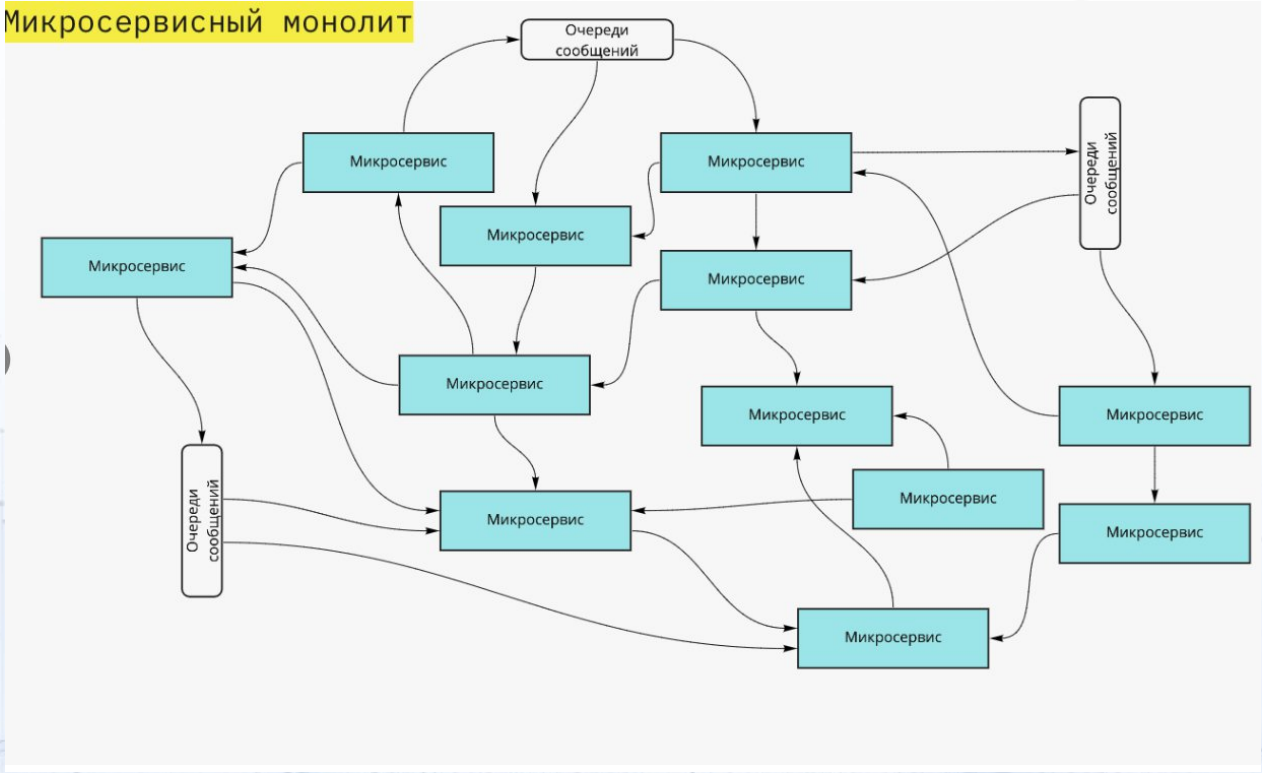
История из жизни - как мы делали MVP

MVP – это minimum VIABLE product: минимально жизнеспособный продукт

Какие могут быть подводные камни тут?

История из жизни - как мы делали MVP

- И что по итогу? Микросервисный монолит в одном репозитории



SolarLab>_

Микросервисы - как я их вижу

Микросервисы - как я их вижу

Характеристики микросервисов

- Архитектура микросервисов состоит из небольших, независимых и слабо связанных между собой служб.
- Каждая служба является отдельной базой кода, которой может управлять небольшая команда разработчиков.
- Службы можно развертывать независимо друг от друга. Разработчики могут обновлять существующую службу без повторной сборки и повторного развертывания всего приложения.
- Службы отвечают за сохранение собственных данных или внешнего состояния. В этом состоит отличие от традиционной модели, в которой сохранение данных обрабатывается на отдельном уровне.
- Службы взаимодействуют между собой с помощью четко определенных API-интерфейсов. Сведения о внутренней реализации каждой службы скрыты от других служб.
- Службы не должны(но могут) совместно использовать один и тот же стек технологий, библиотеки или платформы.

