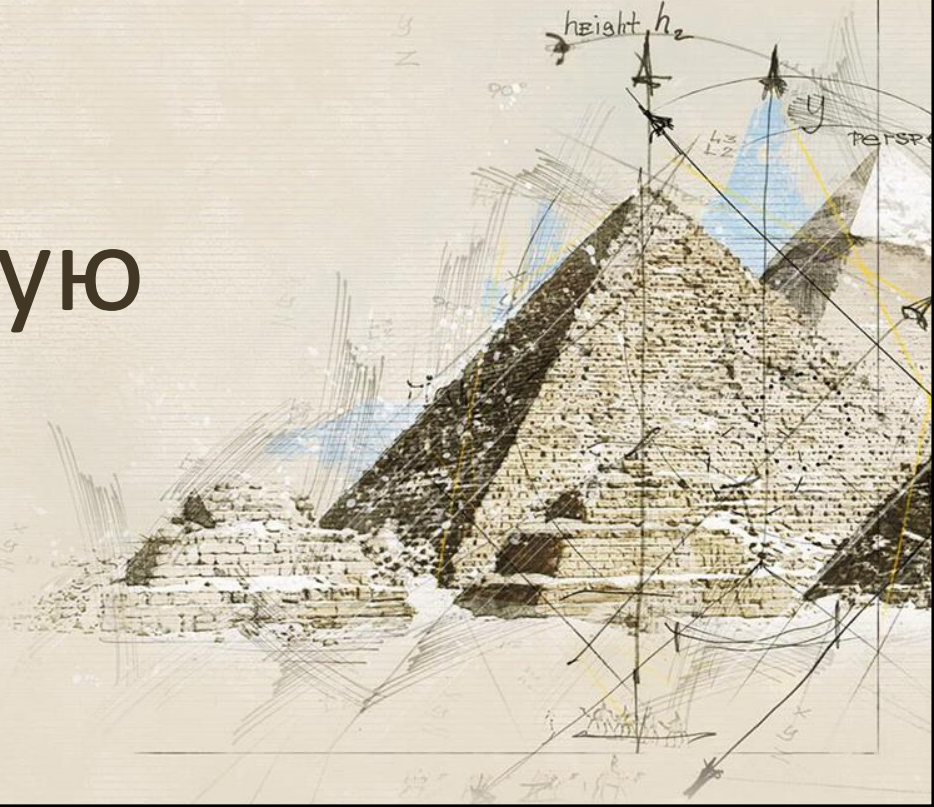
Введение в микросервисную архитектуру

НГУ, январь’20

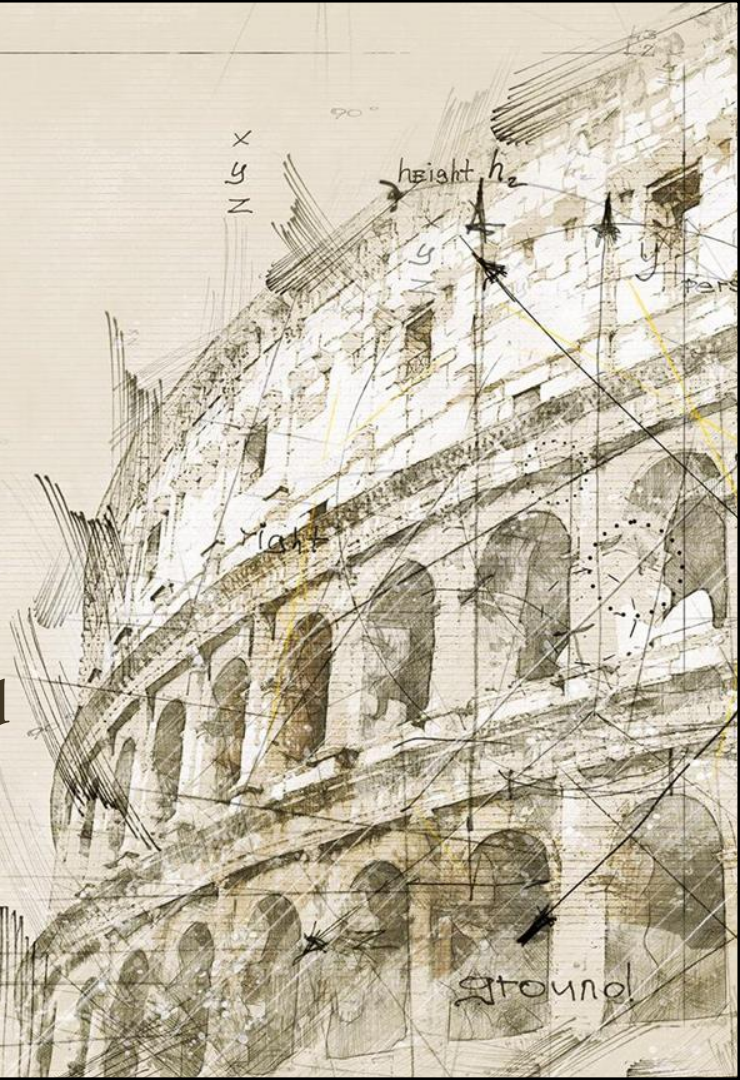


Привет!

Я – Владимир Плизгá

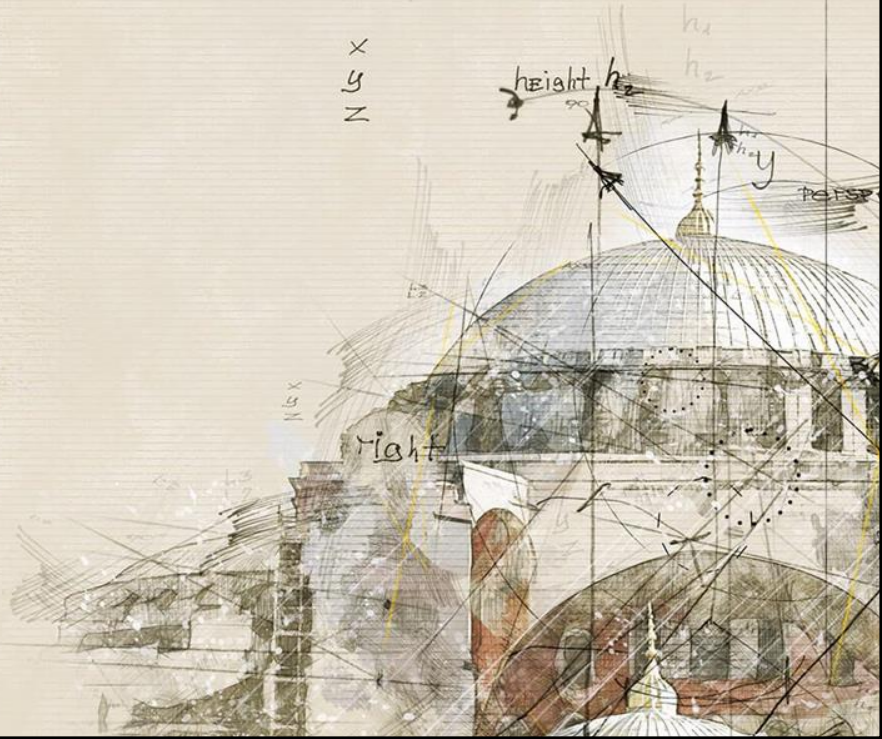
- Java разработчик (backend)
- TechLead серверной команды PrePaid
- ≈8 лет в деле

ЦФТ ЦЕНТР ФИНАНСОВЫХ ТЕХНОЛОГИЙ



1.
Для начала

Цели и вопросы



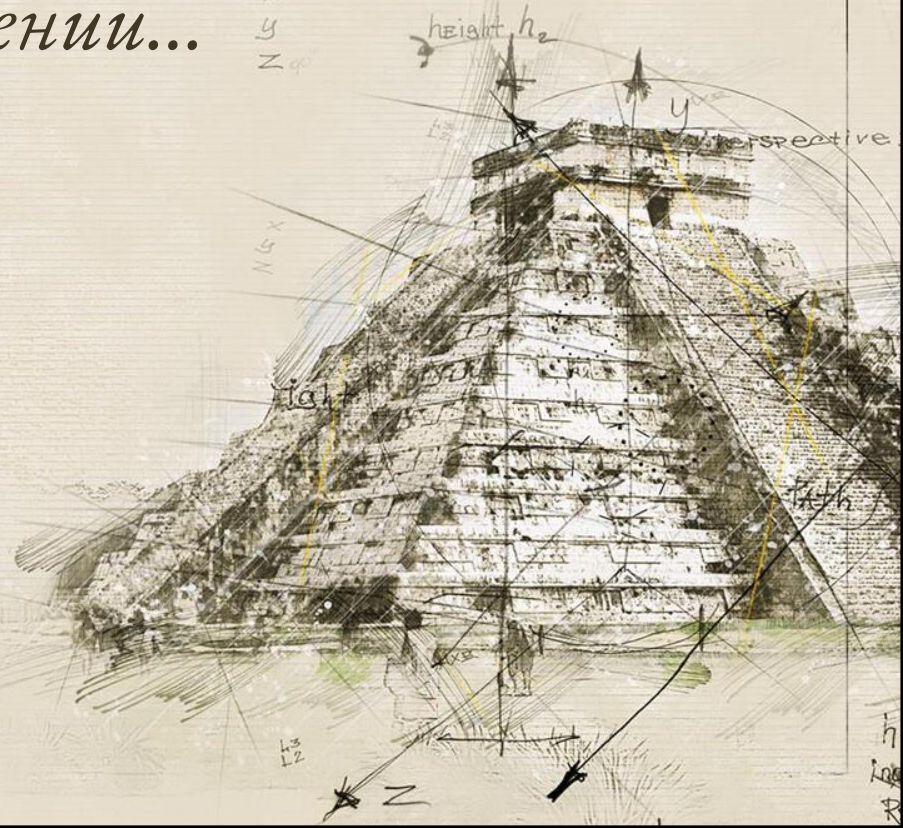
Что это такое?

Оставим вопрос открытым.
Пока нам достаточно
интуитивного понимания.

Цель – ответить на 1 вопрос:

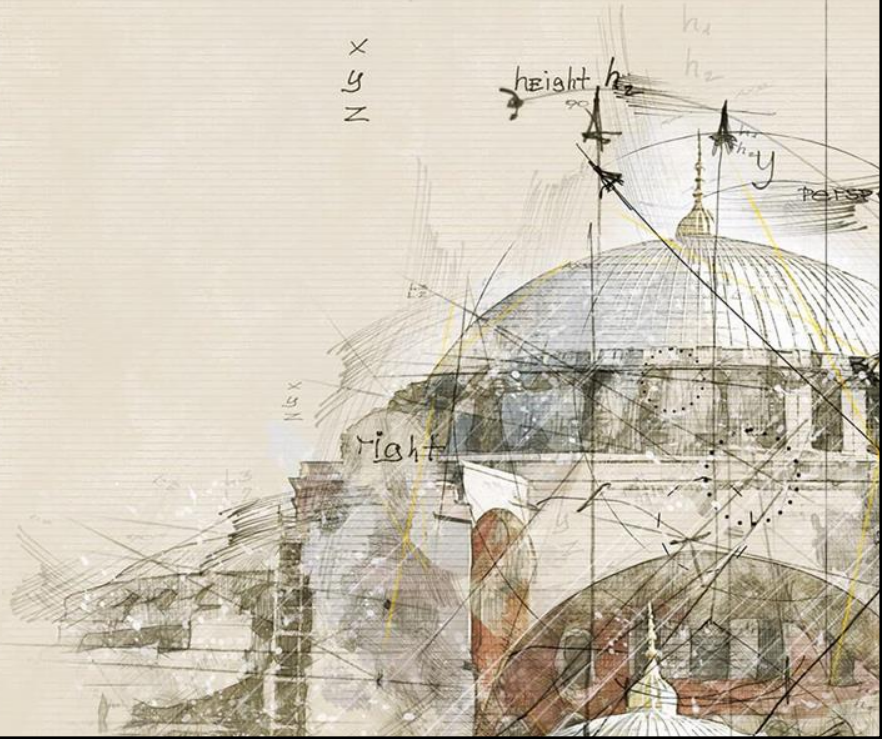
ДЛЯ ЧЕГО ЭТО ВООБЩЕ НУЖНО?

