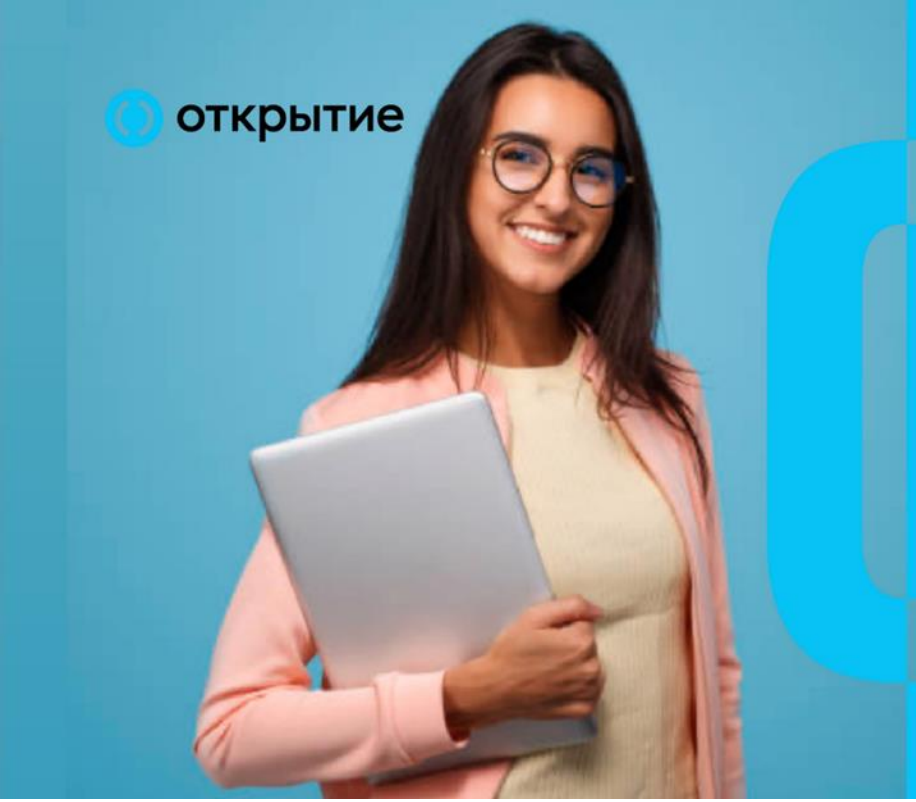


ИТ-архитектура банка

Лекция 1: Понятие микросервисной архитектуры

открытие | Банк

О лекторе

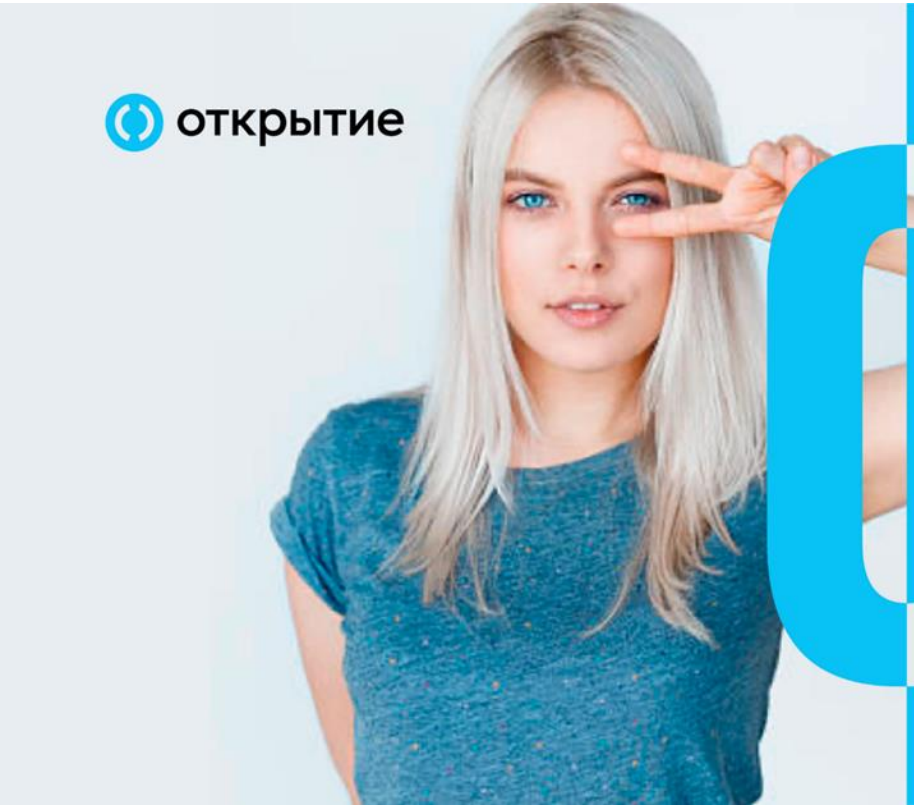
Андрей Зорин
Начальник управления
Банк Открытие

Более 15 лет профессионального опыта в ИТ и менеджменте:

- Начинал как разработчик. Работал как в маленьких командах, так и в международных проектах
- В 2013 году перешел на «темную сторону», в банк. Работал как линейным руководителем, так и руководителем проектов
- С 2015 года в Банке «ФК Открытие»
- С 2018 года отвечаю за развитие Единой Сервисной Платформы розничного блока (eg. Digital Platform)
- Кроме платформы, отвечаю за разработку всех каналов обслуживания розничных клиентов (интернет и мобильный банк, Единое Фронтальное Решение, сайт банка и т.д.)

Есть ряд достижений и наград, среди которых:

- Лучший интернет банк 2017-2018 годов (Markswebb)
- 2-3 место среди мобильных банков 2017-2018 (Markswebb)
- Лучший проект года 2019 за ЕФР (внутренняя награда банка Открытие)
- Лучшее внедрение цифровой платформы в розничном банке 2020 (Comnews)



0,

Обзор курса

Темы и контрольные мероприятия

Введение в дизайн микросервисов

открытие | Банк

- Понятие микросервисной архитектуры и SOA.
- История появления. Отличия от ранних решений. Предпосылки и решаемые проблемы. Недостатки. Опыт Банка и причины внедрения.
- Дизайн микросервисов. Доменная модель. Ошибочные применения.
- Транзакции в микросервисах. Паттерн Saga.
- Консолидация данных. Агрегаты и reporting. Event Sourcing.

Реализация микросервисной архитектуры

- Версионирование и поддержка целостности микросервисной системы. Устранение ада зависимостей на примере решения Банка.
- Коммуникация сервисов между собой. Почему важны стандарты и какие они бывают. CDC (client driven contracts) и кононические модели данных. Under- и Overfetching данных.
- GraphQL как современный язык разработки контрактов микросервисов. Многоязычные среды. Альтернативы и проблемы.
- Событийная модель и брокеры сообщений.
- Ролевая модель и управлени доступа. Доступ клиент-сервер и сервер-сервер. Токены, API key, использование различных каналов аутентификации.