Архитектура микросервисов состоит из небольших, **независимых** и слабо связанных между собой служб.

Что есть независимые?

Независимость – когда от вас ничего не зависит.

- **Технологическая независимость**: Сервисы могут быть написаны на разных языках (C#, Java, Python) и использовать разные базы данных (SQL, NoSQL).
- **Жизненный цикл**: Каждый сервис имеет свой цикл разработки, тестирования и развертывания.
- **Масштабирование**: Сервисы масштабируются независимо (например, можно увеличить количество инстансов сервиса оплаты при высокой нагрузке).

Архитектура микросервисов состоит из небольших, независимых и **слабо связанных** между собой служб.

Что есть слабо связанные?

Слабая связность

- Сервисы **не зависят** от внутренней реализации друг друга.
- Взаимодействие происходит через **публичные API** или асинхронные сообщения (например, через брокеры RabbitMQ, Kafka).
- Нет общей базы данных: каждый сервис управляет **своими данными** (принцип *Database per Service*).

SolarLab>_

Преимущества и недостатки микросервисов

Преимущества и недостатки микросервисов

Преимущества микросервисов

- Гибкость разработки
- Команды могут работать над разными сервисами параллельно.
- Возможность использовать разные технологии для разных задач.
- Масштабируемость
- Можно масштабировать только те сервисы, которые испытывают нагрузку.
- Отказоустойчивость
- Сбой одного сервиса не приводит к падению всей системы.
- Независимое развертывание
- Обновления и исправления внедряются быстрее.
- Поддержка Agile и DevOps
- Упрощается CI/CD (непрерывная интеграция и доставка).

Преимущества и недостатки микросервисов

Недостатки микросервисов

- Сложность управления
- Требуется оркестрация (Kubernetes), мониторинг, логирование.
- Увеличивается количество сетевых вызовов.
- Проблемы с транзакциями
- Реализация распределенных транзакций сложна (шаблон Saga).
- Высокие требования к инфраструктуре
- Нужны инструменты для автоматизации (Docker, Kubernetes).
- Риск «распределенного монолита»
- Если сервисы слишком тесно связаны, преимущества теряются.

SolarLab>_

Советы по ведению микросервисов

Советы по ведению микросервисов

- Если у вас есть несколько сервисов на одной и той же технологии(например REST API .net), и вы постоянно создаёте новые микросервисы, тогда выгодно иметь готовый шаблон сервиса, который уже содержит в себе всю необходимую информацию для старта,чтобы не писать совсем с нуля
- Если у вас код повторяется – выносите в пакеты
- На каждый сервис своя БД. Не ходите к соседним сервисам за данными. Это уже SOA
- Сделайте начальное создание сервиса(его билд + деплой + тесты) в едином стиле и как можно более безболезненно
- Если микросервисы ходят к друг другу за данными – то обычно это вырождается в лапшисервисы
- Избегать лапши сервисов(Service Spaghetti)

SolarLab>_

Вопросы?

Вопросы?

SolarLab>_

Ссылки

Ссылки

1) **Микросервисы на монолите**

https://habr.com/ru/companies/domclick/articles/533082/

2) Книга Фаулера Microservices:

https://martinfowler.com/articles/microservices.html

3) Жизненный опыт https://t.me/AlekseyAndreev1984

4) **Микросервисная архитектура: от монолита к гибкой системе (да, опять)**
https://habr.com/ru/companies/gazprombank/articles/903090/

5) **What are microservices?** https://microservices.io/

6) https://www.youtube.com/watch?v=lL_j7ilk7rc

SolarLab>_

Спасибо за внимание