Архитектура ИС

Регина Гатауллина
Product manager Bookmate

Курс Системный аналитик
Модуль Жизненный цикл ПО

Регина Гатауллина

Product manager Bookmate

Аккаунты в соц.сетях

- https://www.facebook.com/rigataullina/
- instagram.com/regza.stories
- t.me/ReginaGataullina
- linkedin.com/regina-gataullina-956b5683/

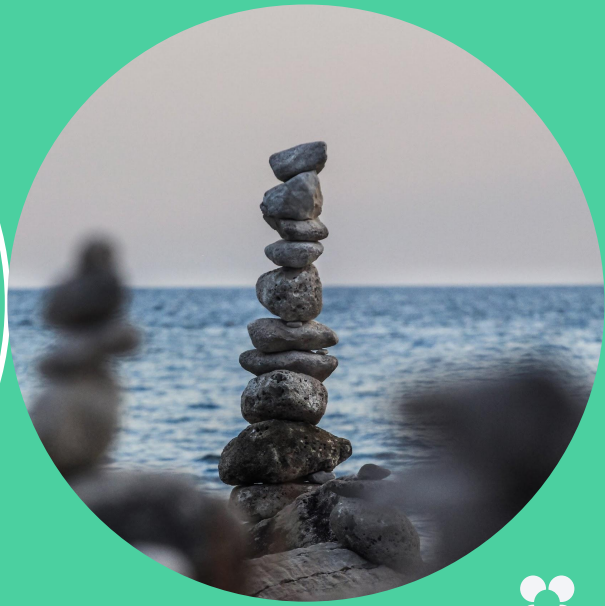
План занятия

1. Архитектура ИС. Что это? Классификация. Эволюция
2. “Монолит” vs SOA vs Микросервисная
3. Интеграция. Брокеры сообщений
4. Риски при проектировании Архитектуры

- слайды с важной информацией, которую нужно запомнить, будут помечены значком “photo”

Архитектура ИС
Что это?

1



Определение

1

концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы

2

базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и окружением, а также принципы, определяющие проектирование и развитие системы (ГОСТ Р 57100-2016/ISO/IEC/IEEE 42010:2011)

3

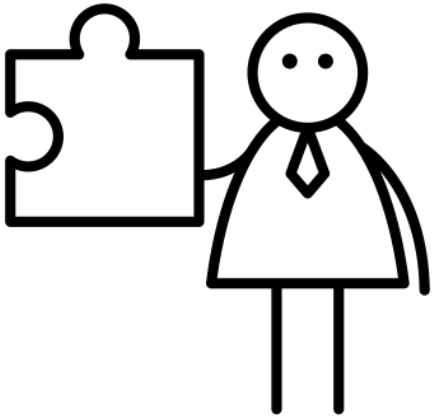
набор значимых решений по поводу организации системы программного обеспечения, набор структурных элементов и их интерфейсов, при помощи которых компонуется система вместе с их поведением, определяемым во взаимодействии между этими элементами, компоновка элементов в постепенно укрупняющиеся подсистемы, а также стиль архитектуры, который направляет эту организацию (элементы и их интерфейсы, взаимодействие и компоновка)