“ *Всё познаётся в сравнении...*



2. Традиционный подход

Как учили нас деды



1 человек – 1 программа (aka 1:1)

- Абсолютно нормальный подход
- Используется давно
- Подходит многим
- Вряд ли исчезнет

Масса позитивных примеров

- Лабораторная/курсовая работа
- Домашний сайт
- Десктопная/консольная утилита
- Telegram-бот
- Мобильное приложение
- *(впиши своё)*

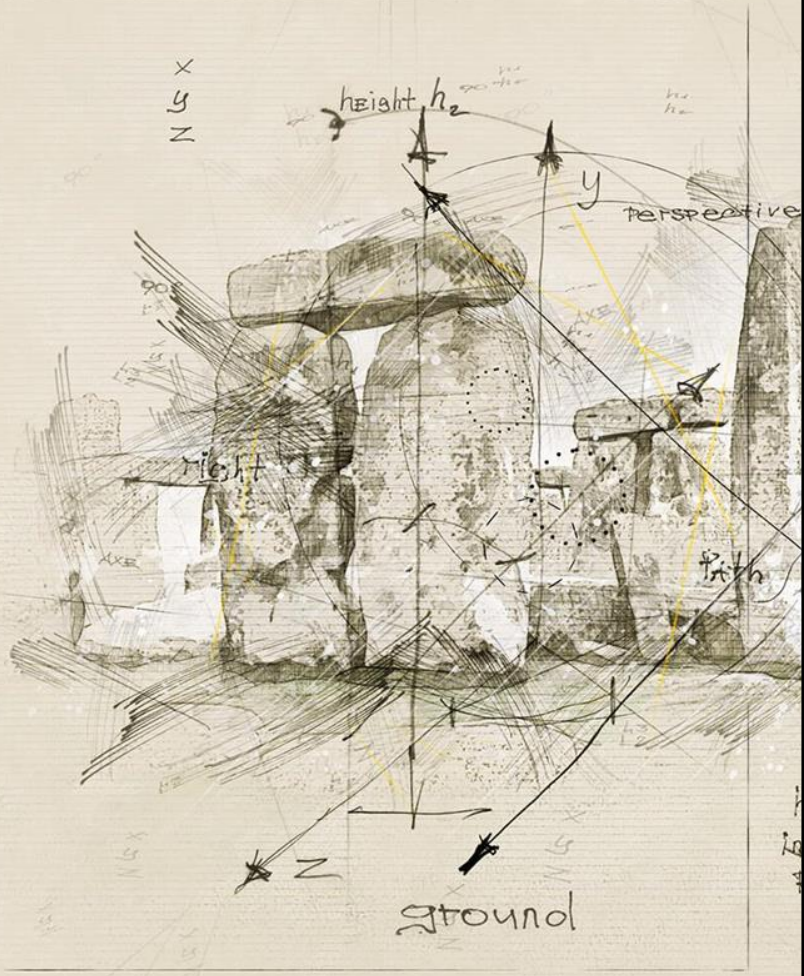
Причины успеха

- Программа **умещается** в 1 голове
- **Легко** меняется (даже вся)
- **Мало** пользователей
  - **Можно** обновлять редко и полностью

А что дальше?

С ростом возможностей растут и сложности


Примерные размеры программ
14 000
12 000
10 000
8 000
6 000
4 000
2 000
0
1 000
5 000
10 000
500 000
Строк кода (LoC)
Курсач
Хомяк
Настолка
Интернет-банк


height h_2
y
perspective
ground


Примерные размеры программ
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
1 000
5 000
10 000
500,000
1,000,000
5,000,000
Строк кода (LoC)
Курсач
Хомяк
Настолка
Интернет-банк
Биллинг
АБС

Сложности большого приложения

- **Не помещается** в 1 голове
- Меняется всё труднее
  - **Сложно** разбираться


class Shape
class Circle : public

Сложности большого приложения

- **Не помещается** в 1 голове
- Меняется всё труднее
  - **Сложно** разбираться
  - **Много** неочевидных связей

https://twitter.com/glorphindale/status/1199663658193891328

Сложности большого приложения

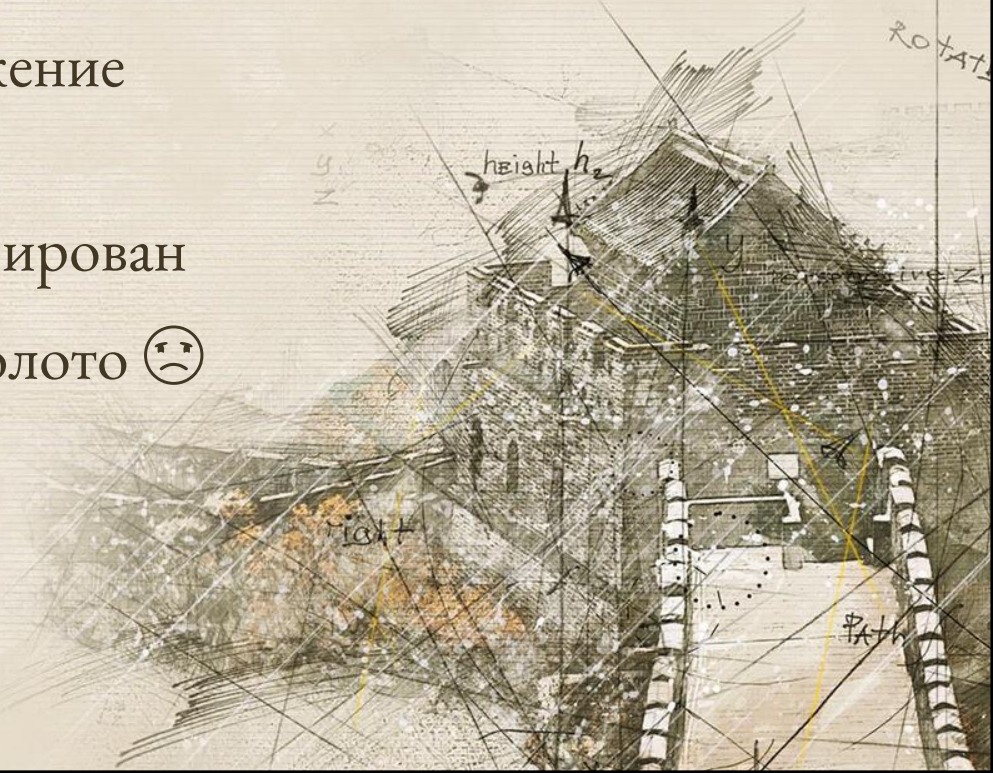
- **Не помещается** в 1 голове
- Меняется всё труднее
  - **Сложно** разбираться
  - **Много** неочевидных связей
  - **Страшно** что-то менять

Следствия растущей популярности

- Нужны обновления
  - Часто
  - Выборочно
  - Незаметно
- Нужно масштабирование

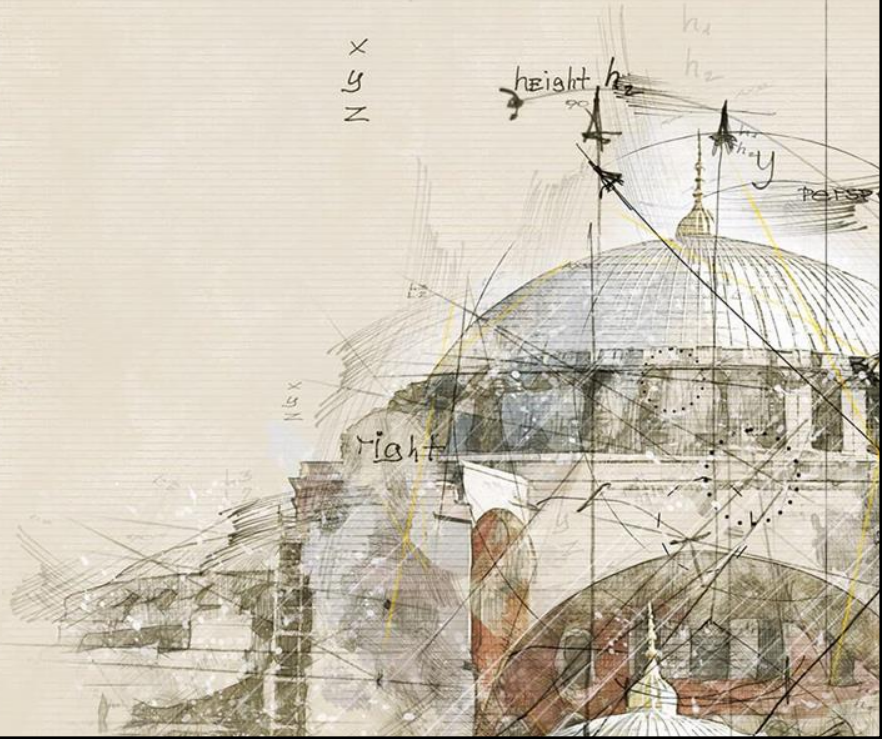
Ситуация глазами разработчика

- Трудно развивать приложение
  - Любая правка чревата
  - Стек технологий фиксирован
- Проект превращается в болото ☹



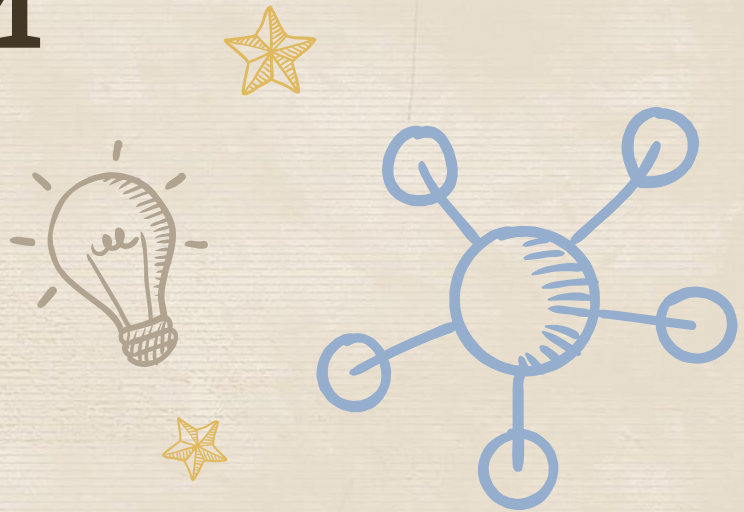
3. Микросервисы

Желанное спасение. Или нет.



Разделяй и властвуй

Человеческая мудрость и инженерный подход в одном флаконе



«Divide Et Impera» в действии

Модули

Классы и пакеты в ООП. Помогают бороться с паразитными связями в исходном коде.

Библиотеки

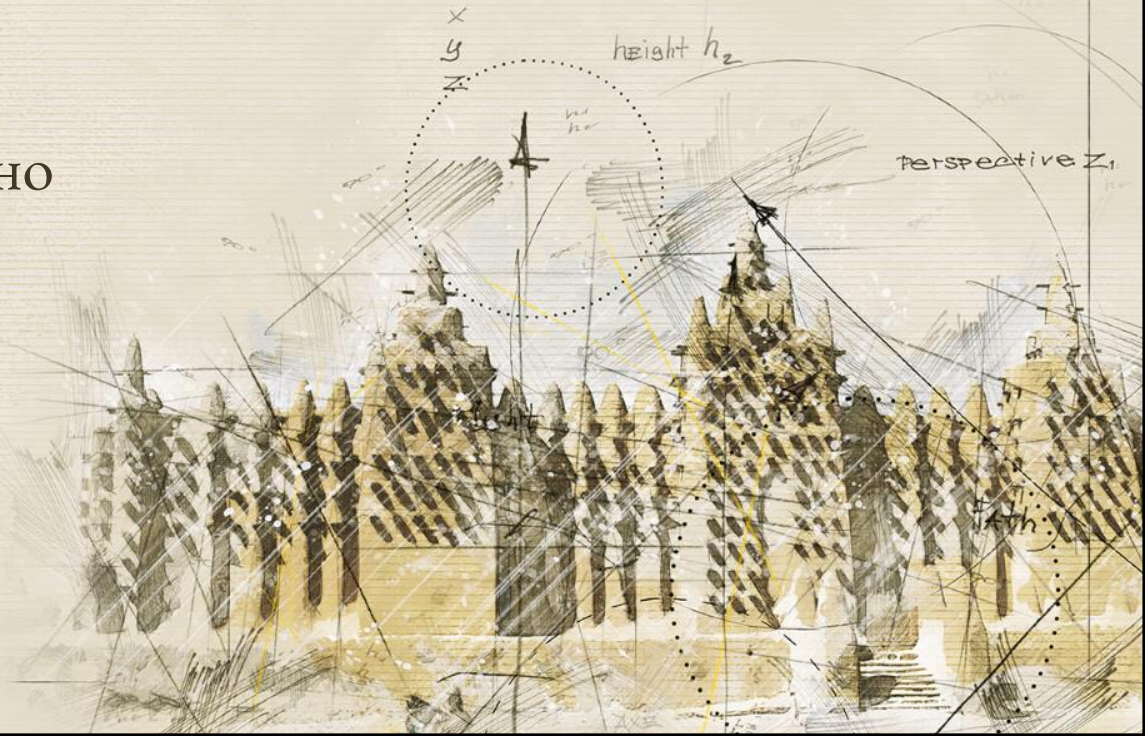
JAR, DLL, PYM, ... Помогают (частично) обновлять приложение частями.