Упаковка и развертывание микросервисов

- Зачем нужна контейнеризация. Краткое введение в контейнеры и Kubernetes.
- Sidecar как модель переиспользования кода в контейнеризованных архитектурах.
- Масштабирование микросервисов. Разбор подходов.
- Проблемы решения проблем развертывания без контейнеров. Детали функционирования CI серверов и как они ограничивают процессы работы поверх виртуальных машин.
- Подход к управлению конфигурацией. Облачная конфигурация против начальной инициализации.

Сборка, тестирование и поставка

- Чем отличаются CI и CD. Почему их не всегда стоит объединять и когда стоит разделять.
- Тестирование в микросервисных архитектурах. Почему не работает end to end тестирование.
- Стратегия непрерывного развертывания. Положительные эффекты и экономическая целесообразность.

Организационный дизайн

- Стратегии масштабирования Agile. SAFe, LeSS и Spotify.
- Почему архитектура важна на масштабе и почему об этом нельзя не думать заранее.
- Влияние архитектуры на дизайн команды. Feature team и компонентные команды.
- Подход Inner source и совместное владение кодом.

Контрольные мероприятия

По итогам курса выставляется итоговая 10 бальная оценка, формируемая на основе:

- 4 промежуточных теста в конце каждой 4-й лекции. До 10 вопросов с вариантами ответов. 0-1 балла за каждый тест.
- Устный экзамен. 2 вопроса. 0-4 балла, до 2 баллов за вопрос.
- Активное участие в дискуссиях на лекциях. 0-2 балла:
  - Оценивается наличие вопросов к преподавателю по пройденному за предыдущий академический час материалу и/или наличие дополнительных вопросов, выходящих за пределы рассказываемого материала, но связанных с темой лекции.

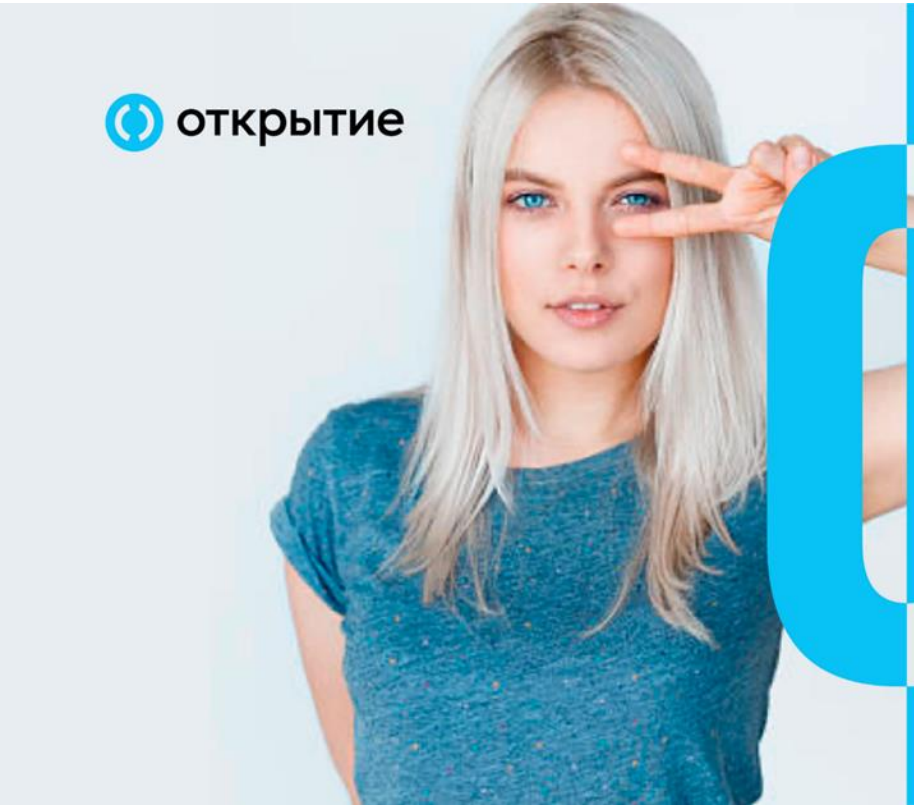
Группа в Telegram

Вопросы можно задавать в группе курса в Telegram:

«ИТ-архитектура банка. ВШЭ осень 2021»

Ссылка на группу:

https://t.me/joinchat/vSZp9Iafi7MxN2E6



0,

Понятие SOA

Понятие SOA

Се́рвис-ориенти́рованная архитекту́ра (СОА, англ. **service-oriented architecture- SOA**) — модульный подход к разработке программного обеспечения, базирующийся на обеспечении удаленного по стандартизированным протоколам использования распределённых, слабо связанных легко заменяемых компонентов (сервисов) со стандартизированными интерфейсами.

