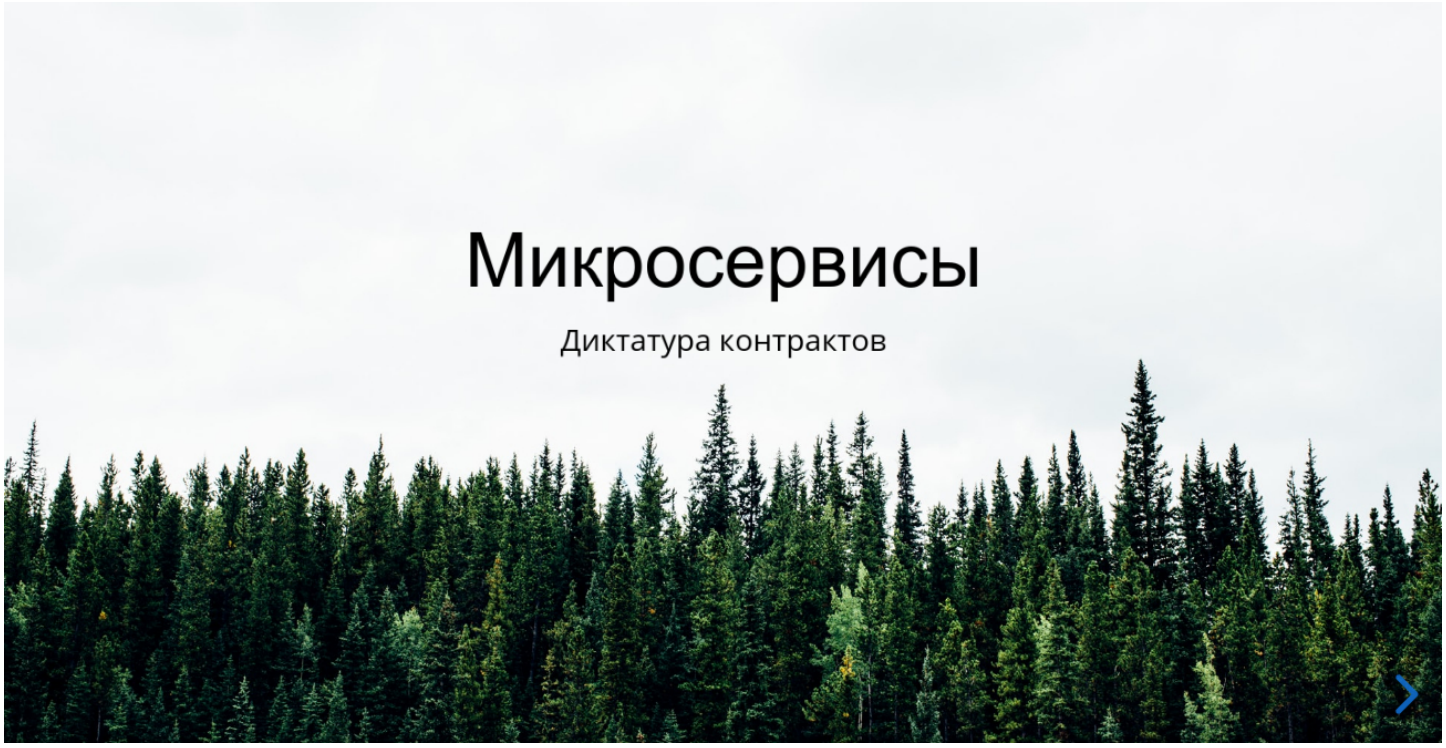
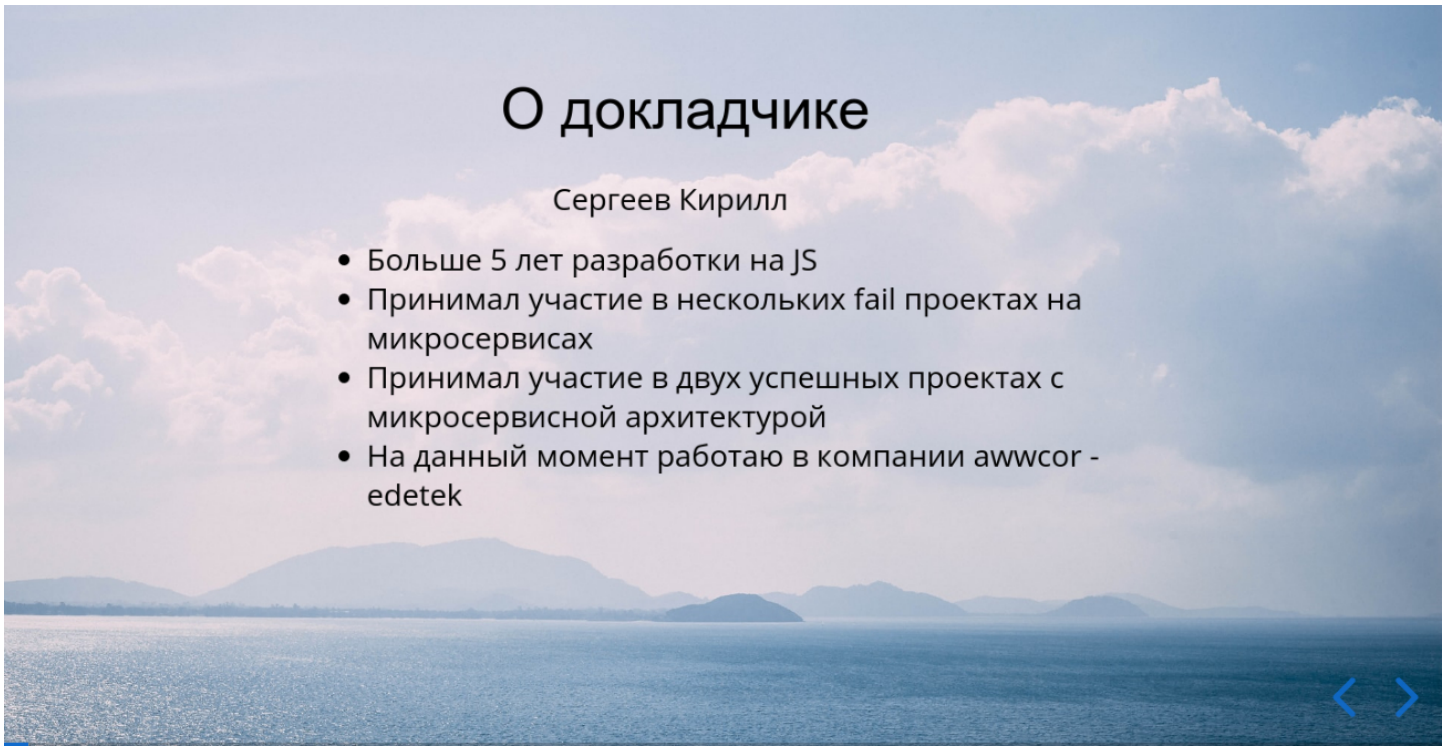