SOA

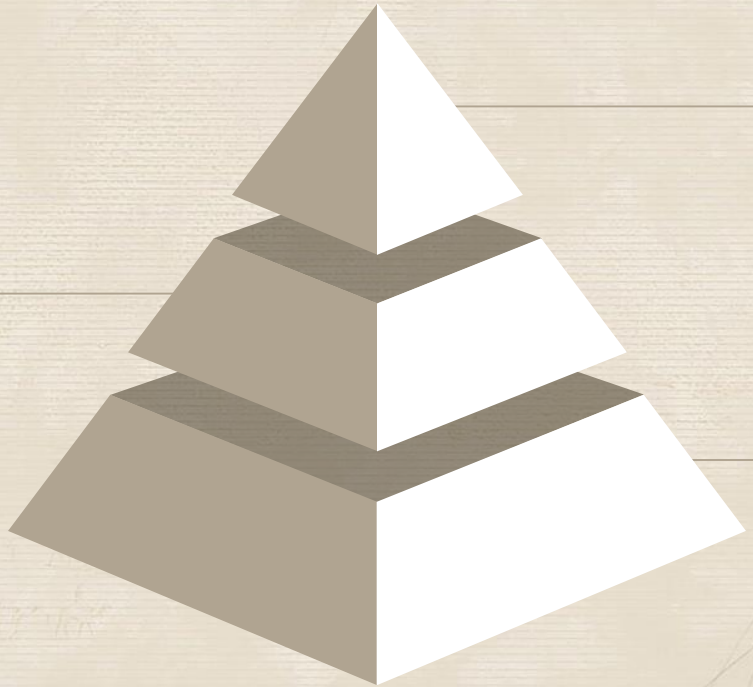
Service **O**riented **A**rchitecture. Помогает (частично) бороться с простоями.

Microservice Architecture (MSA)

Декомпозиция приложения на отдельно **разрабатываемые**, **поставляемые** и **запускаемые** модули.



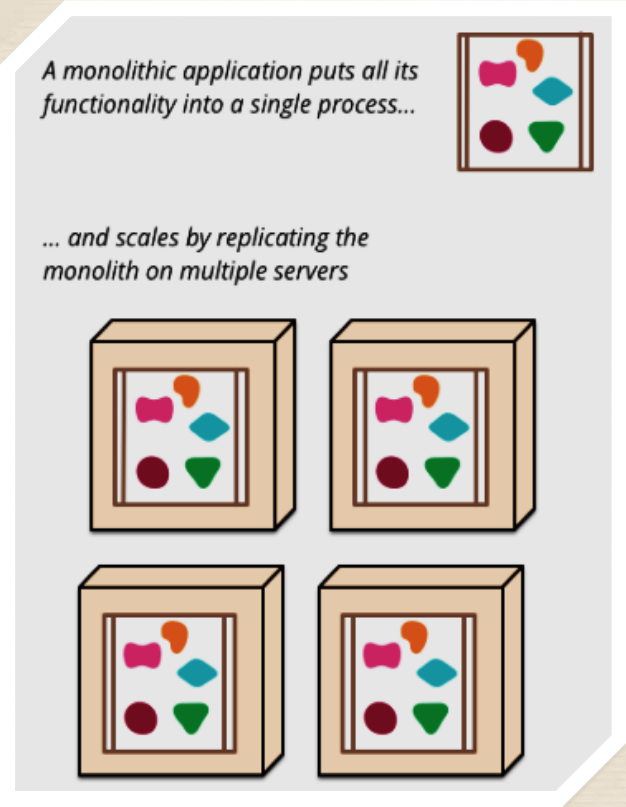
Место MSA в арсенале декомпозиции

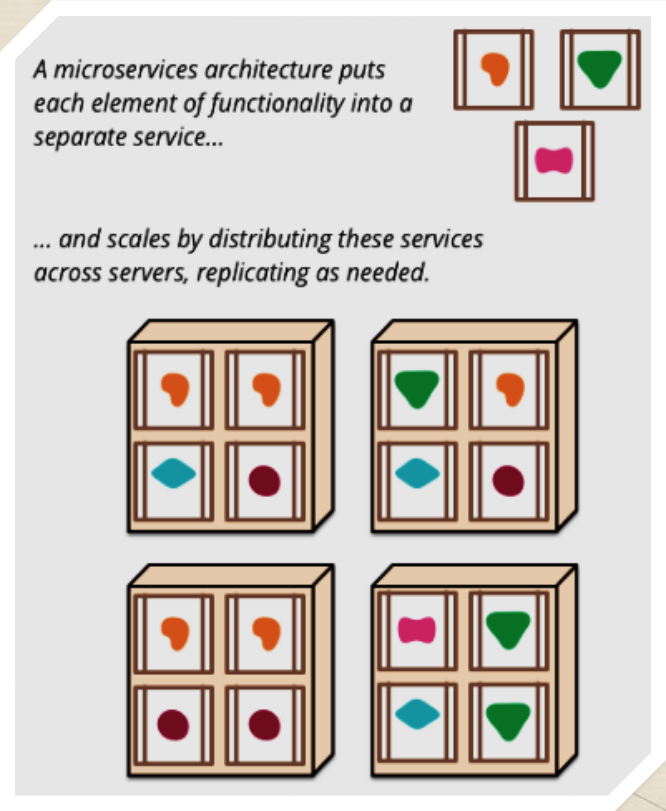


SOA/MSA
≈ **десятки** штук

Библиотеки
≈ **сотни** штук

Классы/пакеты
≈ **тысячи** штук



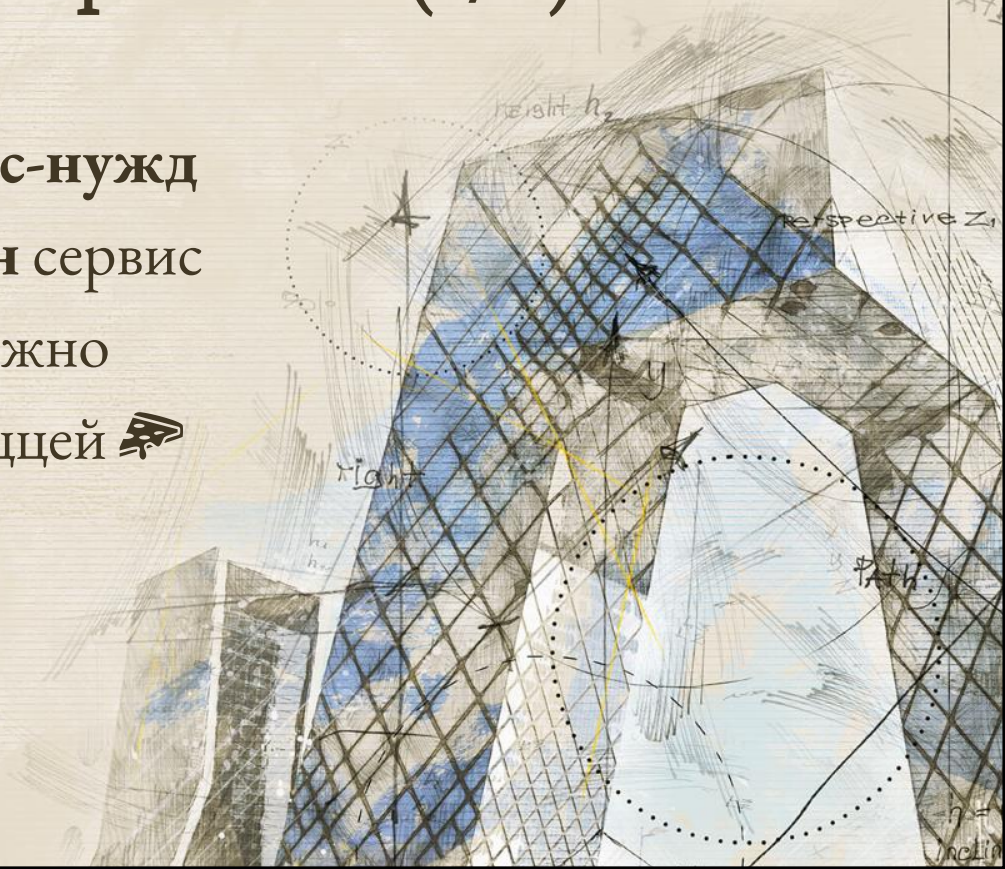


монолит VS микросервисы

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

Свойства микросервисов (1/3)

- Построены вокруг **бизнес-нужд**
  - **Одна** функция ≈ **один** сервис
  - Команда, которую можно накормить **одной** пиццей 🍕

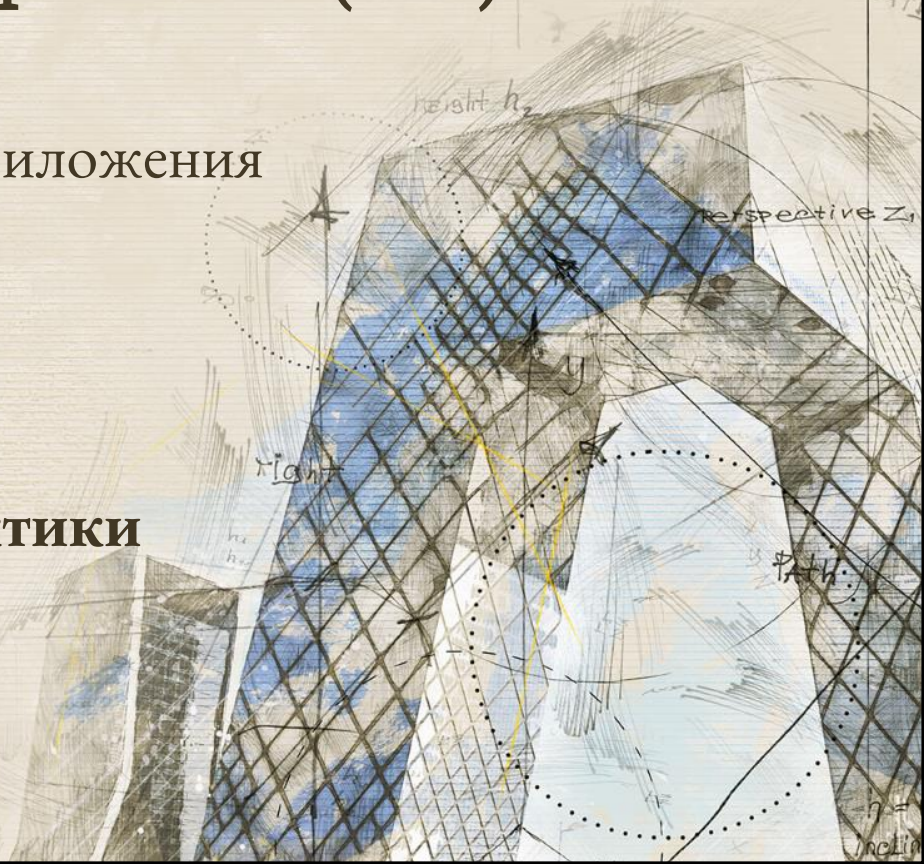


Свойства микросервисов (2/3)

- Соблюдают строгий контракт взаимодействия:
  - Общение только через **API**
  - У каждого **своя БД** (схема)

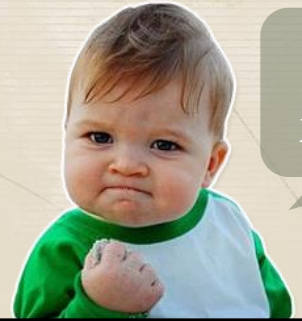
Свойства микросервисов (3/3)

- 12 факторов **cloud-native** приложения
  - Упрощают разработку
  - Упрощают поставку
    (особенно в облако ☁)
  - Включают **лучшие практики**