Сервис-ориентированная архитектура — Википедия

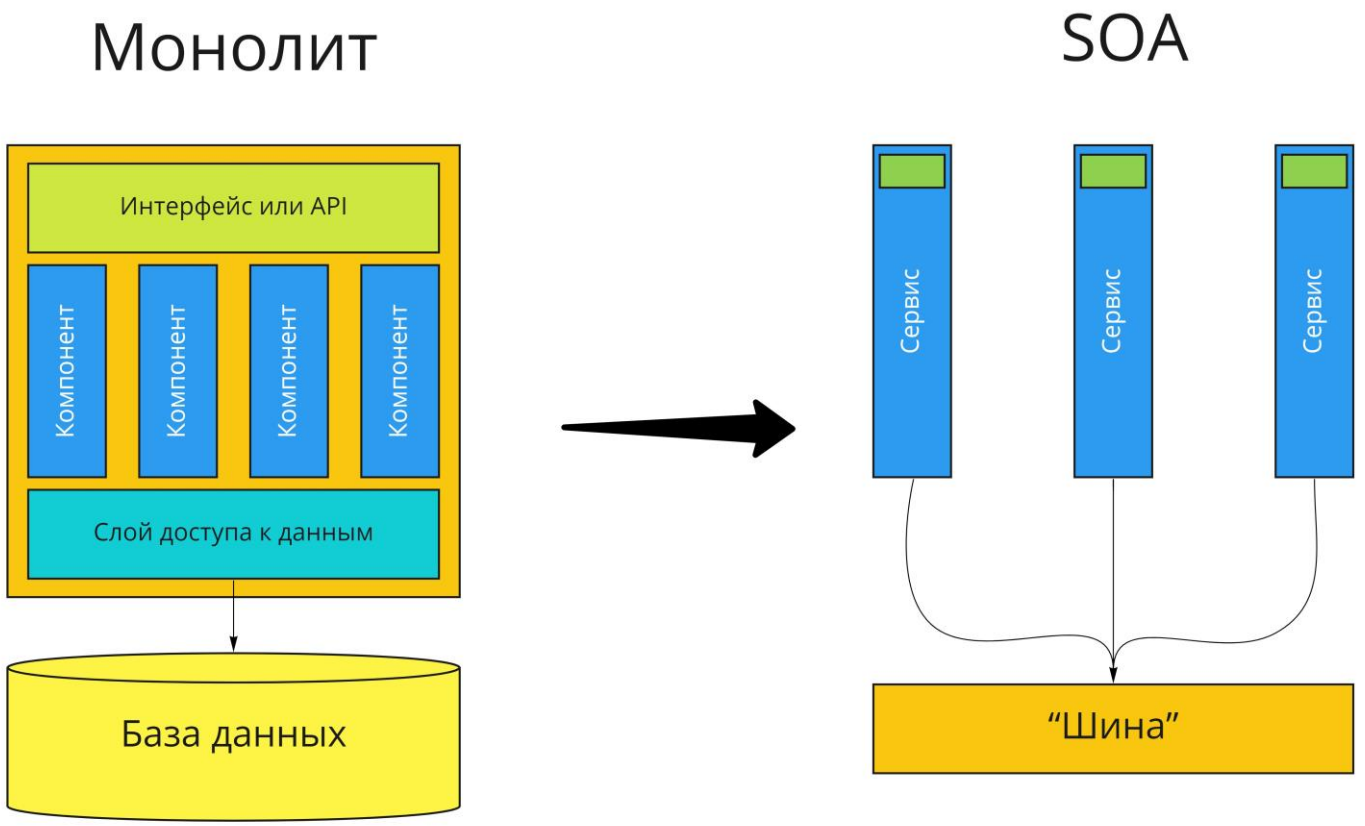
W ru.wikipedia.org › Сервис-ориентированная архитектура

Домашнее чтение:

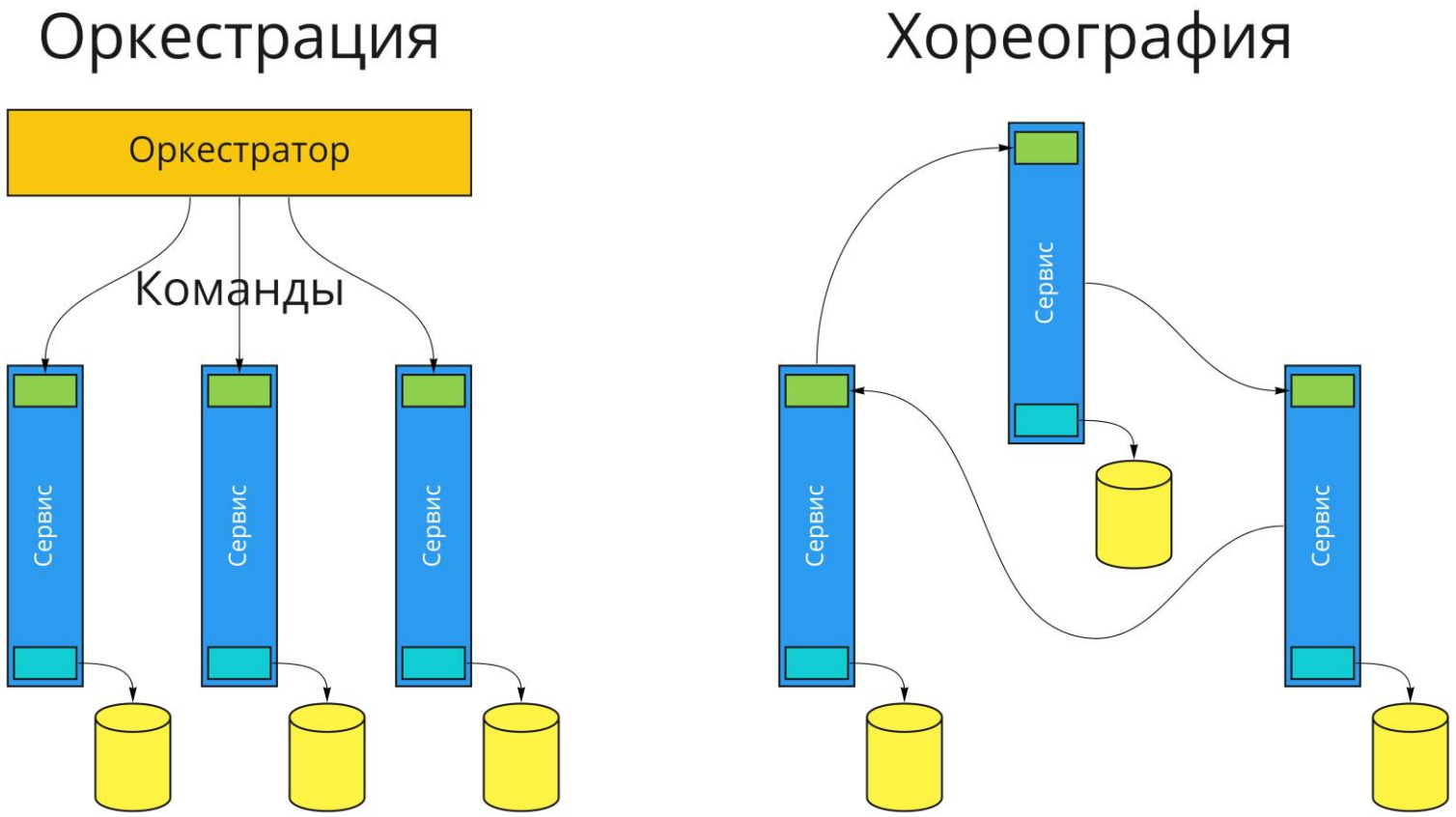
Сервис-ориентированная архитектура (SOA)