Микросервисы
Диктатура контрактов

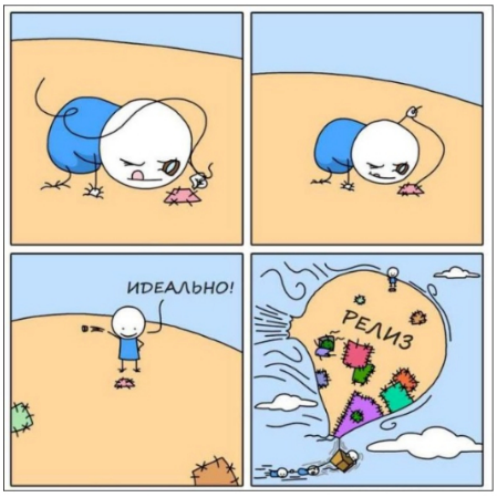

О докладчике
Сергеев Кирилл
Больше 5 лет разработки на JS
Принимал участие в нескольких fail проектах на микросервисах
Принимал участие в двух успешных проектах с микросервисной архитектурой
На данный момент работаю в компании awwcor - edetek

План

1. Зачем нужны микросервисы
2. Плохой пример и минусы
3. Уроки микросервисов
4. Какими технологиями укрощаем
5. Рабочий пример
6. Итоги

Зачем нужны
микросервисы

Все идеально и так



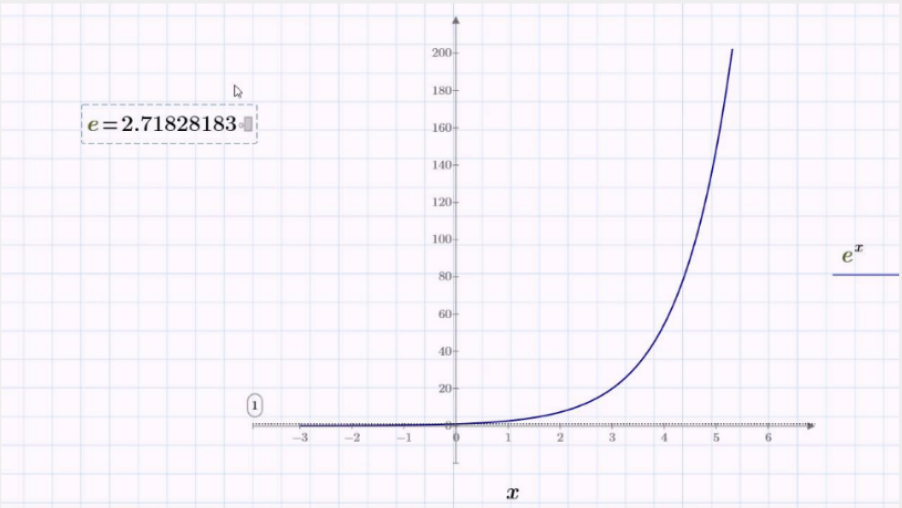
Плюсы микросервисов

- Уменьшенный time-to-market
- Адаптивность к изменениям
- Легкость масштабирования