Эти свойства возвращают лучшее

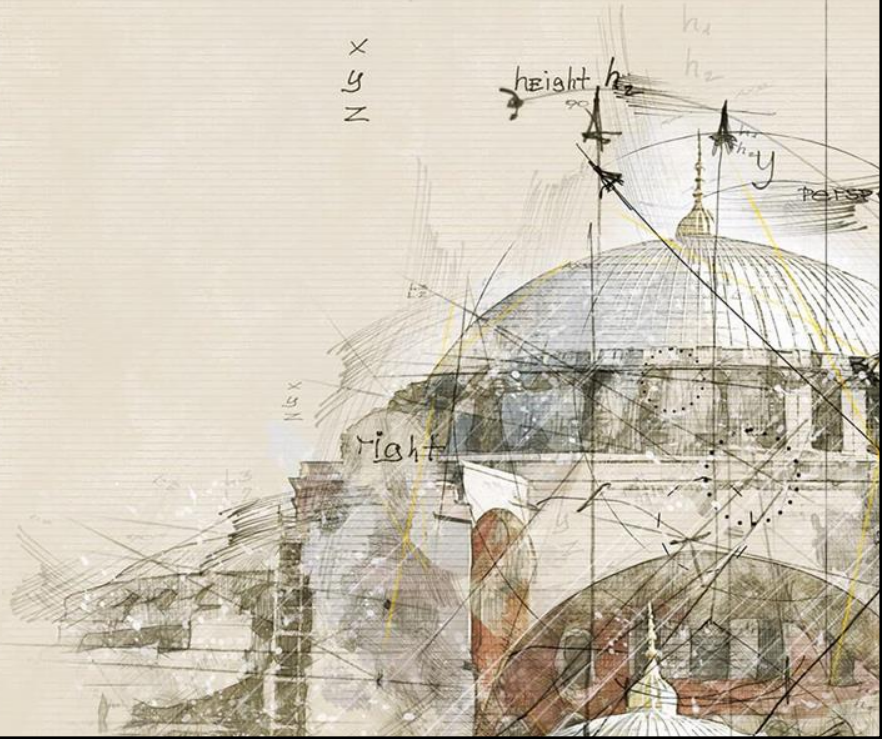
- Программа умещается в 1 голове
- Легко меняется (даже вся)
- Можно обновлять редко и полностью

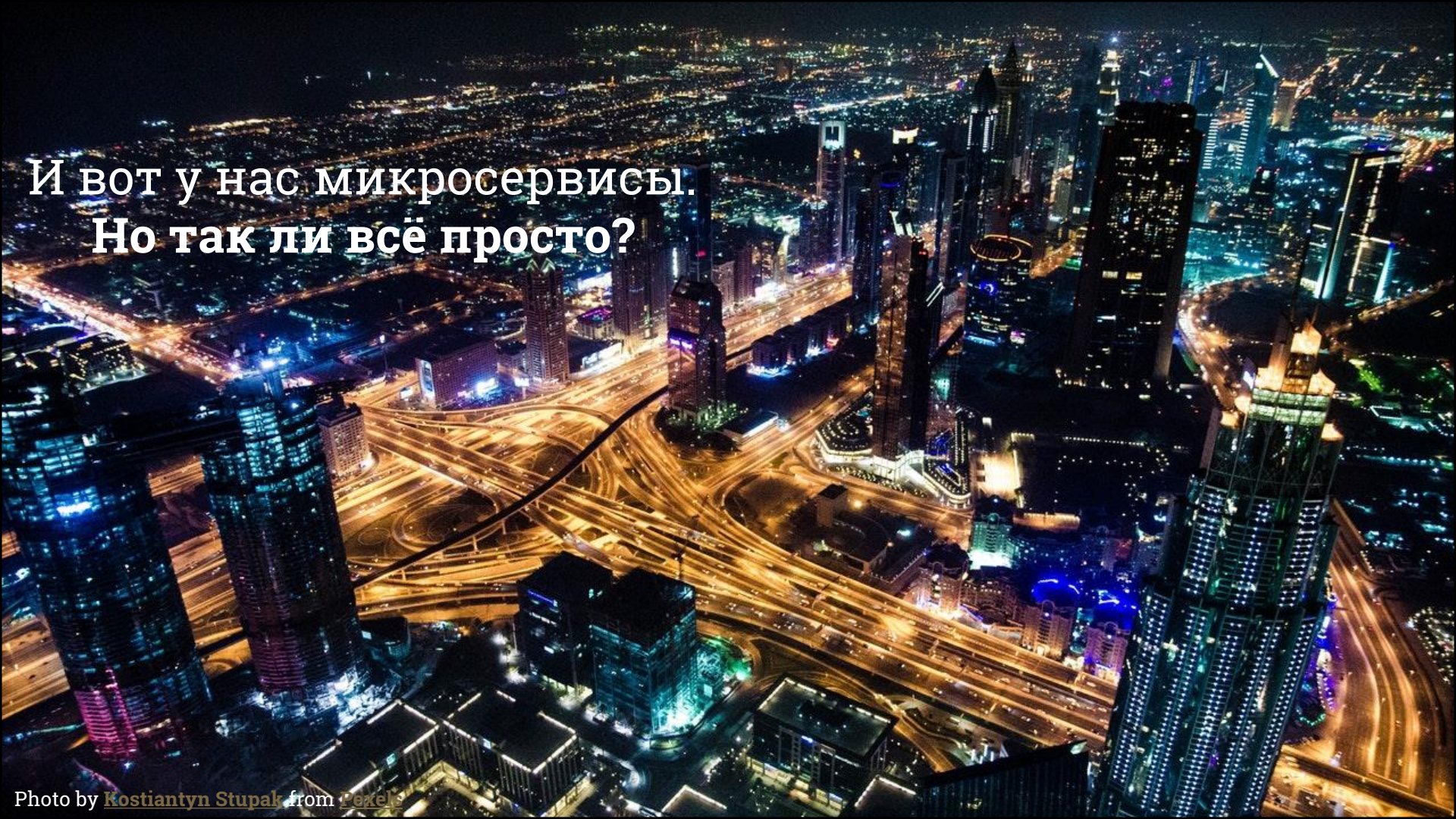


Потому что **маленький**!

4. Прививки реальности

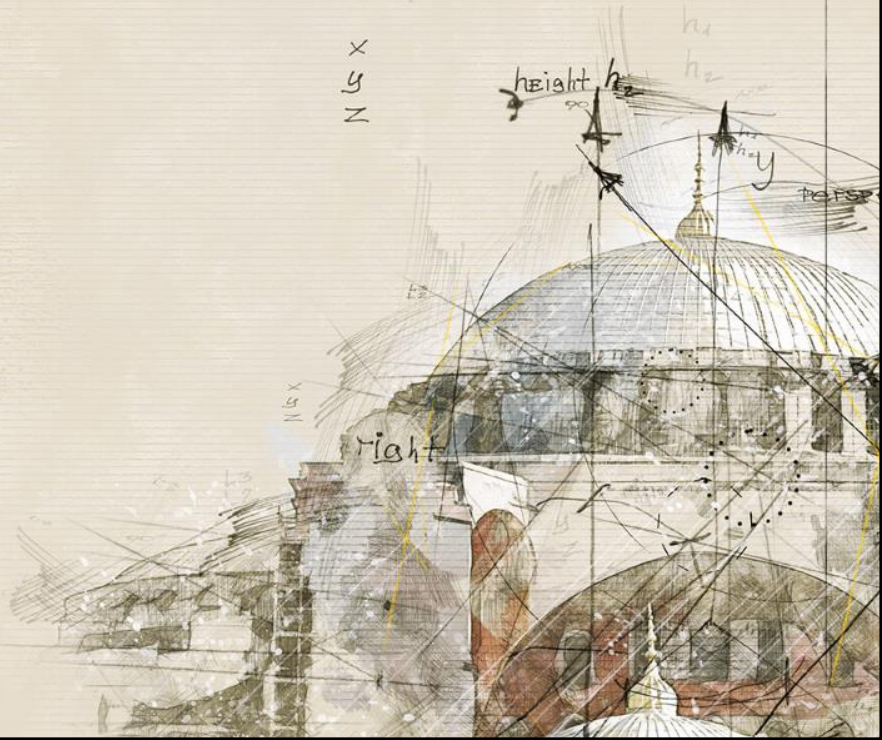
Неочевидные особенности микросервисной архитектуры




И вот у нас микросервисы.
Но так ли всё просто?
Photo by Kostiantyn Stupak from Pexels

1. Как направлять клиентский трафик?

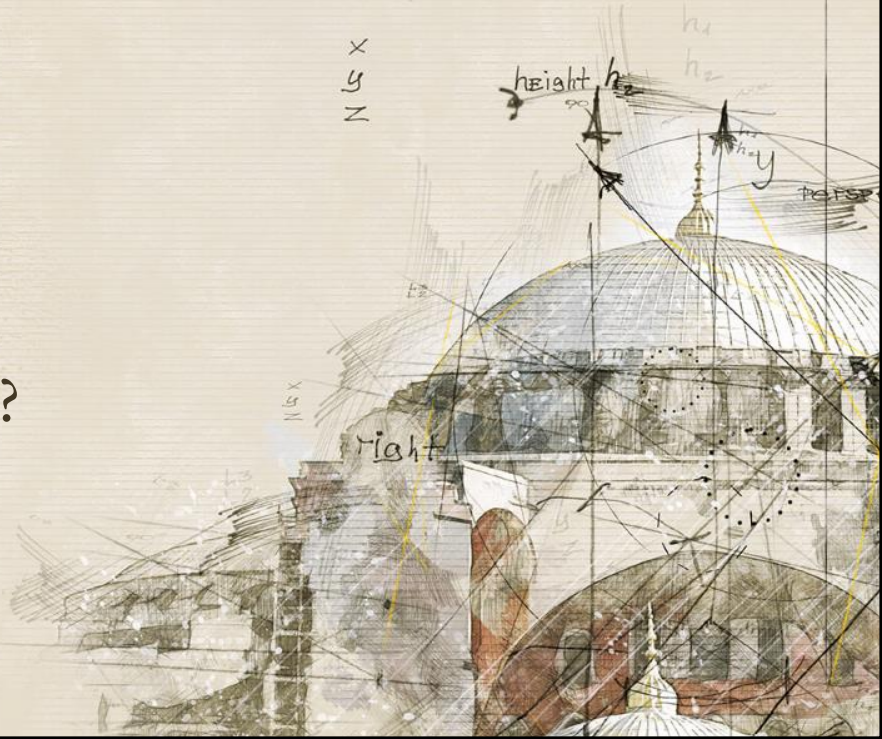
- В обычном приложении:
  - Клиент встроен
  - Сервер один
  - Адрес фиксирован



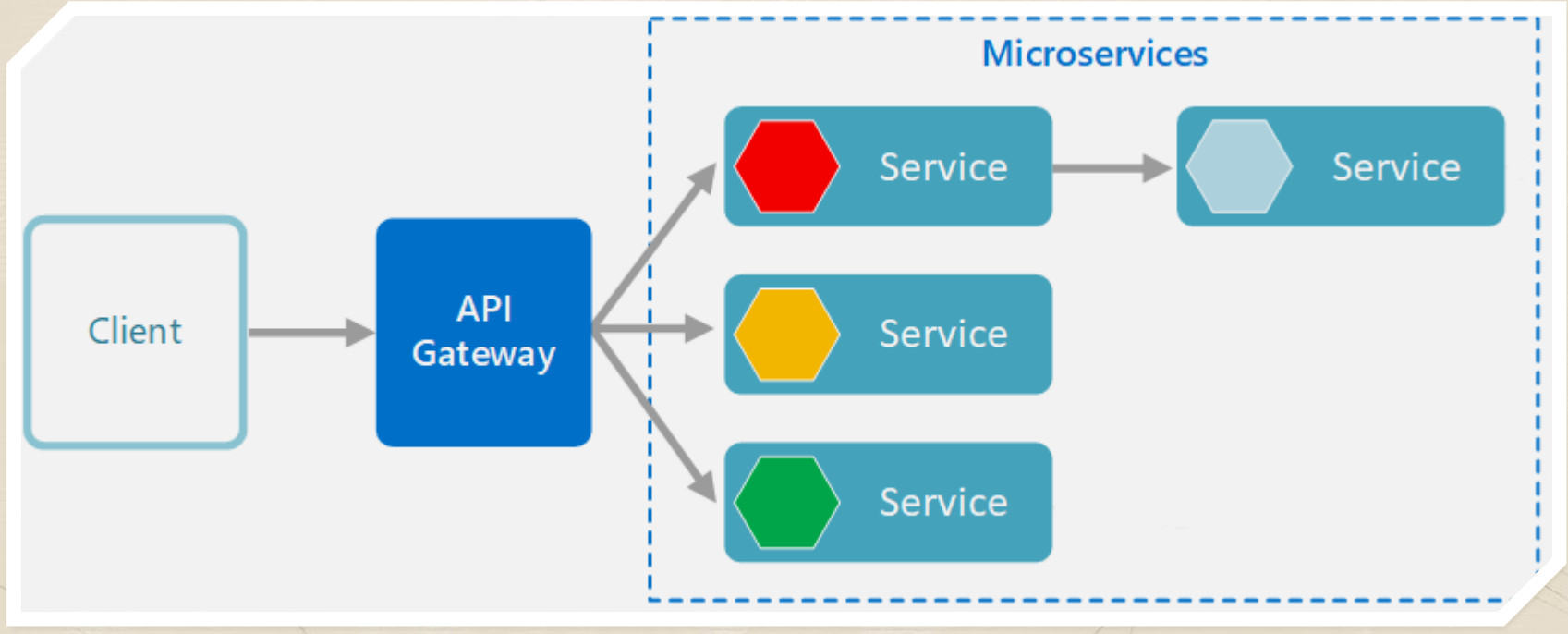

У тебя нет проблем с сервером,
когда им никто не пользуется

1. Как направлять клиентский трафик?