Кейс

Где провести черту, разграничивающую
зоны ответственности

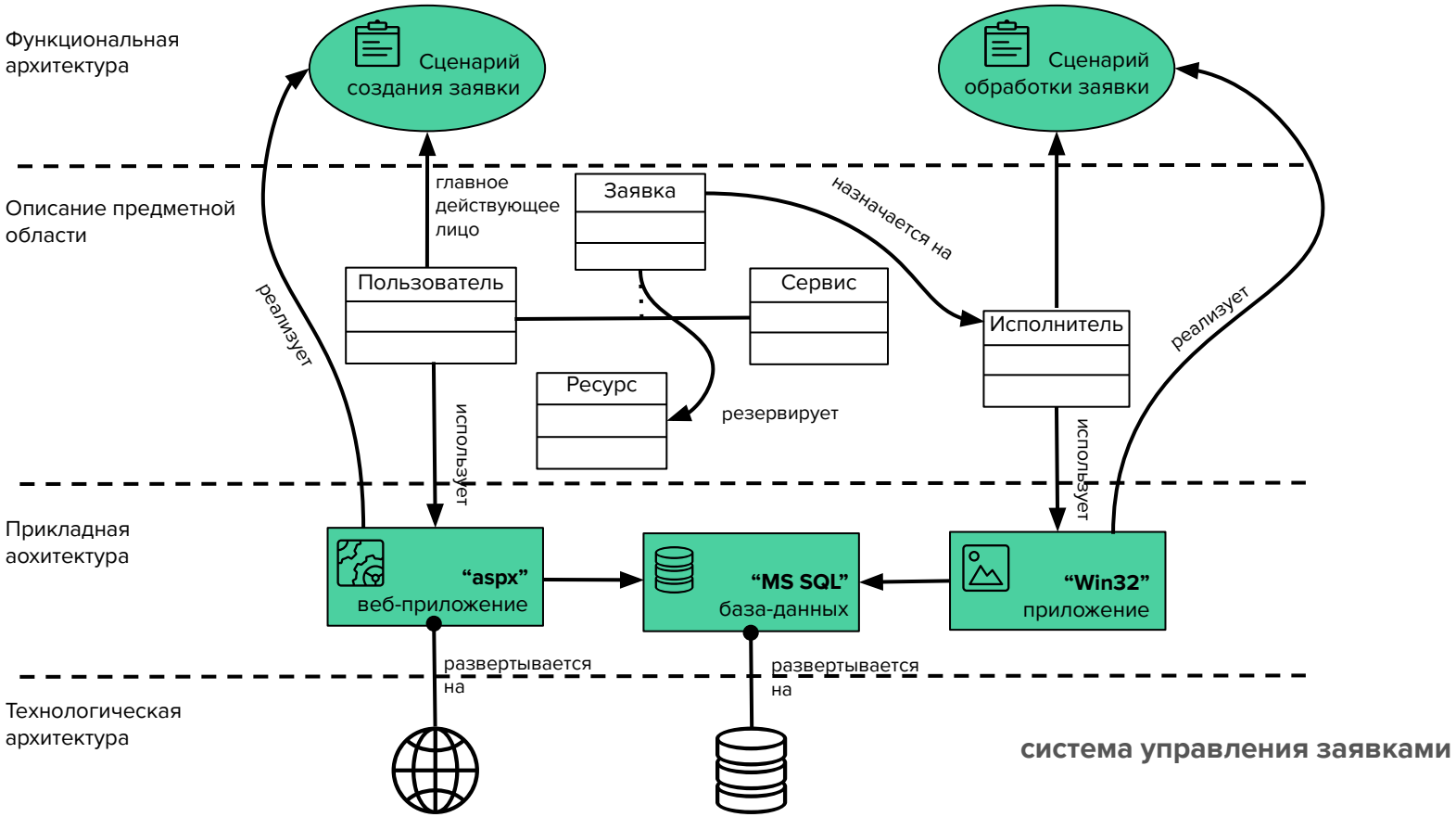
?