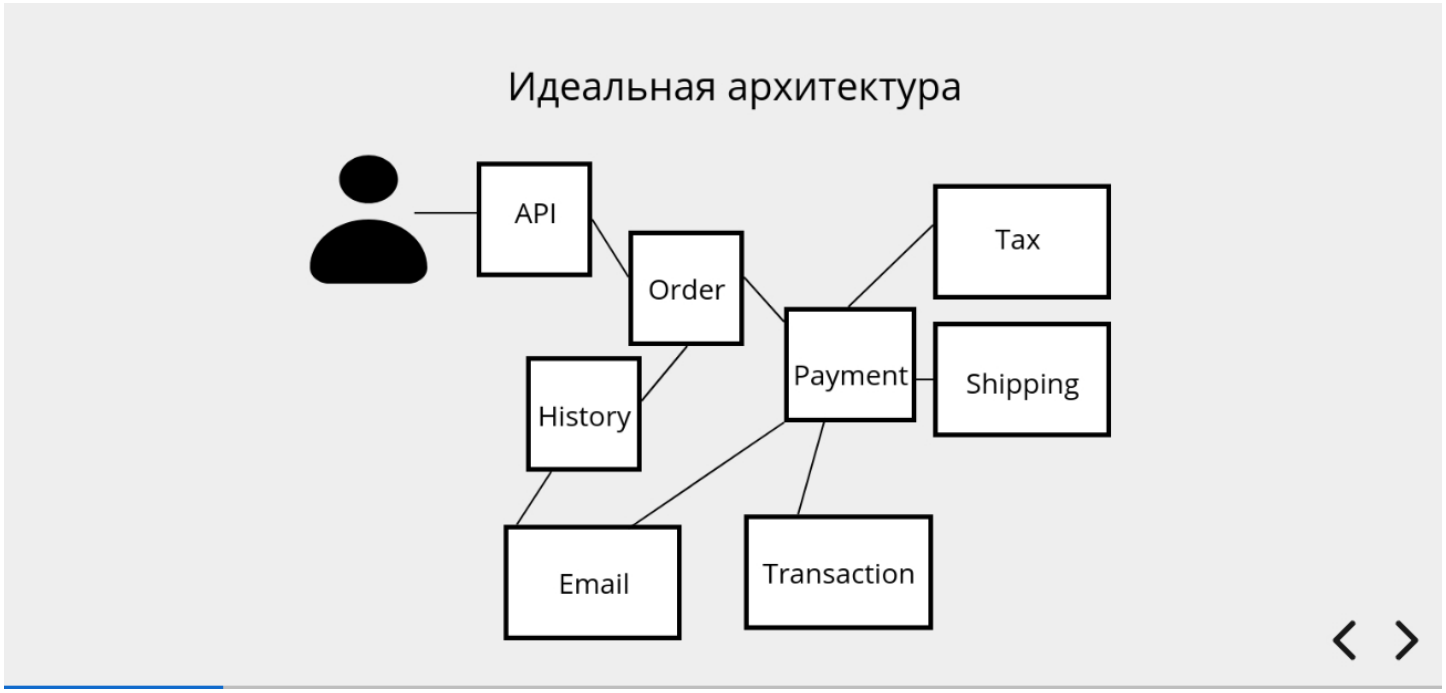
Плюсы микросервисов

- Уменьшенный time-to-market
- Адаптивность к изменениям
- Легкость масштабирования

Экспоненциальный рост проблем относительно сложности


e = 2.71828183
200
180
160
140
120
100
80
60
40
20
1
−3
−2
−1
0
1
2
3
4
5
6
x
e^x

Плохой пример

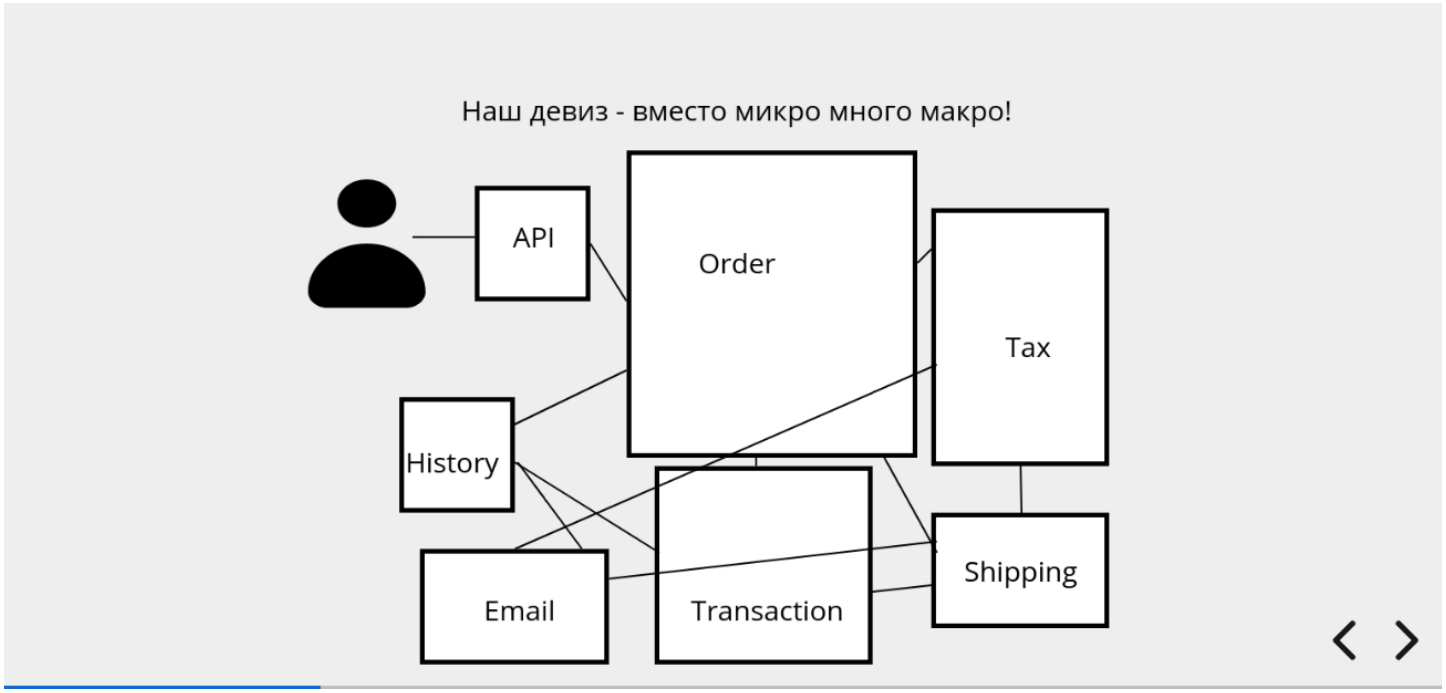

Идеальная архитектура
API
Order
Tax
Payment
Shipping
History
Email
Transaction

Причины хаоса

- Несогласованность взаимодействия сервисов
- Отсутствие определенных границ сервисов
- Возросшая сложность инфрастуктурного кода

Хаос - первые последствия

- Дублирование моделей данных
- Неупорядоченный доступ к хранилищам
- Бессистемное использование кеша
- Увеличение объема ненужного кода


Наш девиз - вместо микро много макро!
API
Order
Tax
History
Email
Transaction
Shipping