- А что если:
  - Клиентов **много**?
  - Среди них **мобилки**?
  - Серверов **много**?
  - Сервера **реплицированы**?



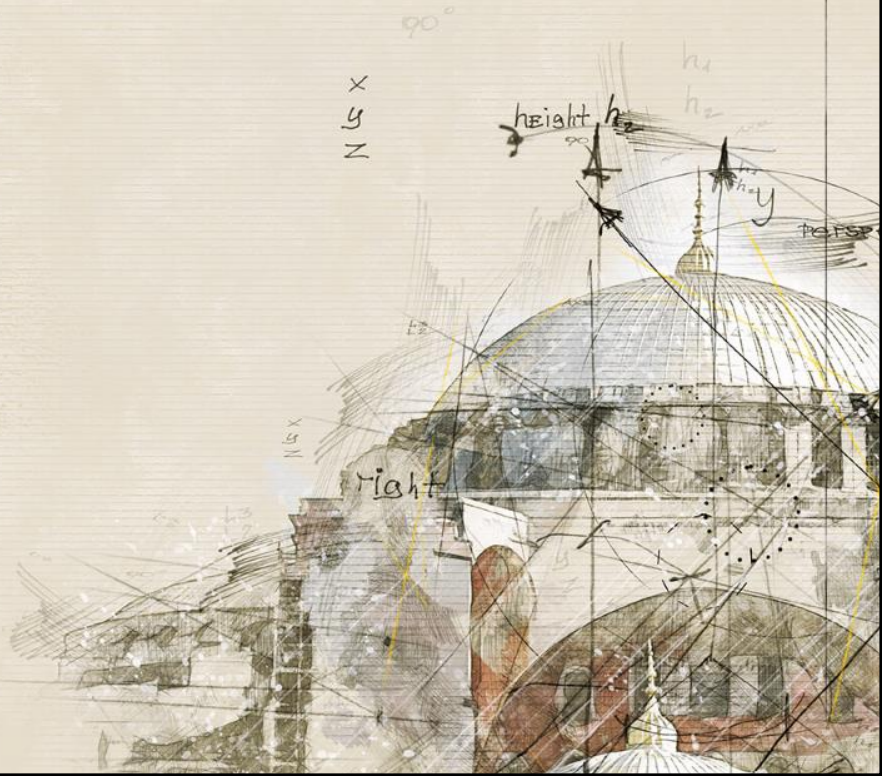
Точка входа (API Gateway)



https://docs.microsoft.com/en-us/azure/architecture/includes/images/microservices-logical.png

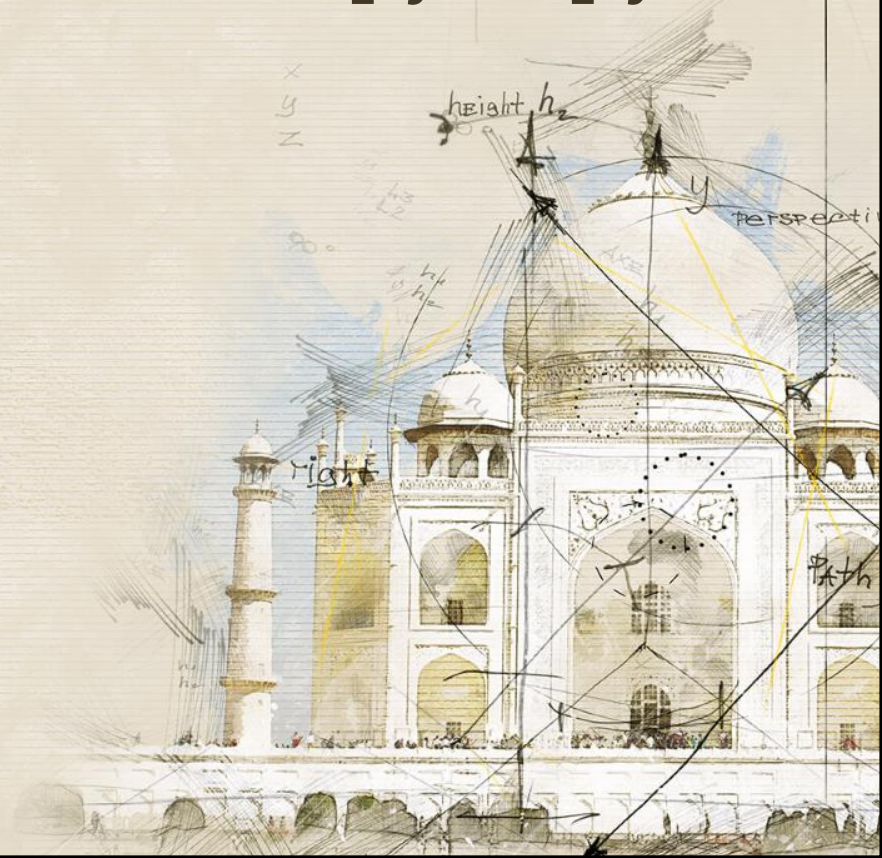
Точка входа (API Gateway) тащит

- Маршрутизацию
- Безопасность
- Трансляцию протоколов
- Кэширование
- Агрегацию запросов
- Мониторинг



2. Как сервисам находить друг друга?

- В обычном приложении:
  - Сервер знает, где он
  - Адрес фиксирован
  - Сервер один

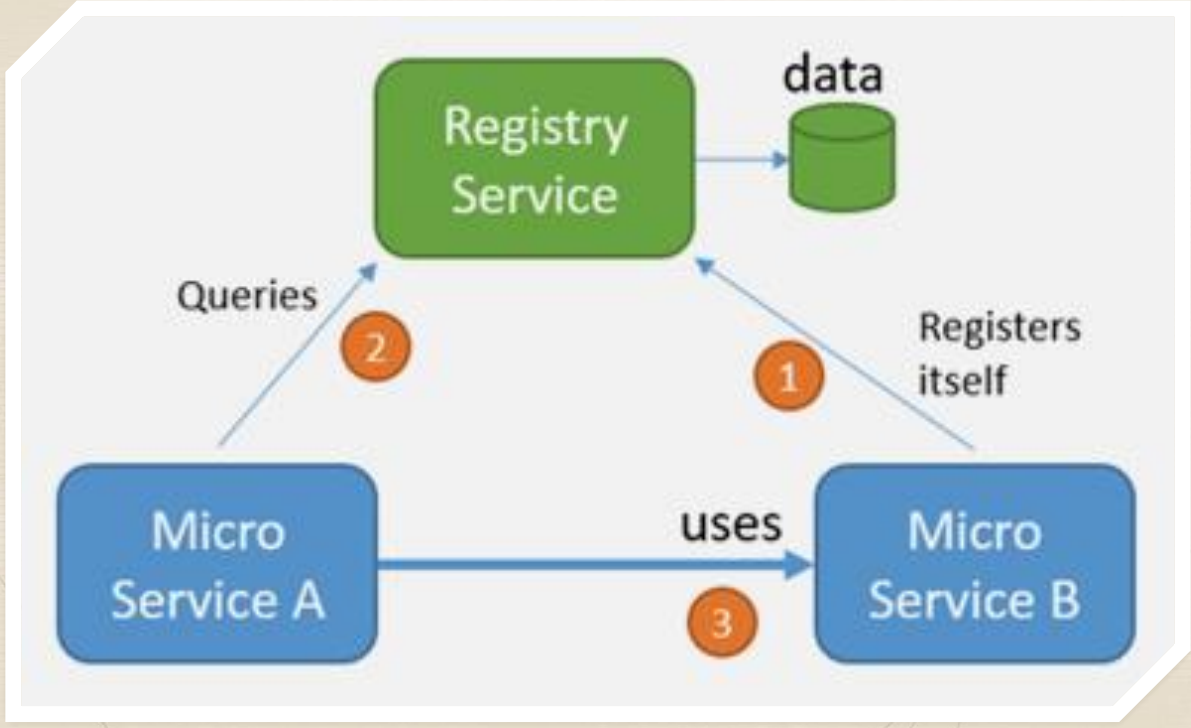


2. Как сервисам находить друг друга?

- А что если:
  - Сервисов **много**?
  - Сервисы **реплицированы**?
  - Адреса **динамические**?

Прописывать вручную уже **не вариант**

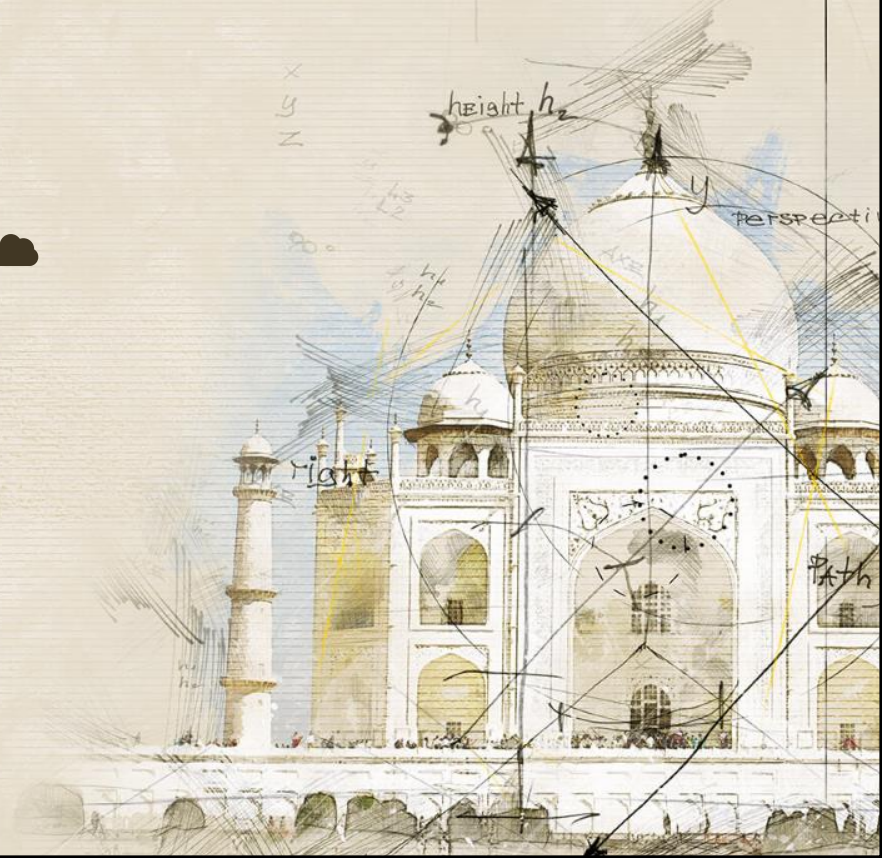
Реестр сервисов (Service Registry)