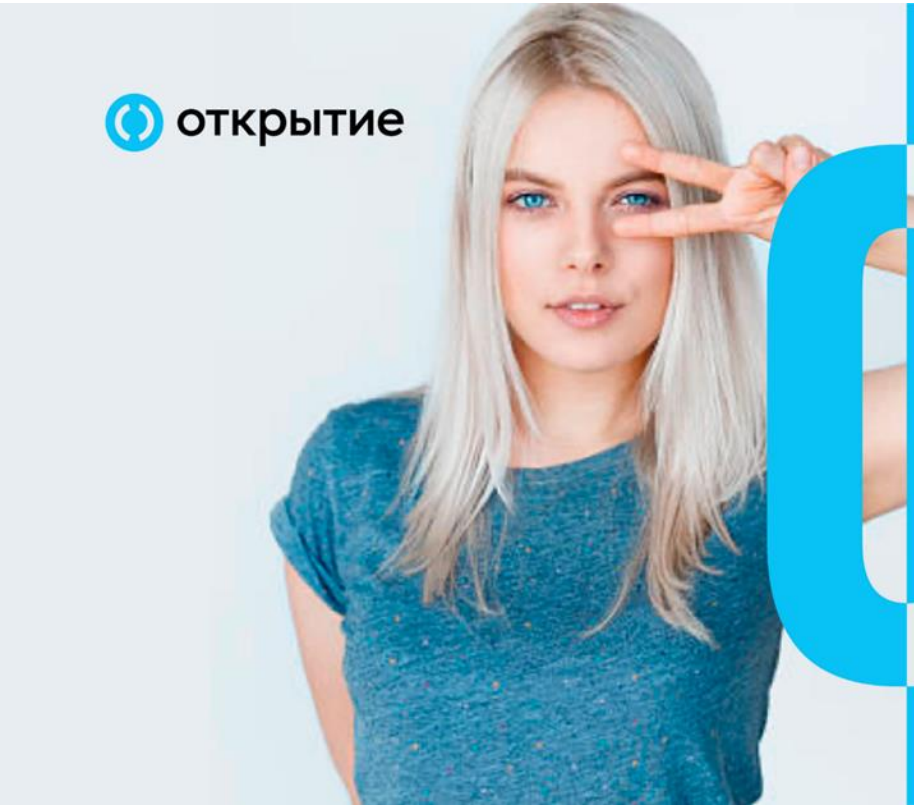
https://habr.com/ru/company/mailru/blog/342526/

Понятие SOA


Монолит
Интерфейс или API
Компонент
Компонент
Компонент
Компонент
Слой доступа к данным
База данных
SOA
Сервис
Сервис
Сервис
“Шина”