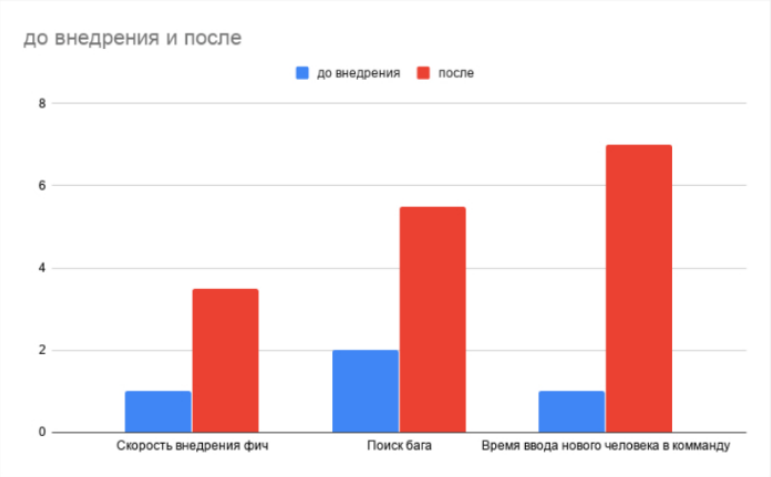
После ускорения

- Fault tolernce tests
- Дублирование логики
- Сильная связанность сервисов
- Возросший объем ненужного кода (инфрастуктура)

Результат

- Тесты на сервисах deprecated
- Возросшее количество deprecated фич
- Возросшая нагрузка на QA инфраструктуру и QA
- Время на поиск и определение ошибок возросло
- Time-to-market стремится к бесконечности
- Возврат?

Статистика

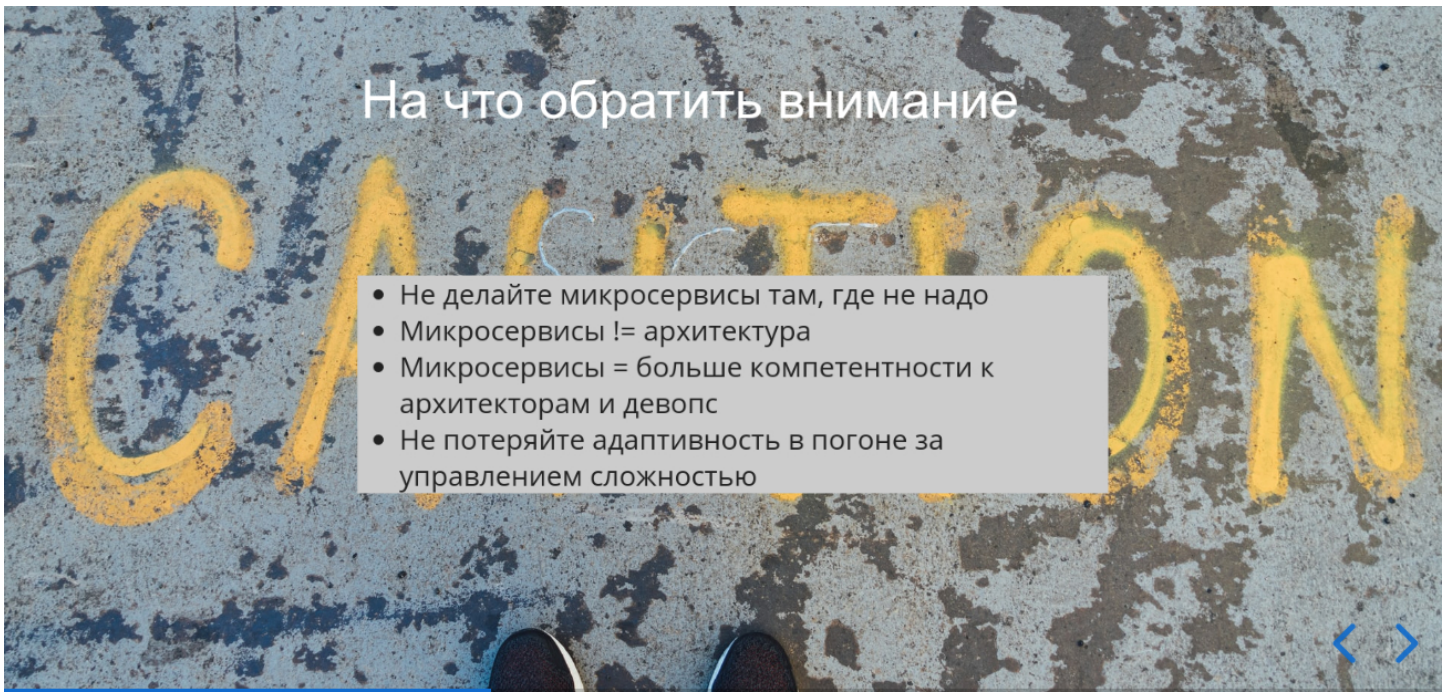

до внедрения и после
до внедрения
после
8
6
4
2
0
Скорость внедрения фич
Поиск бага
Время ввода нового человека в комманду


Хотели как лучше, а получилось как всегда.
(Виктор Черномырдин)
tsitaty.com

Проблемы микросервисов

- Возрастание стоимости обслуживания инфраструктуры
- Проблема централизованных точек
- Проблема взаимодействия сервисов

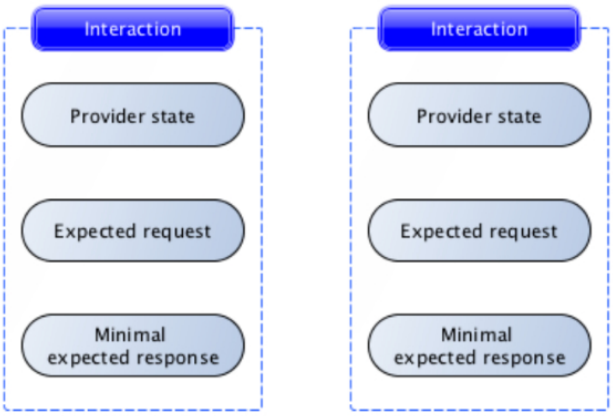
Уроки микросервисов


На что обратить внимание
Не делайте микросервисы там, где не надо
Микросервисы != архитектура
Микросервисы = больше компетентности к архитекторам и девопс
Не потеряйте адаптивность в погоне за управлением сложностью