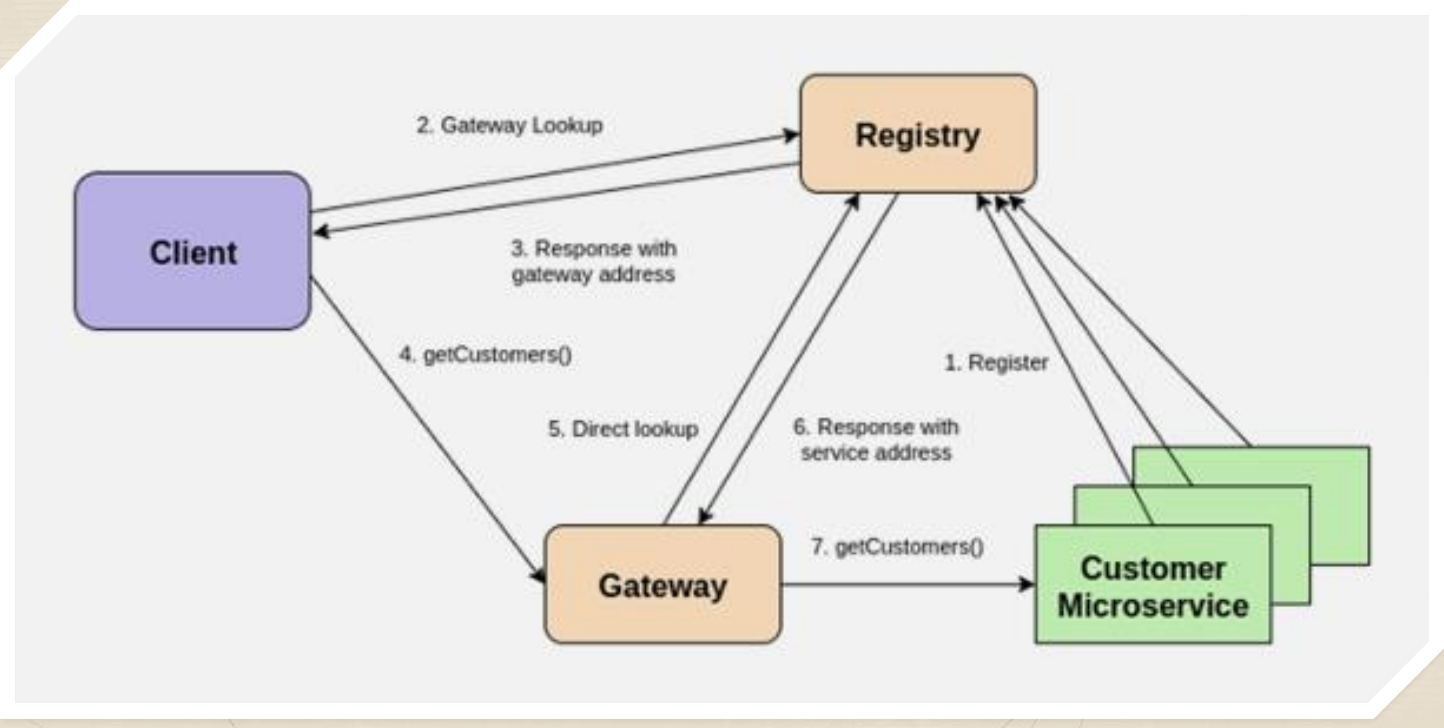
https://blog.mazarin.lk/wp-content/uploads/2016/08/microservices.png

Реестр сервисов помогает

- В **упрощении** конфигурации
- В работе в **эластичной** среде ☁
- В **балансировке** нагрузки
- В **мониторинге**



Реестр сервисов участвует



https://www.eclipse.org/community/eclipse_newsletter/2017/september/images/image006_sm.png

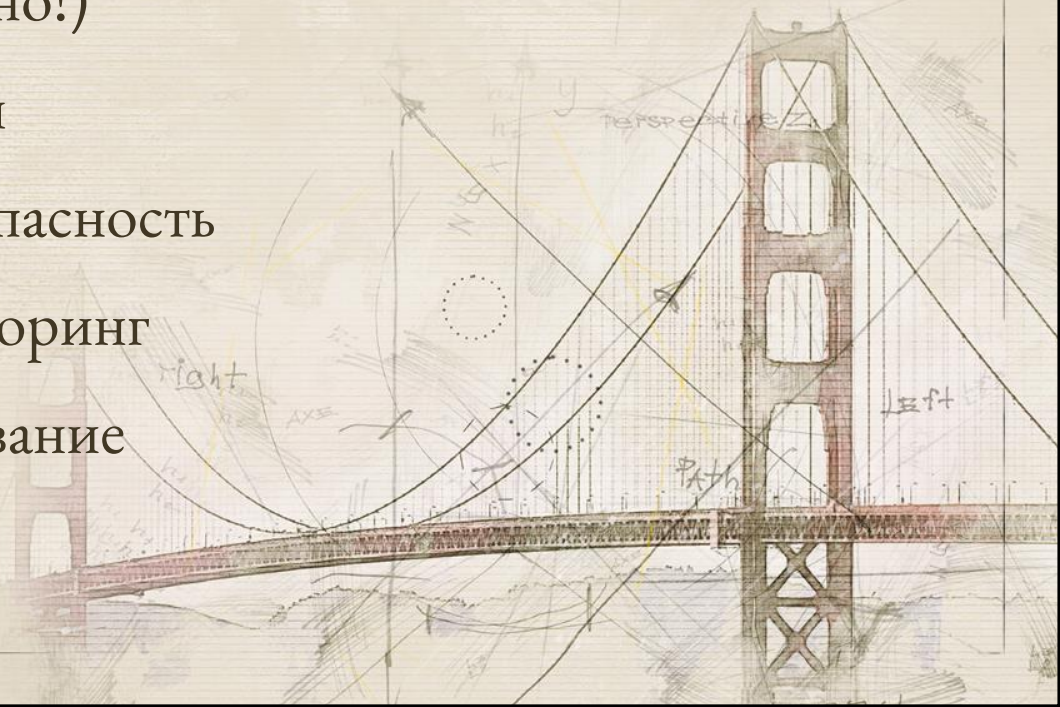
Итог: +2 сервиса на ровном месте!

- Каждый надо **разработать***!
- У каждого свой **жизненный цикл**
- Каждый требует **поддержки**
- И это еще **не всё**!

** обычно на основе библиотеки*

А для частых релизов ещё нужно…

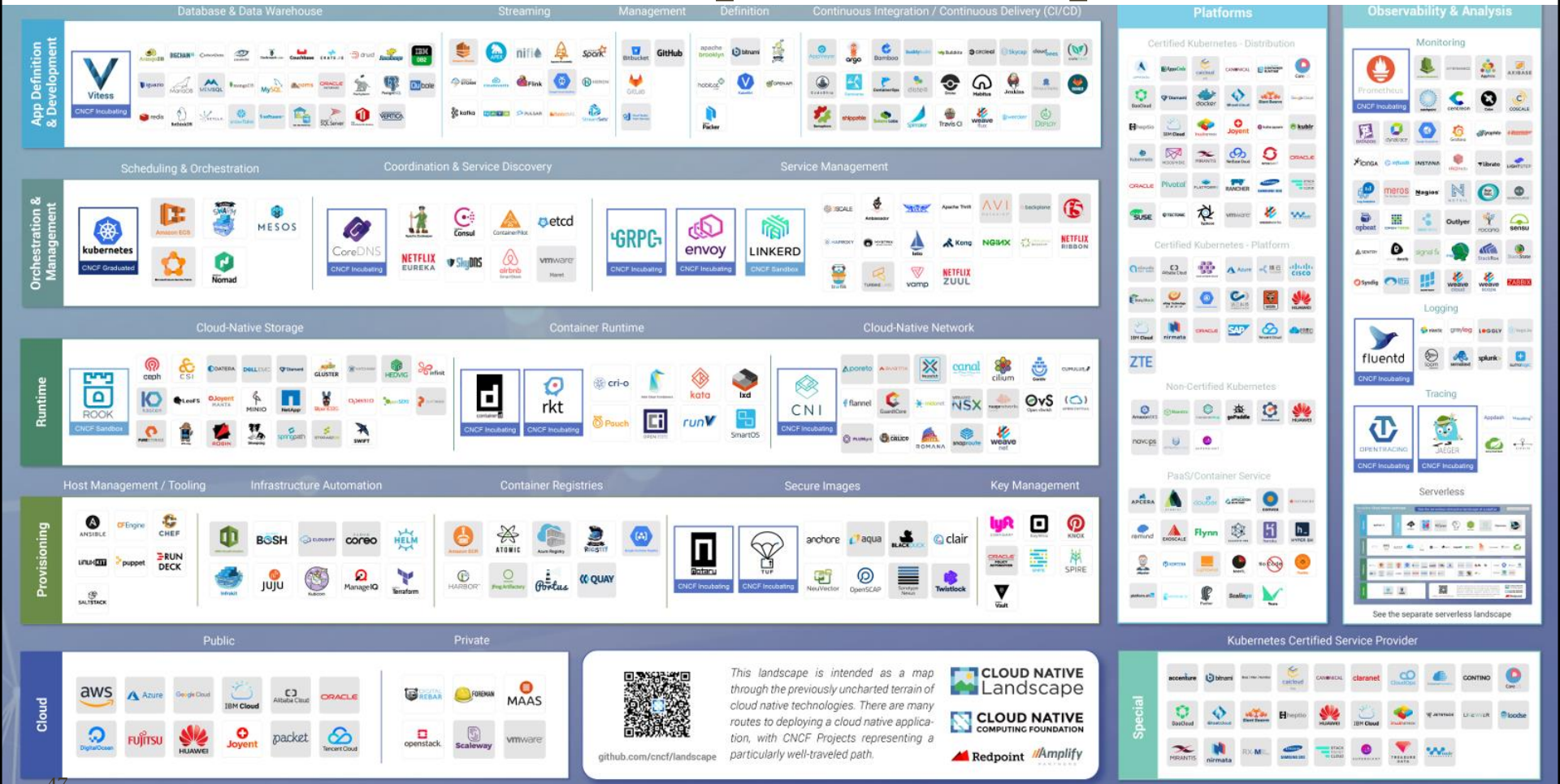
- **Тестировать** (постоянно!)
- **Учитывать** окружения
- **Контролировать** безопасность
- **Предоставлять** мониторинг
- **Обеспечивать** логирование



А для частых релизов ещё нужно…

- Continuous Integration & Delivery (**CI/CD**)
- Контейнеризация (**Docker**)
- Оркестрация (**Kubernetes**)
- Культура (**DevOps**)