Системный аналитик



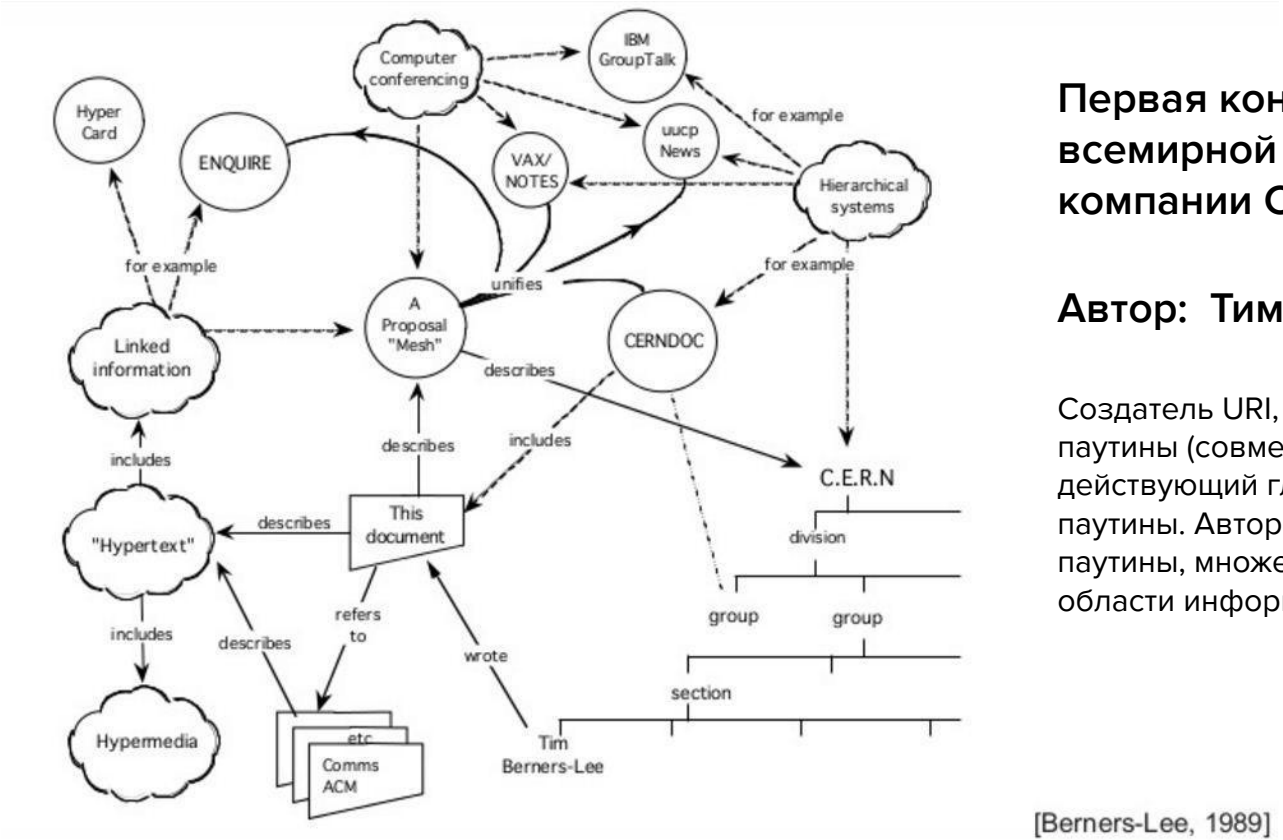
Архитектор

Все понимают Архитектуру по-разному



система управления заявками

“Рисунки” на доске/флипчарте



[Berners-Lee, 1989]

Первая концепция создания всемирной “паутины” ещё внутри компании CERN созданная в 1989г

Автор: Тим Бёрнерс-Ли

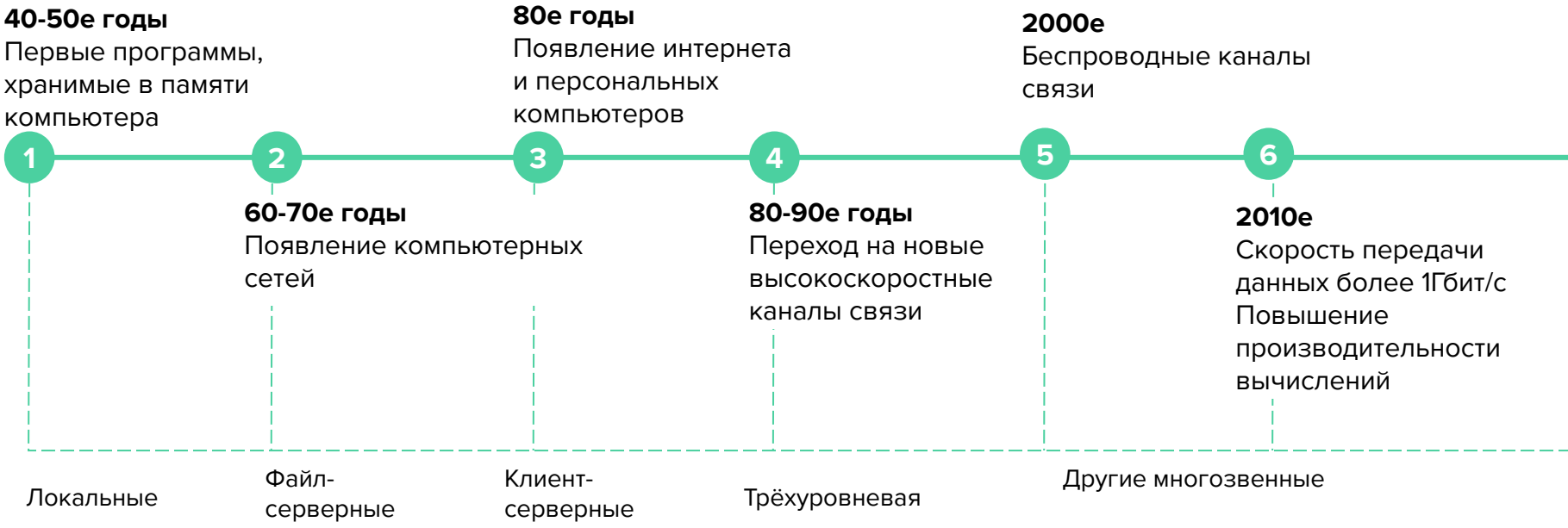
Создатель URI, URL, HTTP, HTML и Всемирной паутины (совместно с Робертом Кайо) и действующий глава Консорциума Всемирной паутины. Автор концепции семантической паутины, множества других разработок в области информационных технологий.