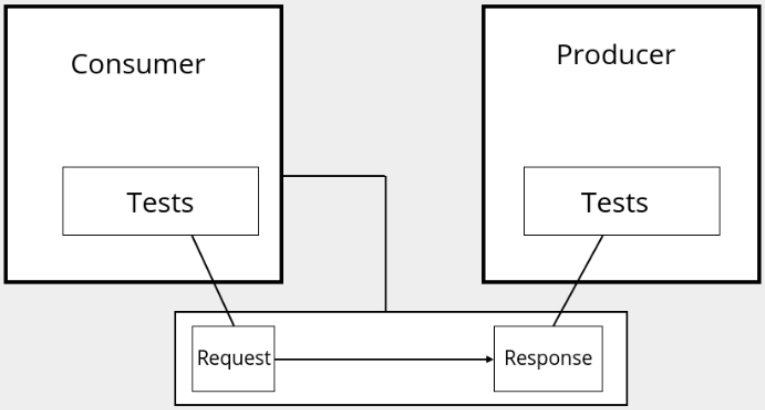

На что обратить внимание
Тестировать должно быть просто и удобно
Тесты и документация должны выводиться
Взаимодействие сервисов должно быть регламентировано и прозрачно

Какими технологиями
укрощаем

PACTS


Interaction
Provider state
Expected request
Minimal expected response
Interaction
Provider state
Expected request
Minimal expected response

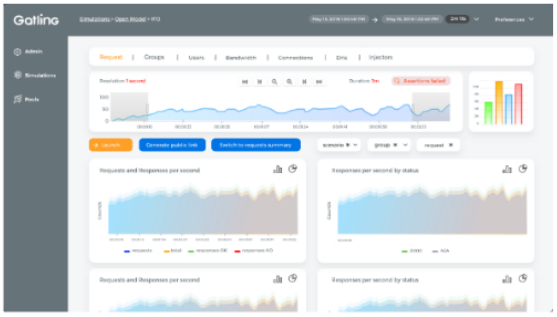
Consumer driver contract



Pact-defined


```
"interactions": [
  {
    "description": "a POST request to create a study",
    "requirement": "Link to BDD scenario",
    "performance": "Link to performance scenario - artillery",
    "providerState": "provider with 0 study",
    "request": {
      "method": "POST",
      "path": "/",
      "headers": {
        "Content-Type": "application/json; charset=utf-8"
      },
      "body": {
        "name": "Superman",
        "superpower": "flying",
        "universe": "DC"
      }
    },
    "response": {
      "status": 201,
      "headers": {
        "Content-Type": "application/json; charset=utf-8"
      },
      "body": {
        "id": 42,
        "name": "Superman",
        "superpower": "flying",
        "universe": "DC"
      }
    }
```


ARTILLERY.IO



Artillery example

```
target: "$CORE/studies"
phases:
  - duration: 60
    arrivalRate: 5
  - duration: 120
    arrivalRate: 5
    rampTo: 50
payload:
    path: "general_studies.csv"
tags:
    - "normal_speed"
```


continuumsecurity
IriusRisk
Open Source
Blog
Resources
Guides
About Us
Contact
BDD-Security
BDD-Security is a security testing framework that uses natural language in a Given, When, Then Gherkin syntax to describe security requirements as features. Those same requirements are also executable as standard unit/integration tests which means they can run as part of the build/test/deploy process.
GET STARTED NOW
BDD-Security
Key Features
Free and Open Source automated testing framework for security
Ready to run on a Continuous Integration Server , as part of the build/test/deploy process
Upgrade DevOps to SecDevOps
Generate reports, to easily be viewed and understood by business and security users
Tests are run dynamically against a deployed application, no need to access your source code

BDD Gherkin

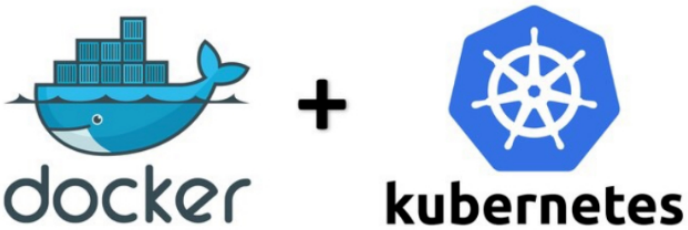
```
Feature: Create study for division

  Scenario: Create a study auth users
   Given the zero studies in db
     And the User is authorized
     And the User has writer role
    Then the User send name and STUDY data
     And the one studies in db
     And the created study is returned
```

Линтеры

Example of lint file

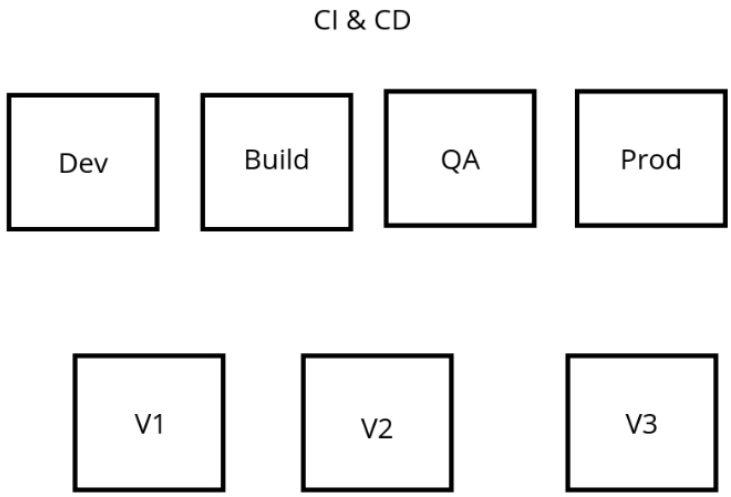
```
{
  "defaultSeverity": "error",
  "linterOptions": {
    "exclude": ["**/*.js"]
  },
  "extends": ["tslint:recommended", "tslint-eslint-rules"],
  "jsRules": {},
  "rules": {
    "quotemark": [true, "single", "avoid-escape", "avoid-template"],
    "object-curly-spacing": true,
    "array-bracket-spacing": [true, "never"],
    "cyclomatic-complexity": [true, 5],
    "arrow-parens": [true, "ban-single-arg-parens"],
    "space-within-parens": true,
    "variable-name": [true, "ban-keywords", "allow-leading-underscore"],
    "trailing-comma": false,
    "only-arrow-functions": false,
    "prefer-const": [true, { "destructuring": "all" }],
    "no-string-literal": false,
    "forin": false,
    "no-bitwise": false,
    "no-console": [true, "log", "error"],
    "object-literal-sort-keys": false
  },
  "rulesDirectory": []
```