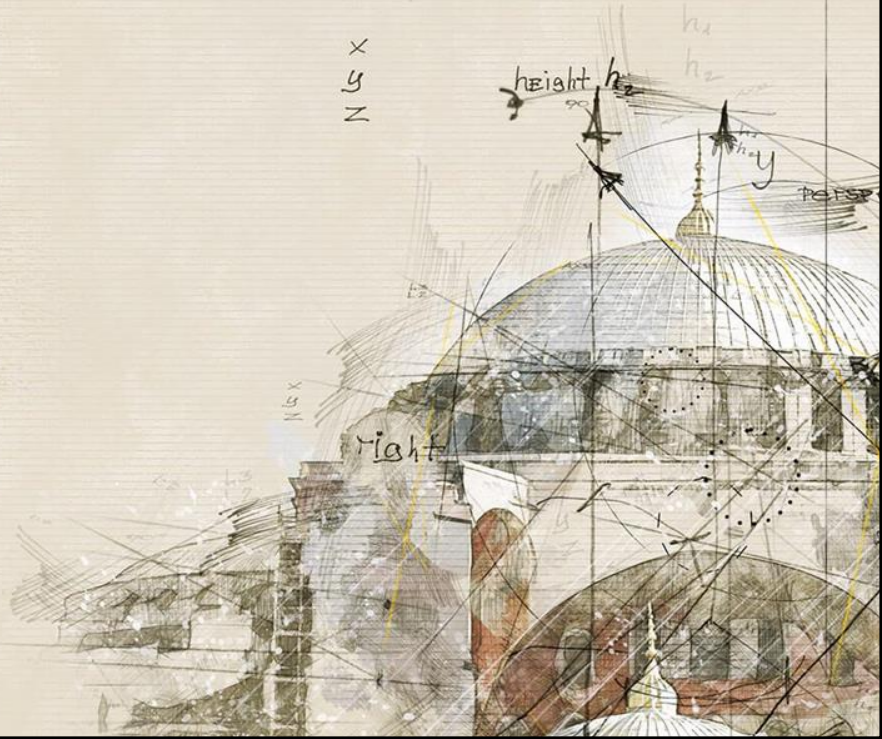
А ещё для частых релизов пригодится…




KUBERNETES
LIGHTYEAR
KUBERNETES EVERYWHERE
imgflip.com

5. Примеры

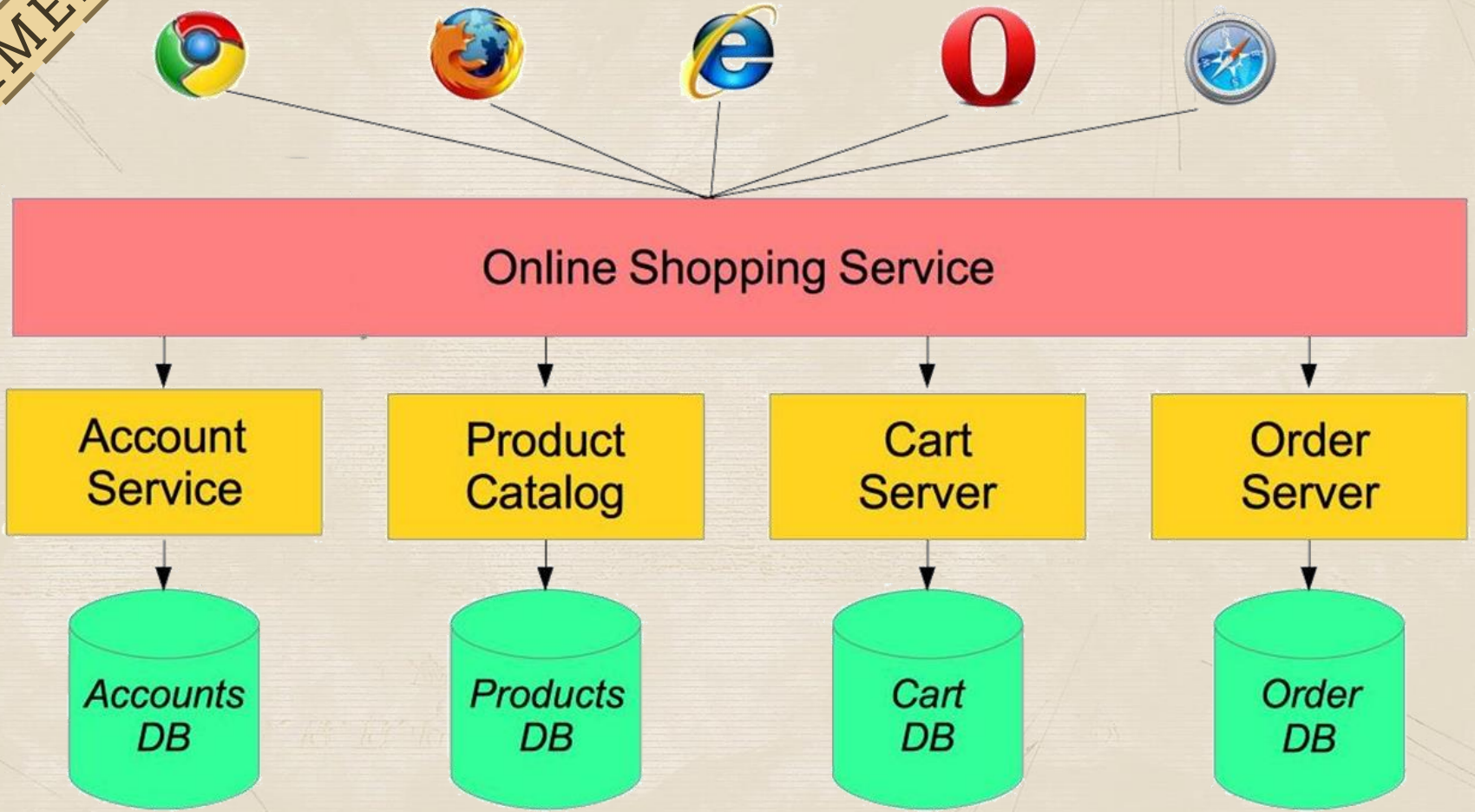
Связь с реальным миром



Да кому это вообще надо?!.

. . .