Хореография и оркестрация

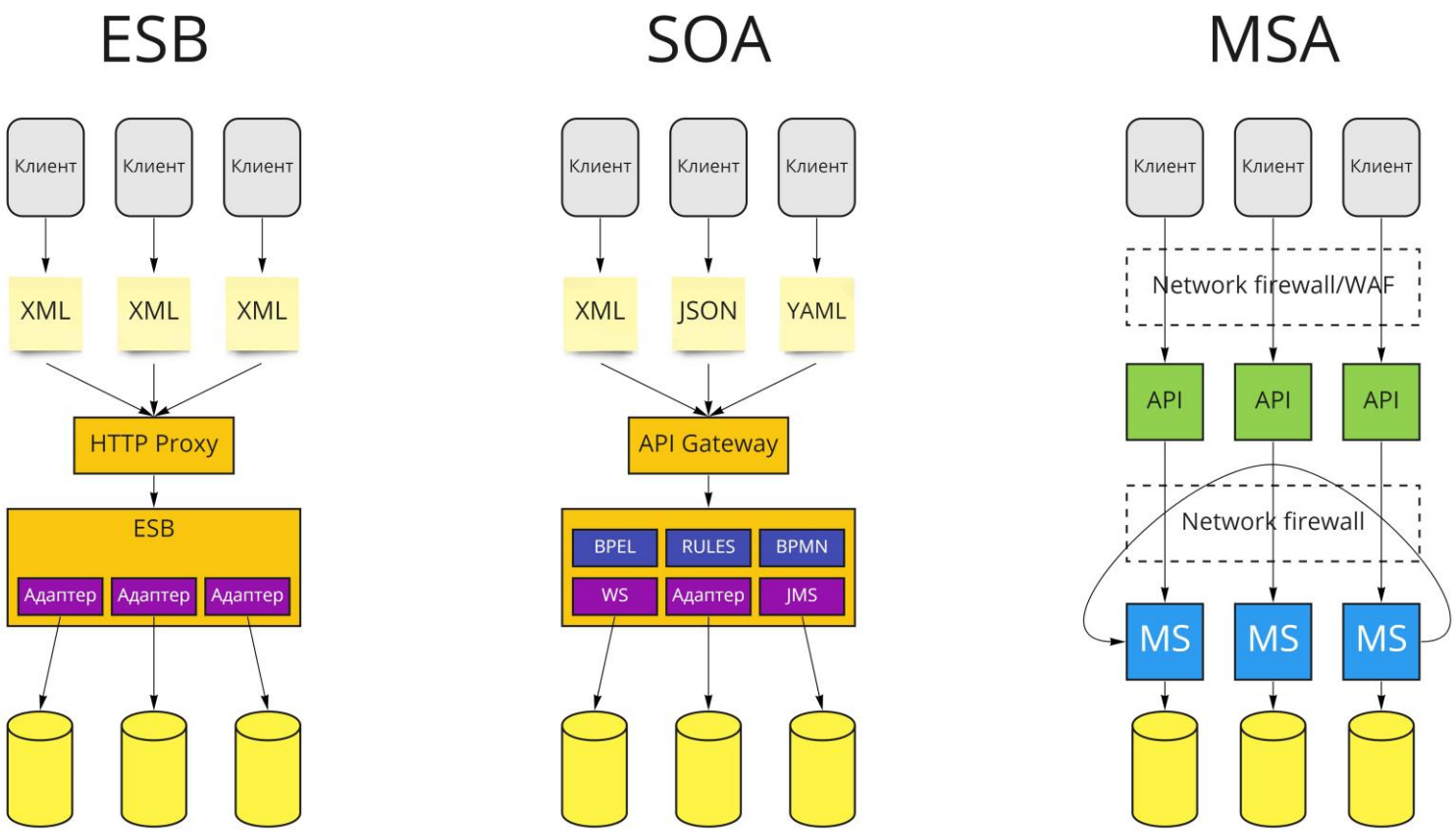




0,

Жизнь до микросервисов

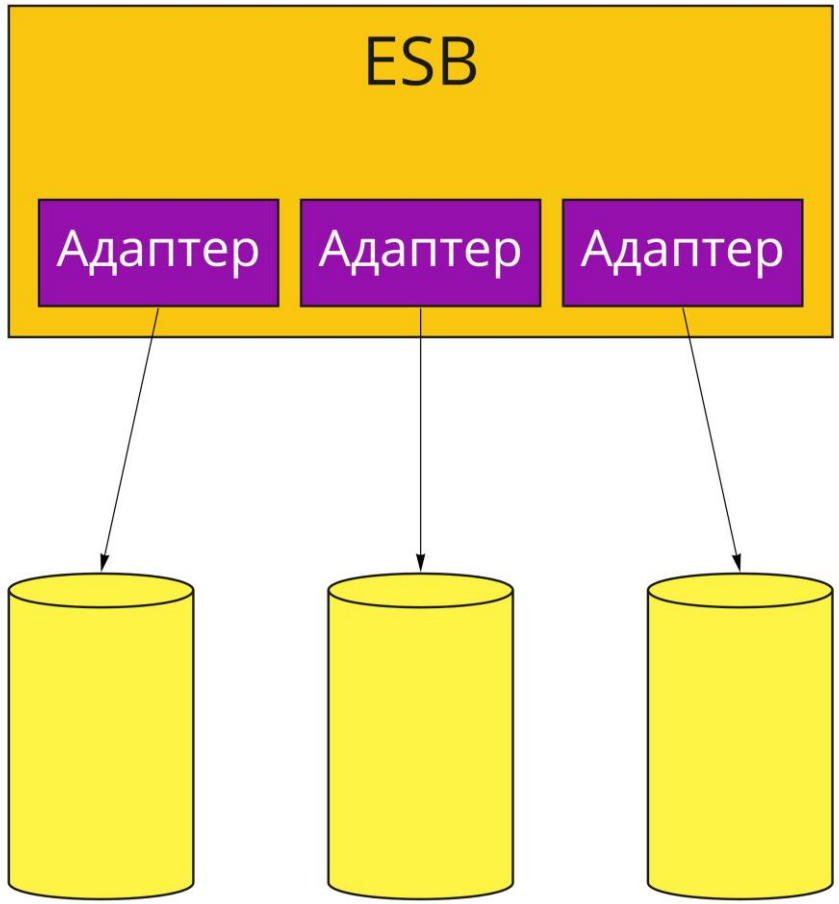
Жизнь до микросервисов



Немного исторических buzzword’ов

- Web Service – термин, порожденный сумрачным гением в эпоху до падения «доткомов» на волне хайпа для обозначения всего взаимодействующего через интернет-технологии
- SOAP – «мыло», основанный на XML стандарт взаимодействия веб сервисов. Связан с WSDL
- WSDL – он же «висдел», стандарт описания схем взаимодействия вебсервисов. Вымер во всех адекватных современных фреймворках
- CORBA, Distributed COM, Java EE, WCF, JMS и т.п. – сумрачные технологии, порожденые в эпоху до микросервисов для коммуникации распределенных приложений

ESB – достоинства и проблемы



Достоинства

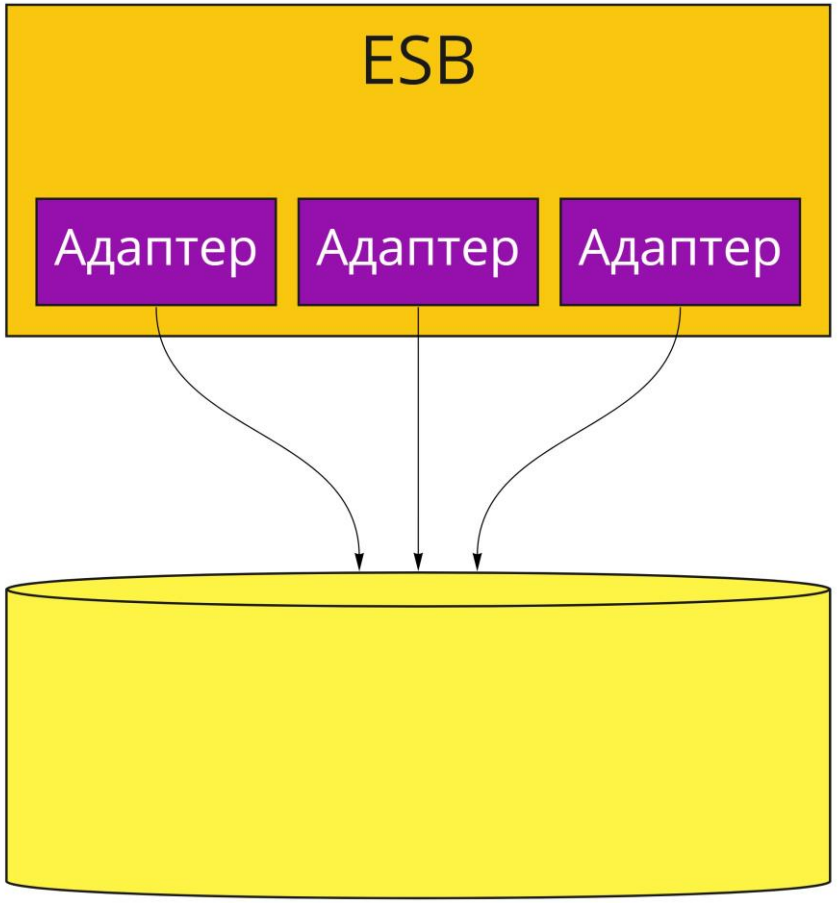
- Позволяет интегрироваться со смежными системами с минимумом издержек
- Весь код в одном месте и пишется стандартно
- Концепция идеально понятна для эксплуатации

Проблемы

- Весь код в одном месте – единая точка отказа
- Работает в рамках проприетарных технологий – контроль версий и DevOps обычно недоступны и воспринимаются развитием ESB как ругательные
- Для взаимодействия обычно нужны проприетарные библиотеки, которых может не быть

В целом – не сопровождаемо на сколь-либо большом масштабе

ESB – в реальном мире

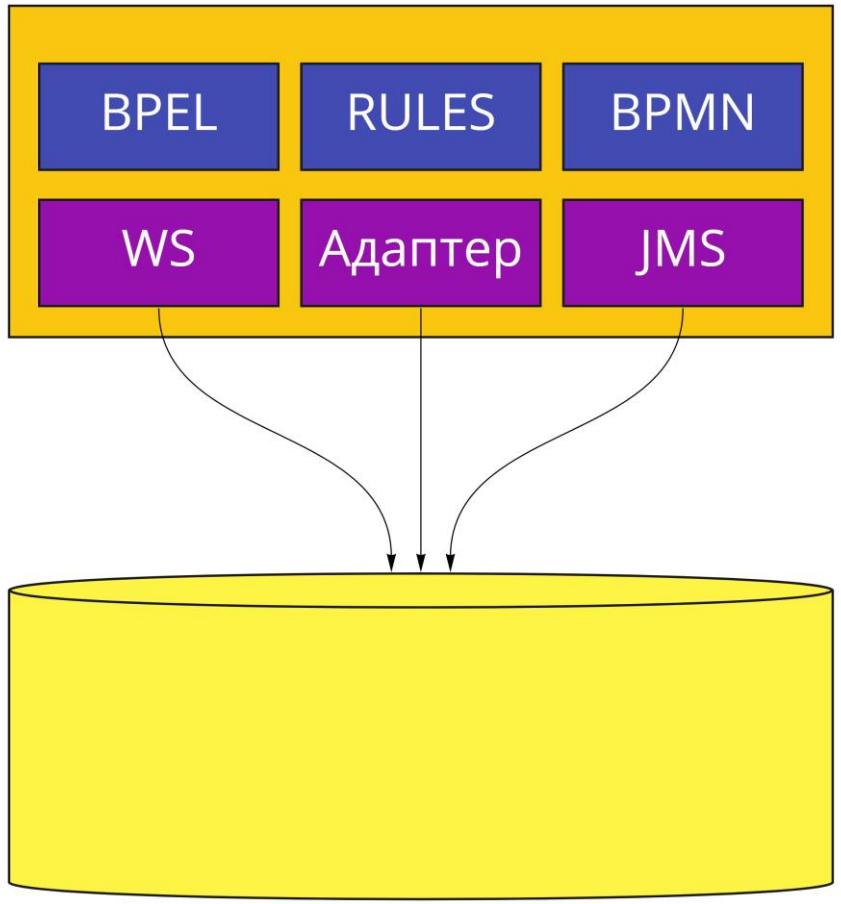


Одна большая проблема

- Никто никогда на практике не делит базы данных пока можно масштабировать СУБД вертикально
- А при современном железе это можно делать очень долго
- И дорого…