DevOps

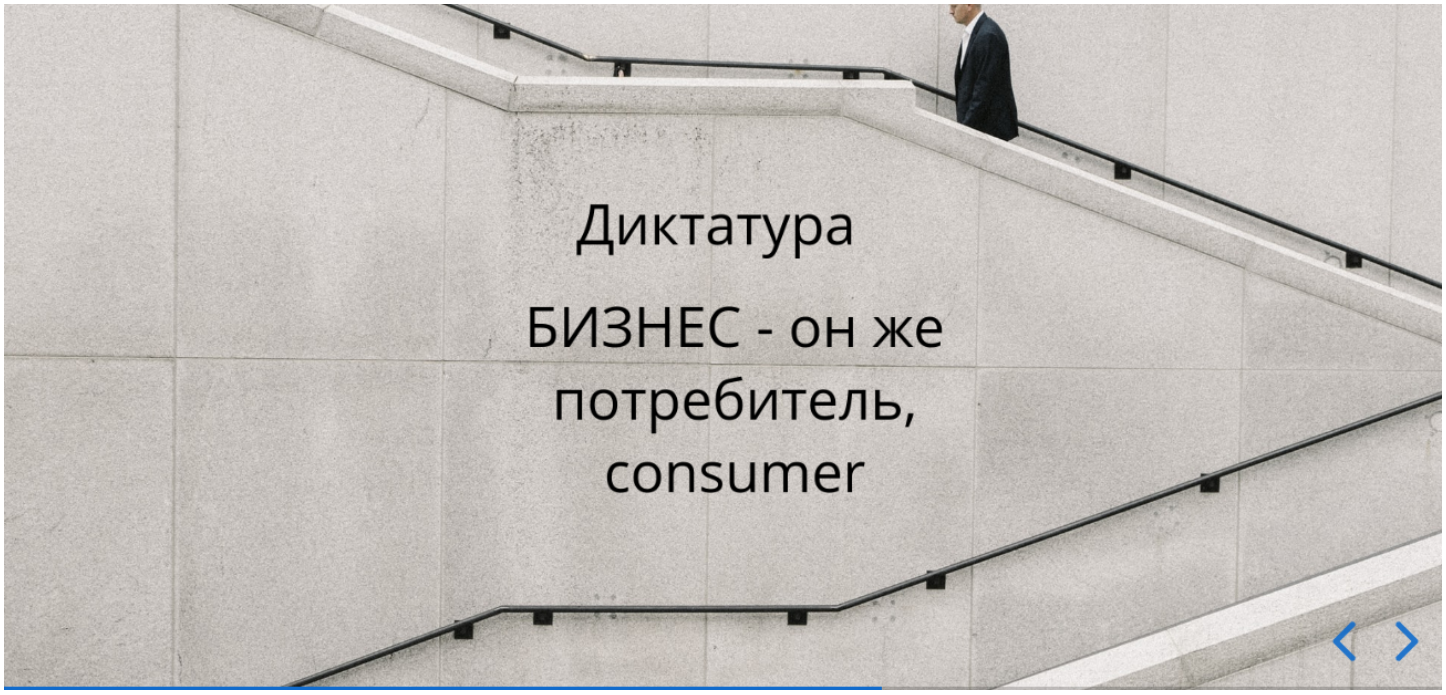


Example of Dockerfile

```
FROM node:8
ENV NETWORK_TYPE DEFAULT_NETWORK_TYPE
ENV NPM_CONFIG_LOGLEVEL warn
ARG RELEASE=latest

RUN apt update && \
    apt install -y python make g++ git build-essential && \
    npm install -g pm2@2.7.1 && \
    mkdir /app
WORKDIR /app

RUN npm install -g chronobank-middleware --unsafe
RUN mkdir src && cd src && \
    dmt init && \
    dmt install middleware-check-bot"#$RELEASE"
CMD pm2-docker start /mnt/config/${NETWORK_TYPE}/ecosystem.config.js
```


CI & CD
Dev
Build
QA
Prod
V1
V2
V3

Рабочий пример


Диктатура
БИЗНЕС - он же потребитель, consumer

Контракт

- Бизнес требование (цель)
- Параметры интерфейса (подробности задачи)
- Параметры производительности (за сколько)

Контракт

- Описание качество кода (параметры линтера)
- Описание архитектуры (тип + линтер)
- Описание параметров QA (процент покрытия + линтер)
- Описание документации (параметры линтера)