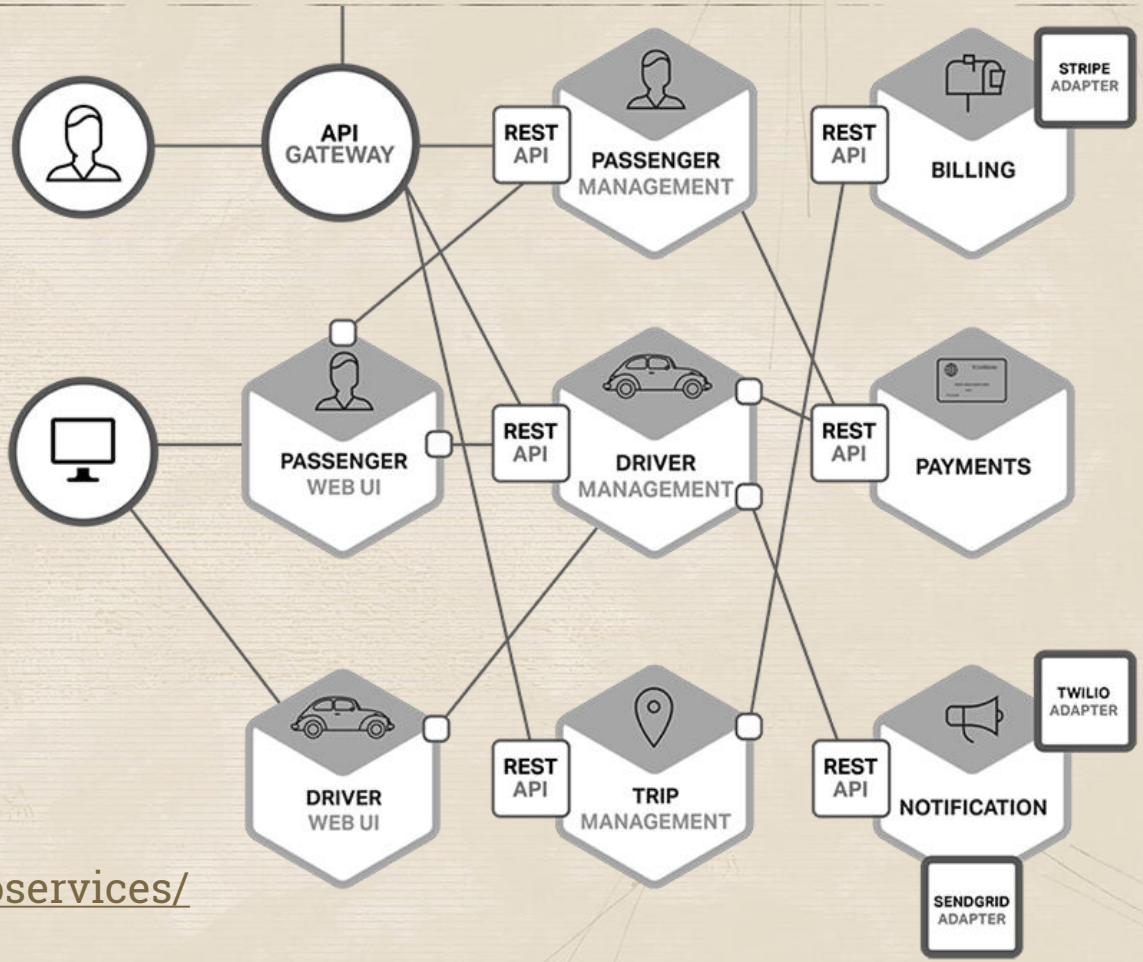
ПРИМЕР



https://spring.io/blog/2015/07/14/microservices-with-spring

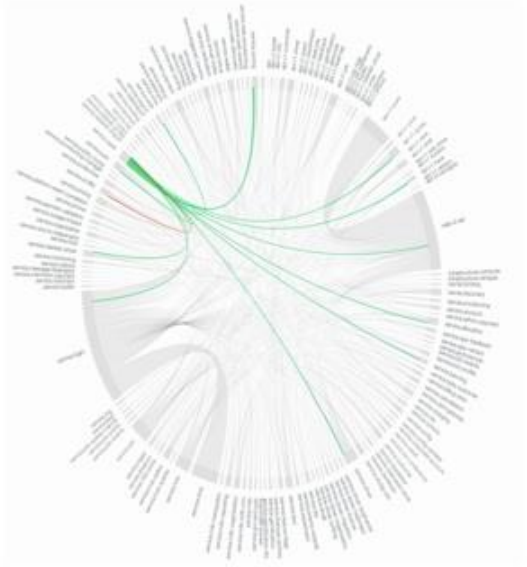
Пример с такси

По мотивам архитектуры UBER'а



https://www.nginx.com/blog/introduction-to-microservices/

450 microservices

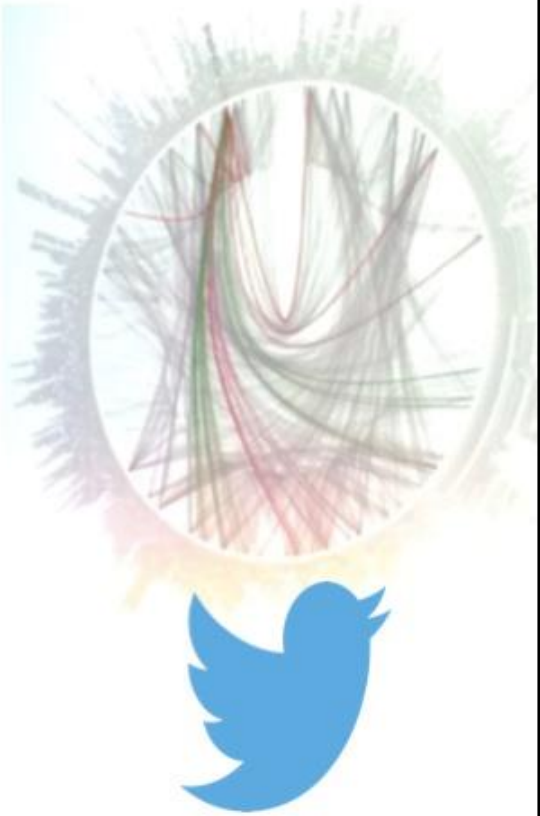


HAIL O

500+ microservices

NETFLIX

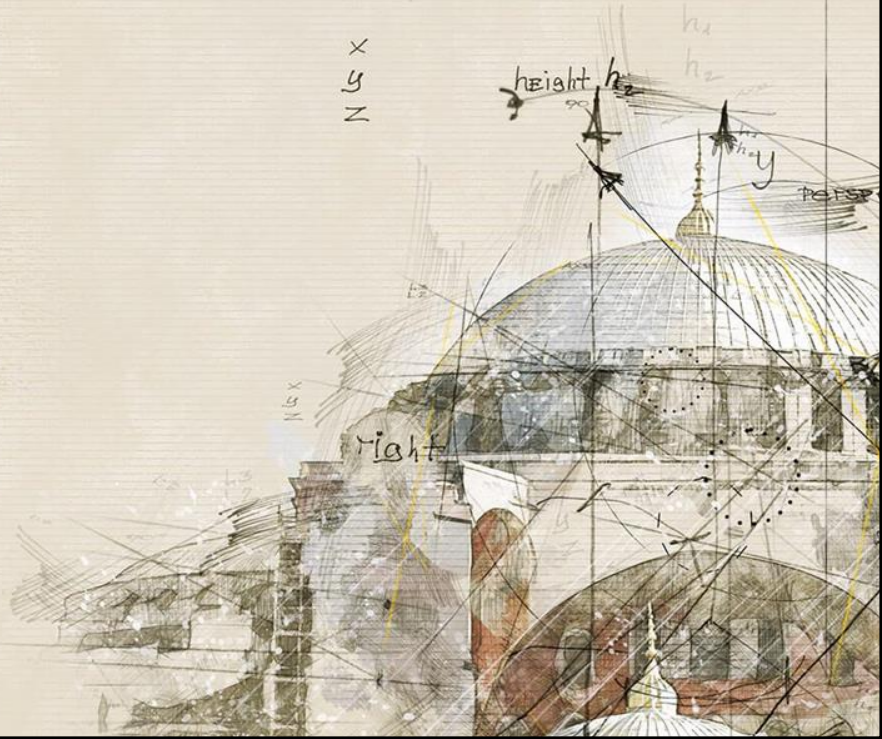
500+ microservices



Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

6. Заключение

Выжимки и выводы



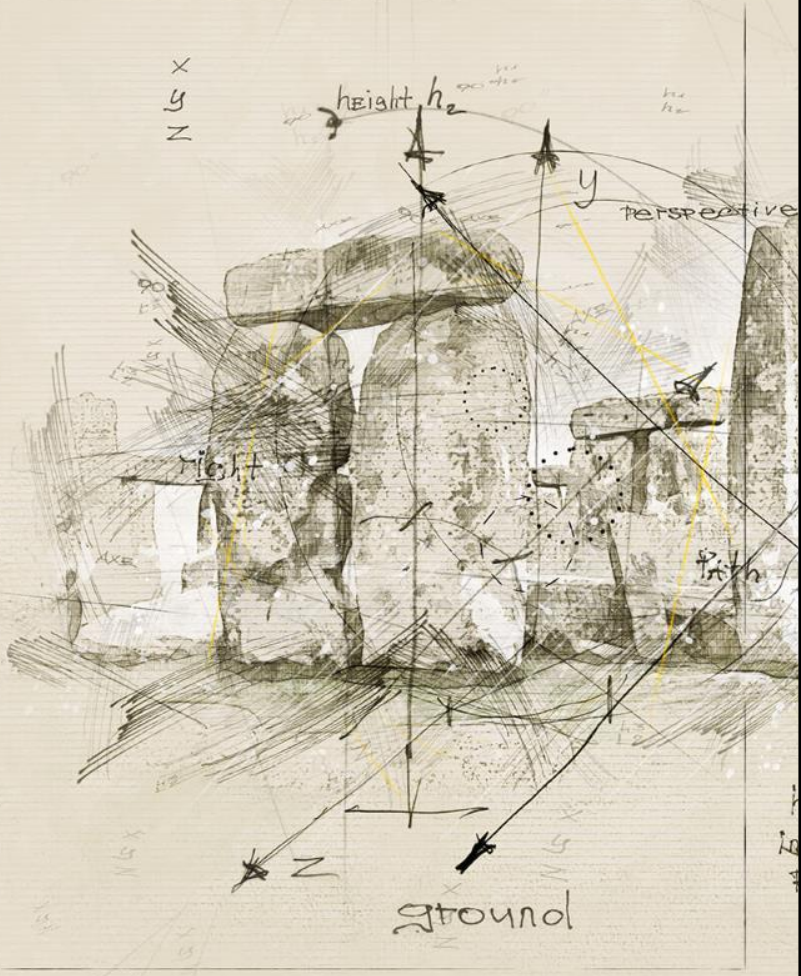
> … you **shouldn't start** a new project **with microservices**, even if you're sure your application will be big enough to make it worthwhile.

Martin Fowler

https://martinfowler.com/bliki/MonolithFirst.html

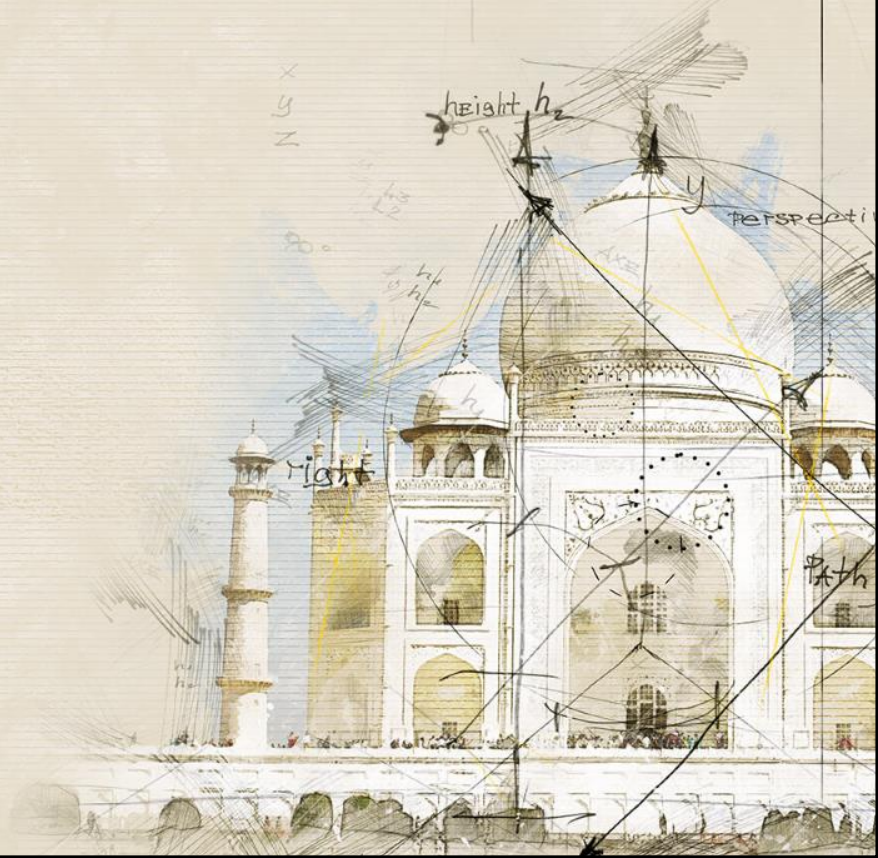
Недостатки MSA

- Накладные **расходы** на сеть
- **Сложность** релиза
- **Трудоемкость** мониторинга
- **Зоопарк** технологий
- ...



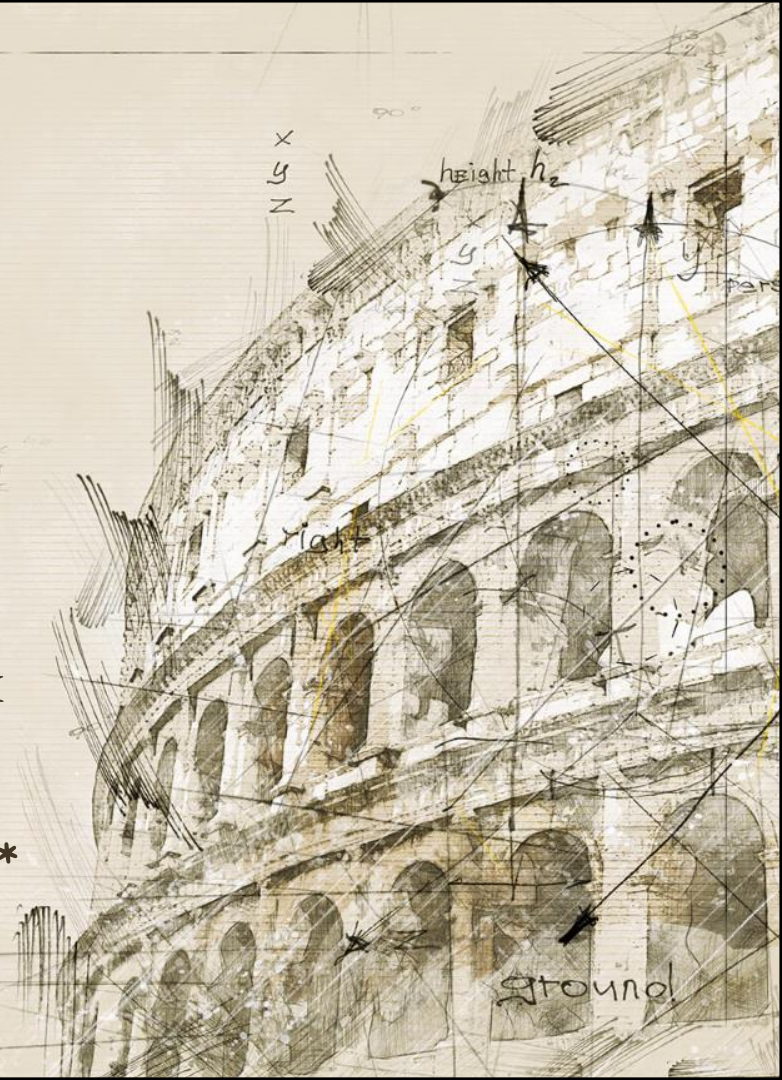
Преимущества MSA

- **Компактность** кода сервисов
- **Изоляция** изменений
- Гибкое **масштабирование**
- Локальность **обновлений**
- Соответствие **командам**
- ...



Выводы

- MSA – **не новшество**
  - и не серебряная пуля
- Высокий **порог** вхождения
- **Единственный** путь развития для одновременно мощных и гибких высоконагруженных приложений*

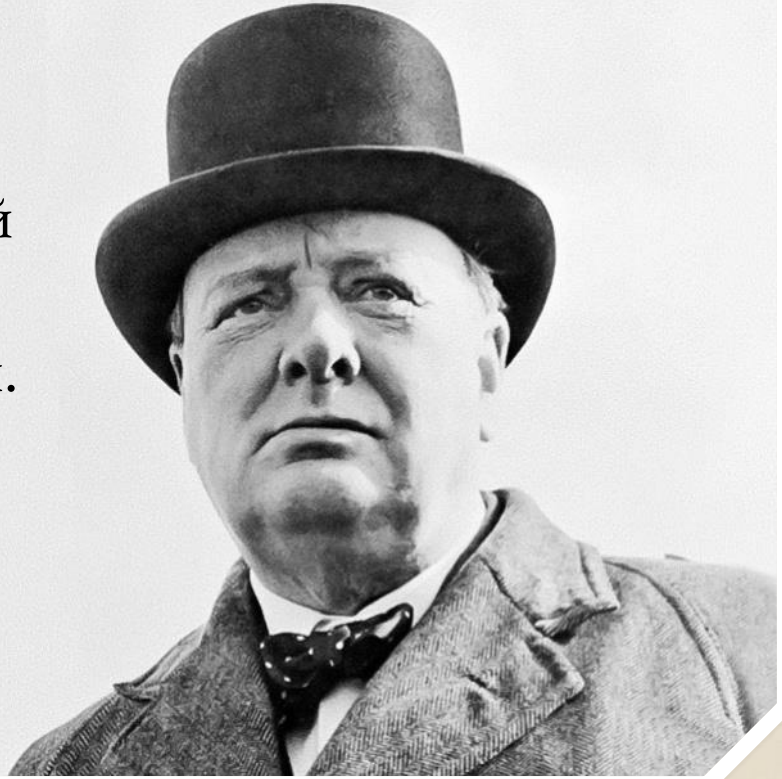


* это и есть ответ на вопрос:

ДЛЯ ЧЕГО ЭТО ВООБЩЕ НУЖНО?

Резюме

Микросервисы – ужасный способ построения сложных веб-приложений. Но лучше пока ничего не придумано.



Спасибо!

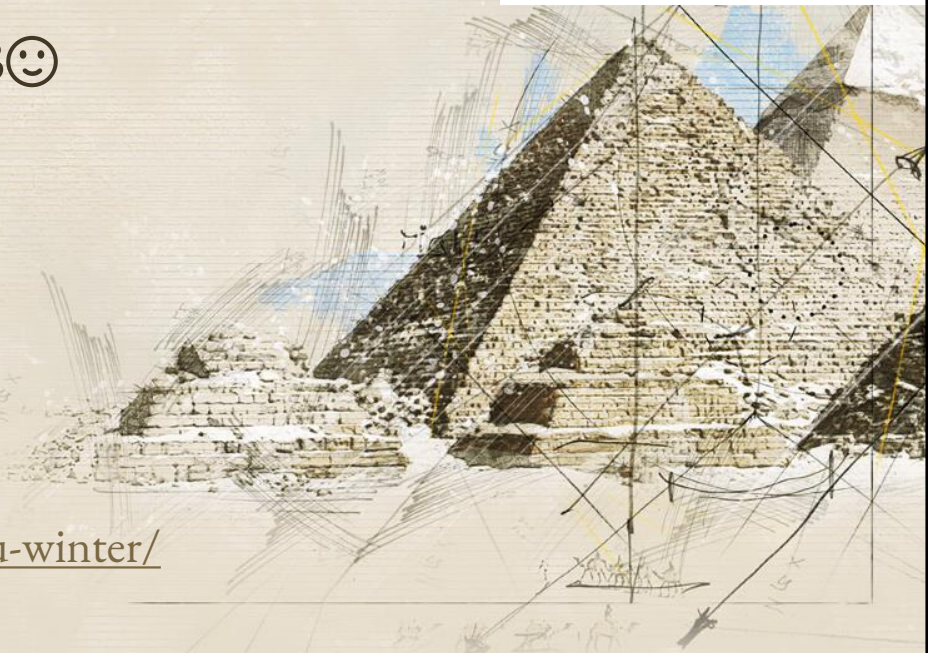
Время для вопросов☺

Владимир Плизгá

@toparvion

Toparvion

https://toparvion.pro/talk/2020/shift-nsu-winter/



Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template and backgrounds by SlidesCarnival
- Photographs by Unsplash & Pexels