Архитектура архитектуры

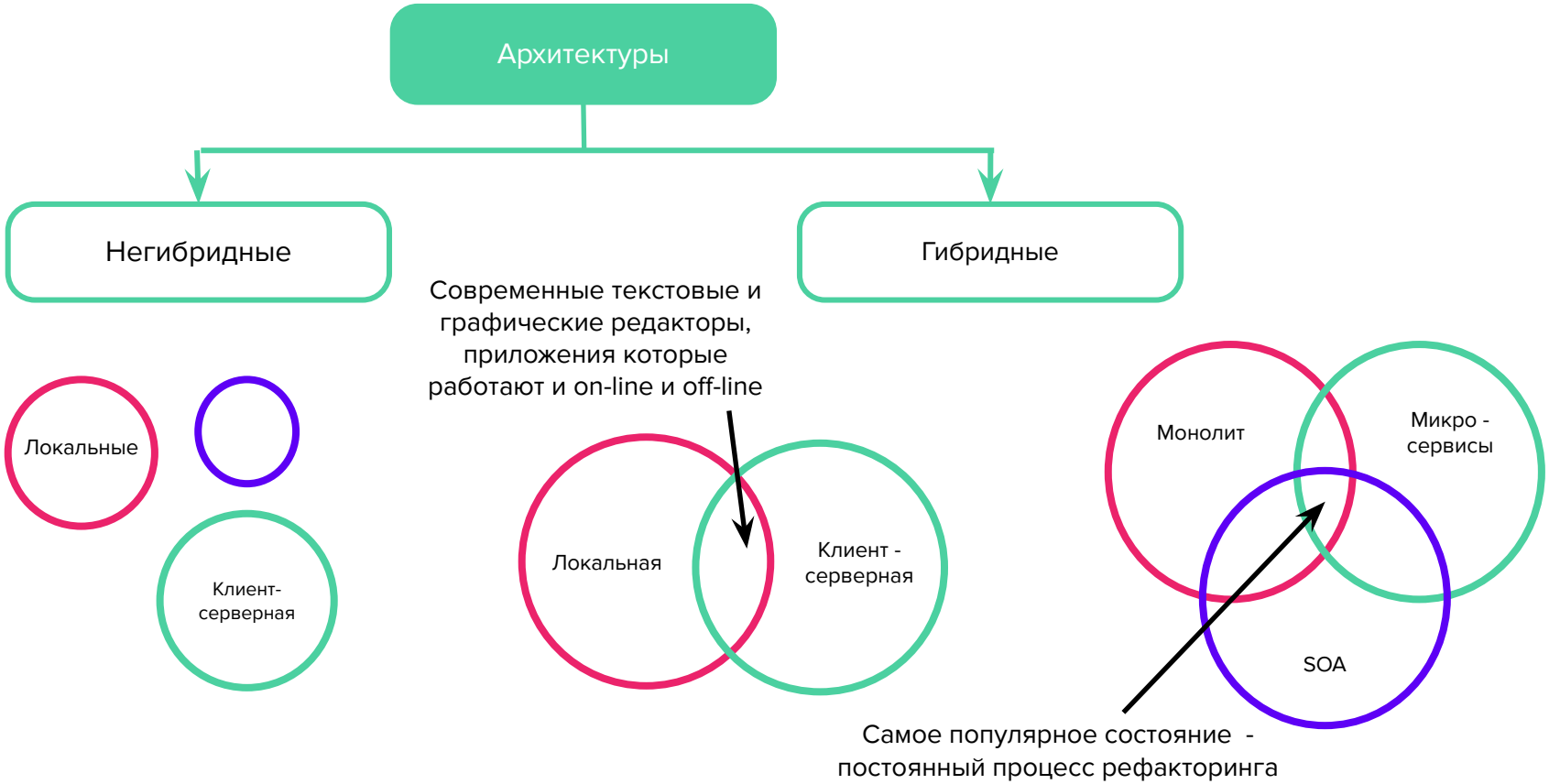
1. что будет делать система? → Базовые параметры и характеристики архитектуры. Возможные риски, ограничения, стоимость владения, экономическая обоснованность.
2. из каких компонентов (частей, модулей) она будет состоять? → Логическая и физическая структура
3. где именно компоненты будут располагаться? → Необходимые элементы ИТ-инфраструктуры для реализации встраиваемой архитектуры ИС – платформа (среда), аппаратный комплекс, СУБД, инструментарий прикладное ПО.
4. каким образом компоненты будут взаимодействовать? → Взаимодействие системных компонентов (подсистемы и модули, синхронность и асинхронность их взаимодействия, каналы коммуникации и их характеристики, протоколы и интерфейсы, тип программного обеспечения промежуточного слоя, форматы файлов, которыми система будет оперировать, и другие особенности)