В итоге – разобрать это на части уже не возможно, только переписать

Развитие SOA само по себе ничего не решило

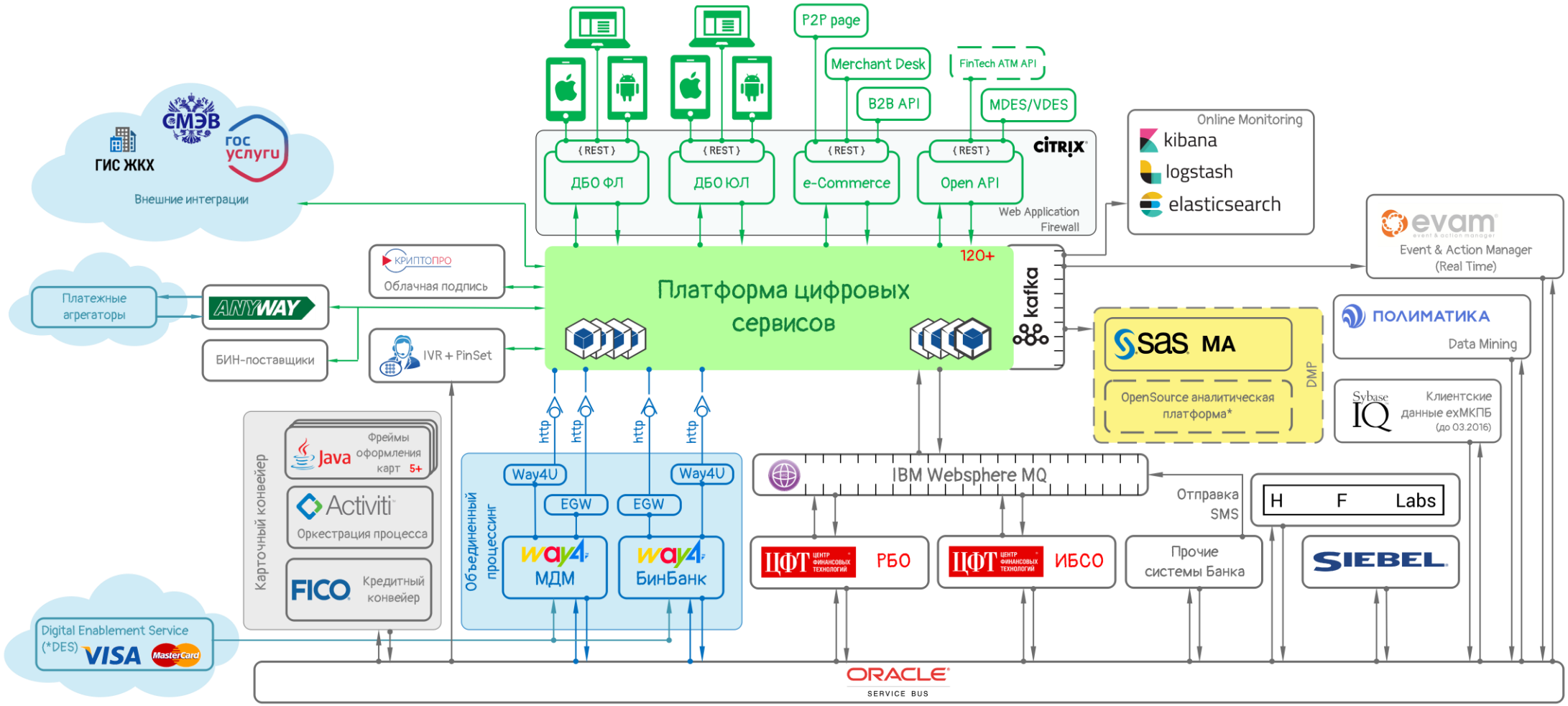


И SOA эту проблему тоже никак не решает на практике

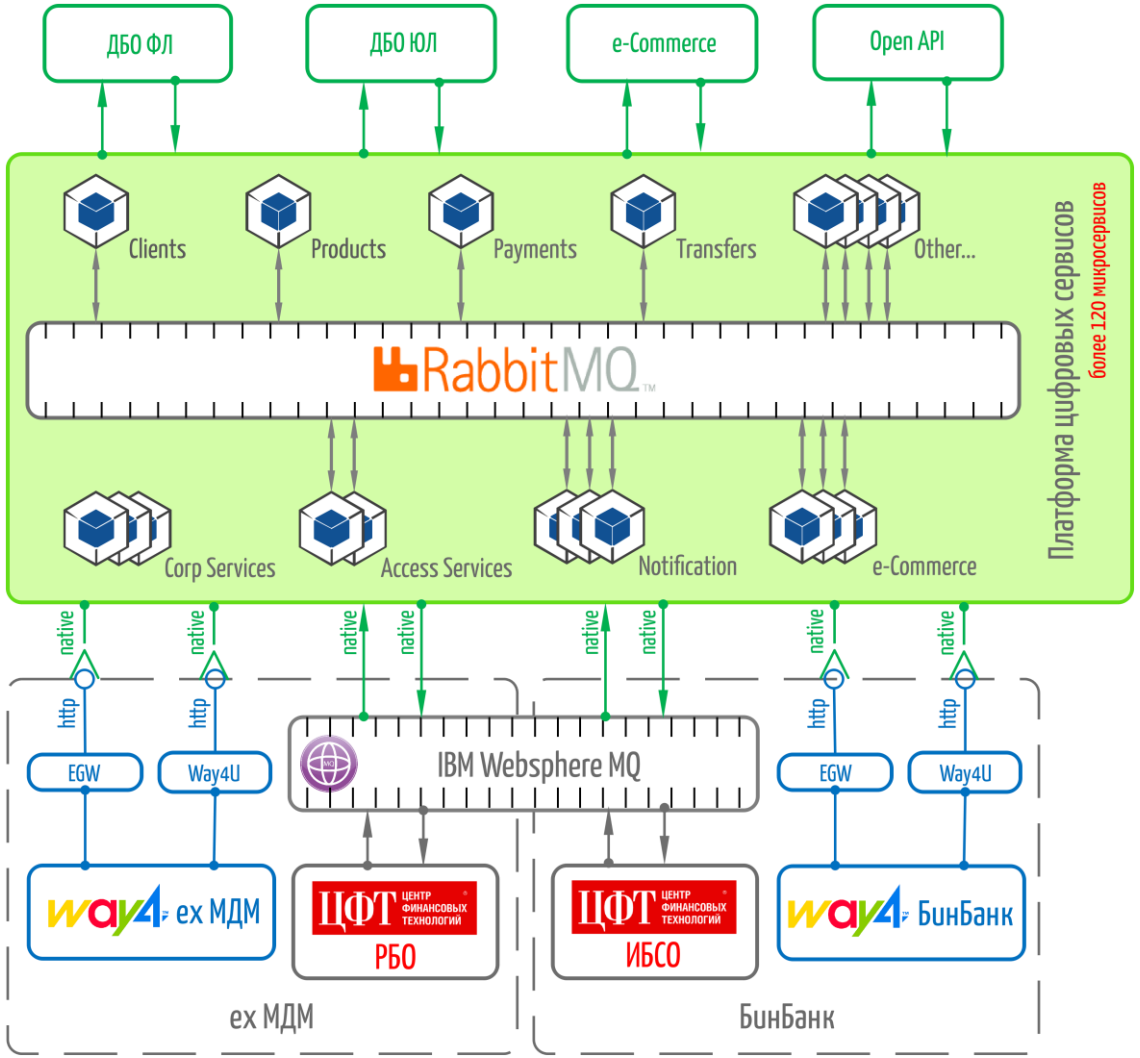
Вообще граница между ESB и SOA на практике отсутствует. Обычно это все **распределенный монолит** из-за обычно используемого подхода к автоматизации через процессы и сценарии