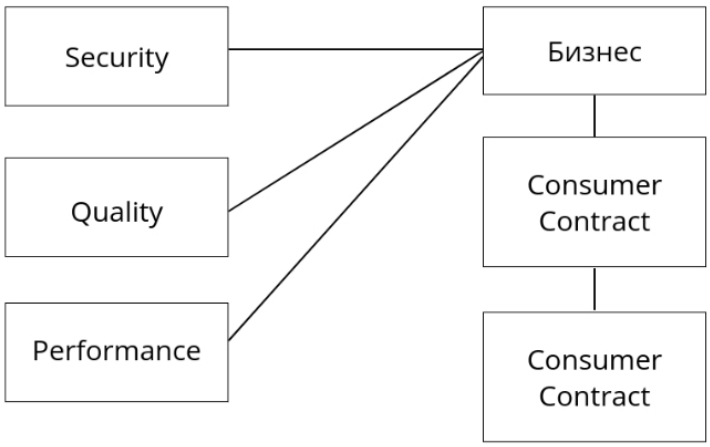
Quality Contract

```
Quality:
   - high_tslint.ts
Architerture:
   - high_layer_architecture.ts
Tests:
   - percent_confirmation: 60
  - tests_high_functional_tslint.ts
Documentation:
   - swagger: 60
```

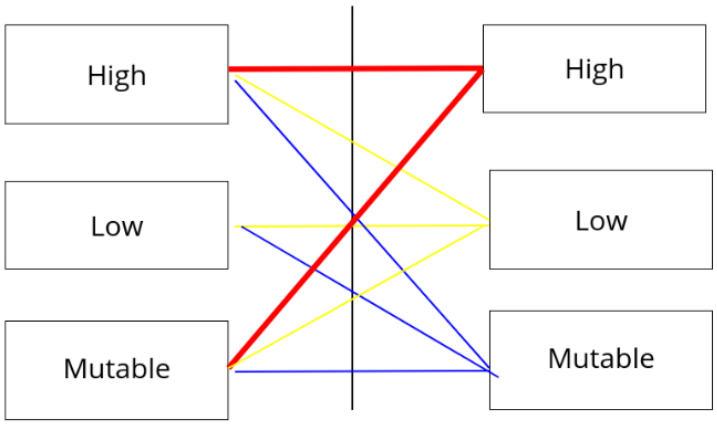
Security Contract

```
Security:
   - permission_level: data_manager
  -  percent_security_data: 65
  - allowed_keywords: studies_keywords.ts
  - user_data:
     - access: allow
     - return:  deny
MutationTests: numbers_false.ts
```

Цепочка контрактов

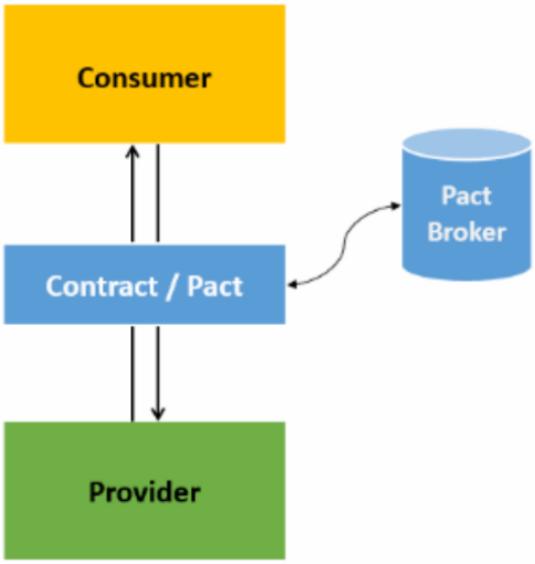


Иерархия контрактов



Брокер контрактов

- Хранилище контрактов
- Производный от Pact Broker
- Каждый контракт имеет свой статус (Draft - InProcess - Реализован - Deprecated)
- Доступен всюду и всегда


Consumer
Pact
Broker
Contract / Pact
Provider

Публикация в broker

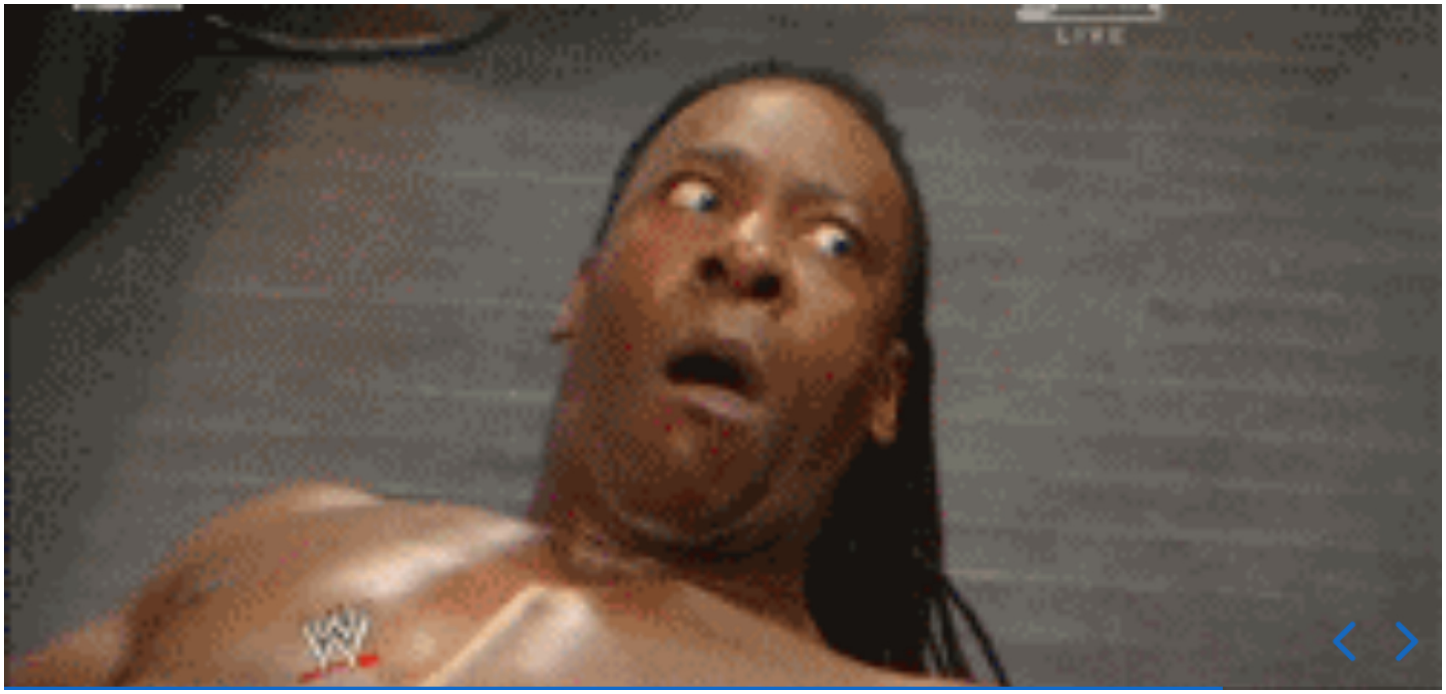
- Какие контракты уже есть
- Есть ли пересечения по именам
- Версия контракта инкрементом
- Контракт нижнего уровня одной версии не может обеспечивать качество хуже чем контракт высшей версии
- Связанность контрактов