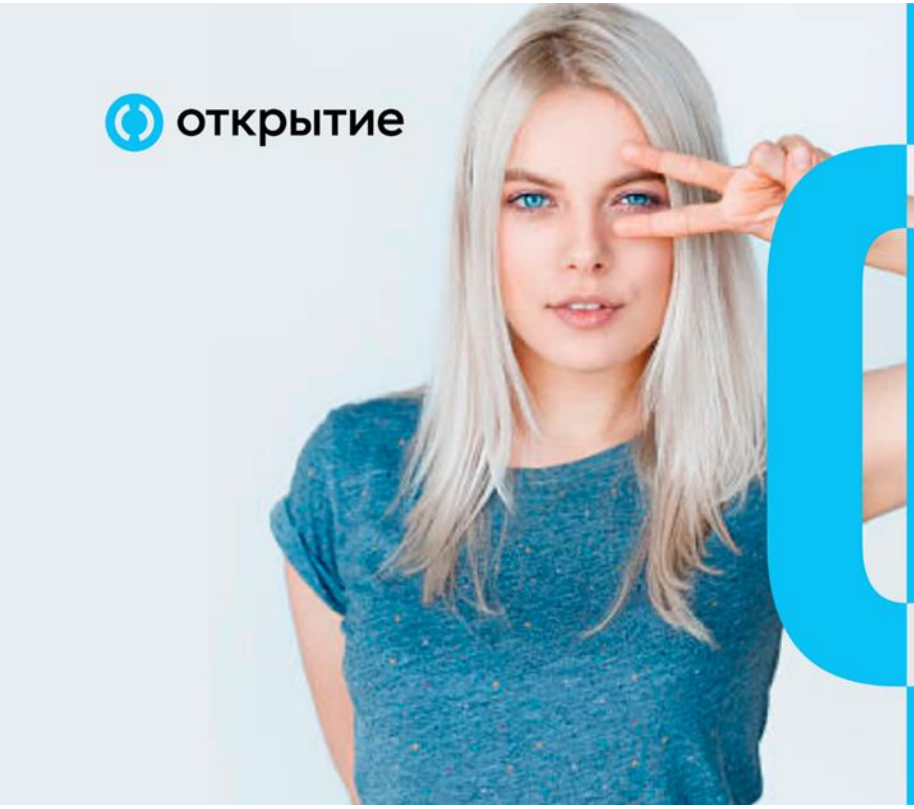
Об этом на следующей лекции

Пример реального ландшафта Банка



Микросервсная часть крупным планом





0,

Микросервисы

Понятие микросервиса

Microservices

a definition of this new architectural term

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

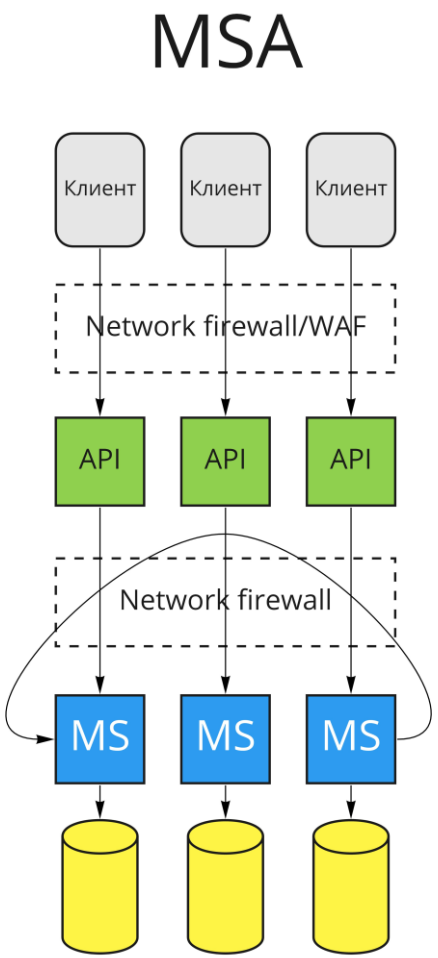
25 March 2014

Домашнее чтение:

Microservices (by Martin Fowler)

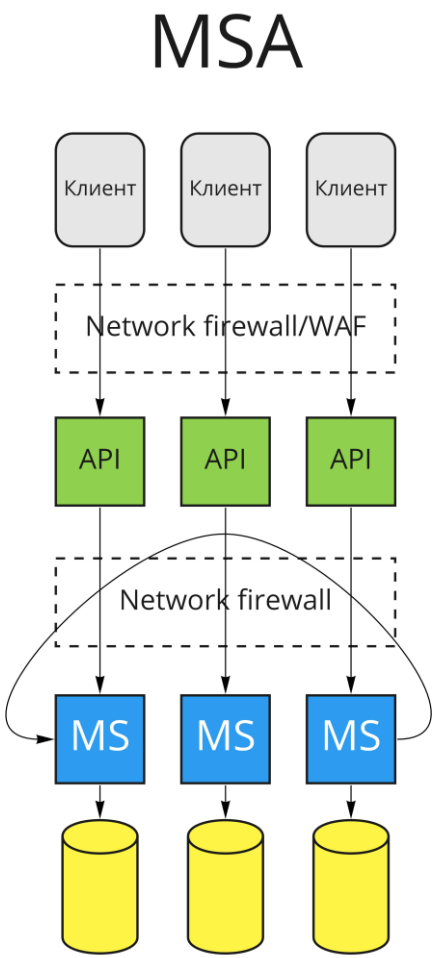
https://martinfowler.com/articles/microservices.html

9 признаков MSA по М. Фаулеру 1/3



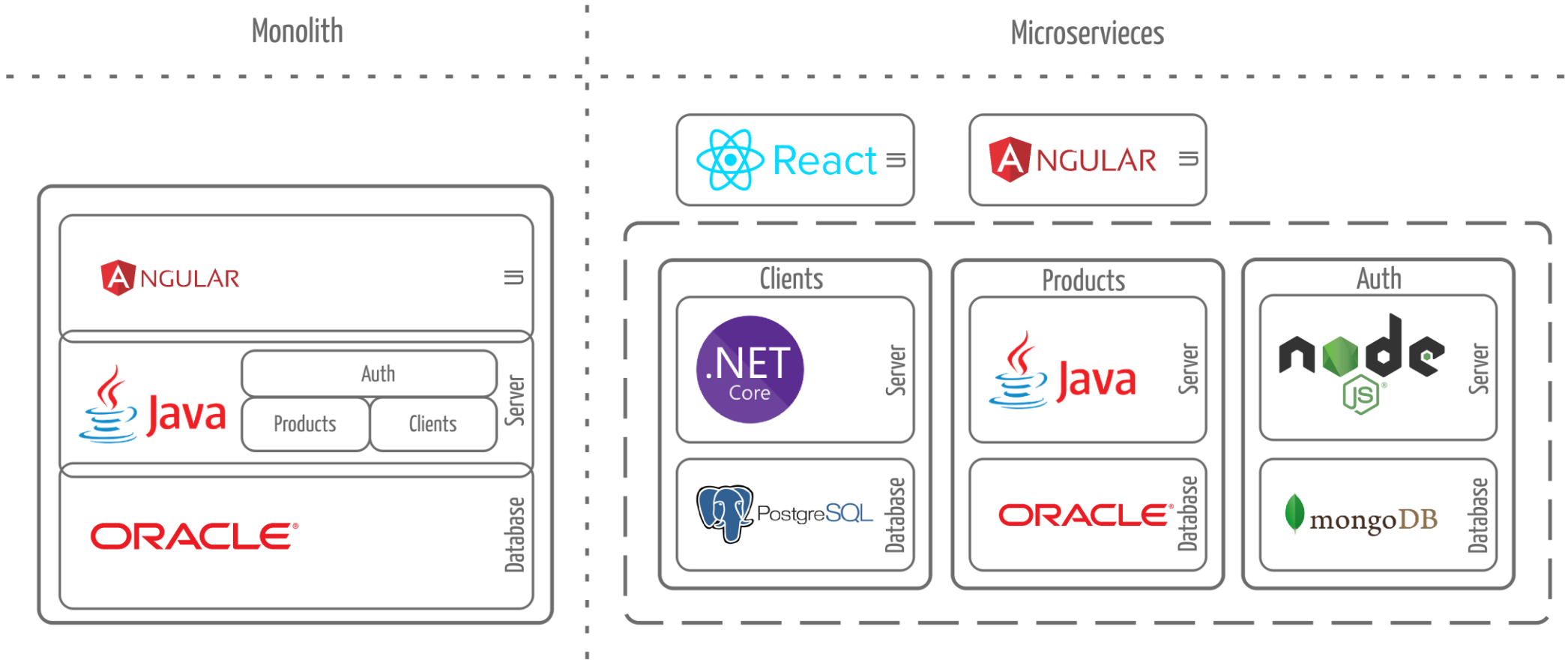
1. **Componentization via Services** – организация компонентов в виде сервисов. По сути, MSA это SOA
2. **Organized around Business Capabilities** – организация вокруг бизнес возможностей. Будем обсуждать в последующих лекциях. Важно то, что MSA это не техническая архитектура, а организационно-техническая, с организацией впереди
3. **Products not Projects** – развитие должно быть организовано вокруг продуктов, а не проектов. Самое важное – у продукта нет предопределенного дня смерти

9 признаков MSA по М. Фаулеру 2/3

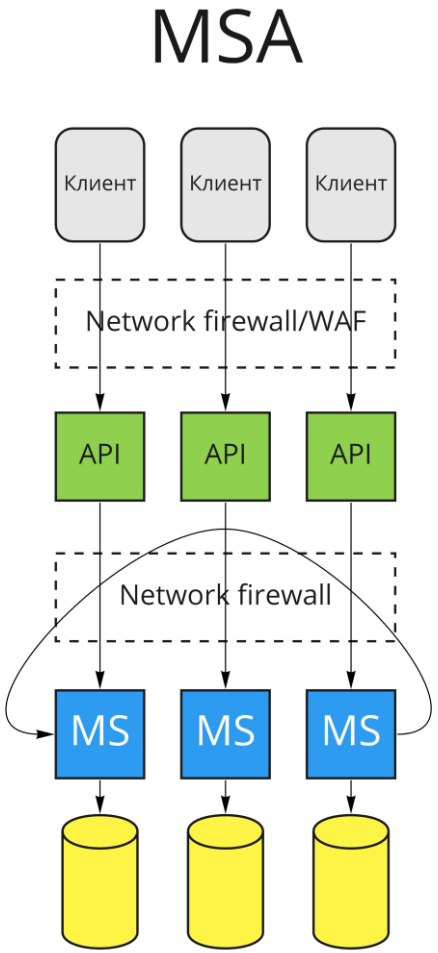


4. **Smart endpoints and dumb pipes** – умные конечные точки и глупые каналы. Фундаментальное различие с предыдущими подходами
5. **Decentralized Governance** – децентрализованное регулирование. Разные сервисы могут быть разработаны на разных технологиях. В реальной жизни часто не удобно
6. **Decentralized Data Management** – децентрализованное управление данными. На практике – у каждого сервиса (домена) своя БД

Пример децентрализации



9 признаков MSA по М. Фаулеру 3/3



7. **Infrastructure Automation** – автоматизация инфраструктуры. На практике – чтобы работало, DevOps практики нужны не на бумаге
8. **Design for failure** – спроектировано для отказа. Система не должна переставать при отказе одного из сервисов. По крайней мере не критичного для конкретного бизнес процесса
9. **Evolutionary Design** – эволюционный дизайн. Нельзя спроектировать сервис изначально верно. Из-за изменений требований разбиение сервисов неминуемо будет меняться

На этом всё. Спасибо!

Вопросы?