Процесс разработки микросервиса

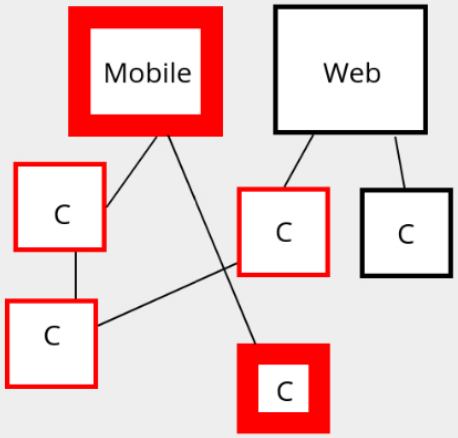
- Написание контракта
- Генерация тестов
- Создание Producer
- Публикация в Broker

Авто деплой

- Сервис проверен линтерем на качество
- Сервис проверен по контрактам producer
- Сервис проверен по контрактам consumers
- Сервис проверен по тестам
- Сервис проверен по безопасности
- Сервис проверен по производительности
- Сервису присвоен номер, тег, автор, контракт
- Сервис вылит и ответил


LIVE

Суть работы



Знание по контракту

1) Формат логов
2) Производительность
3) Интерфейс
4) Стиль кода

Свобода есть осознанная необходимость (Бенедикт Спиноза)

Преимущества системы

- Согласование сервисов
- Deprecated становится явным
- Единый источник требований, проверки и интерфейсов

Преимущества системы

- Ленивая генерация кода
- Синхронизация кода с контрактами
- Все становится кодом - безопасность, devops, qa
- Хорошо расширяемая система

Недостатки системы

- Пионеризм
- Отсутствие готовых решений
- Поддержка новой инфраструктуры
- Время с разработки перекладывается на согласование
- Бюрократия! Бюрократия! Бюрократия!

Результат

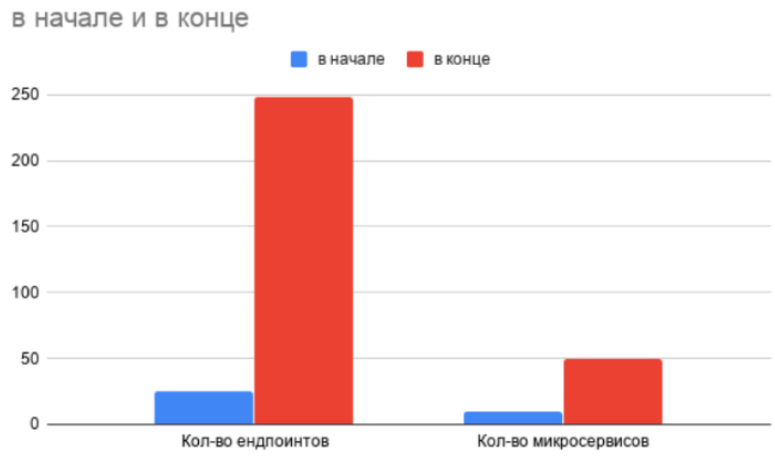
- Контролируемый time-to-market
- Сменяемость комманд без ущерба качества код
- Контролируемое качество кода

Time-to-market - возврат к 1 часу

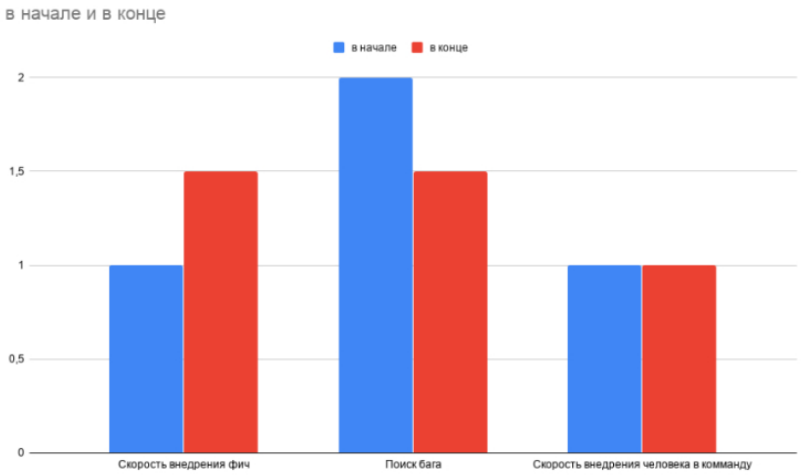
Мы не победили хаос

Мы приблизились к его контролю

Возросла сложность



Статистика


в начале и в конце
в начале
в конце
2
1,5
1
0,5
0
Скорость внедрения фич
Поиск бага
Скорость внедрения человека в комманду


Q&A
https://twitter.com/cloudkserg
https://piterjs.org
https://docs.pact.io/
https://intuit.github.io/karate/
https://gatling.io/
https://www.asyncapi.org/
https://raml.org/
https://continuumsecurity.net/bdd-security/