Классификация

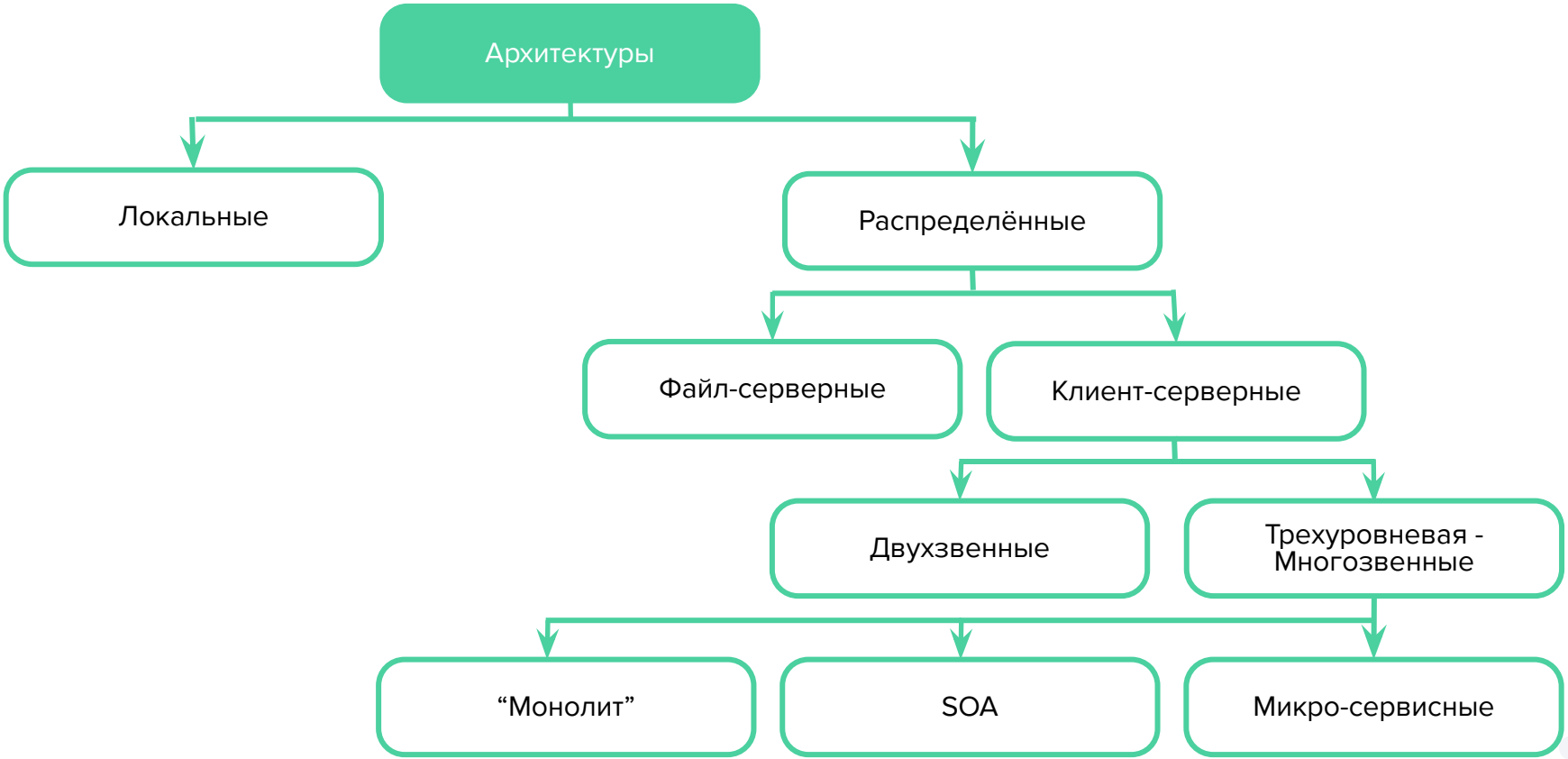
Историческая справка



Типы архитектур



Типы архитектур



Локальные ИС

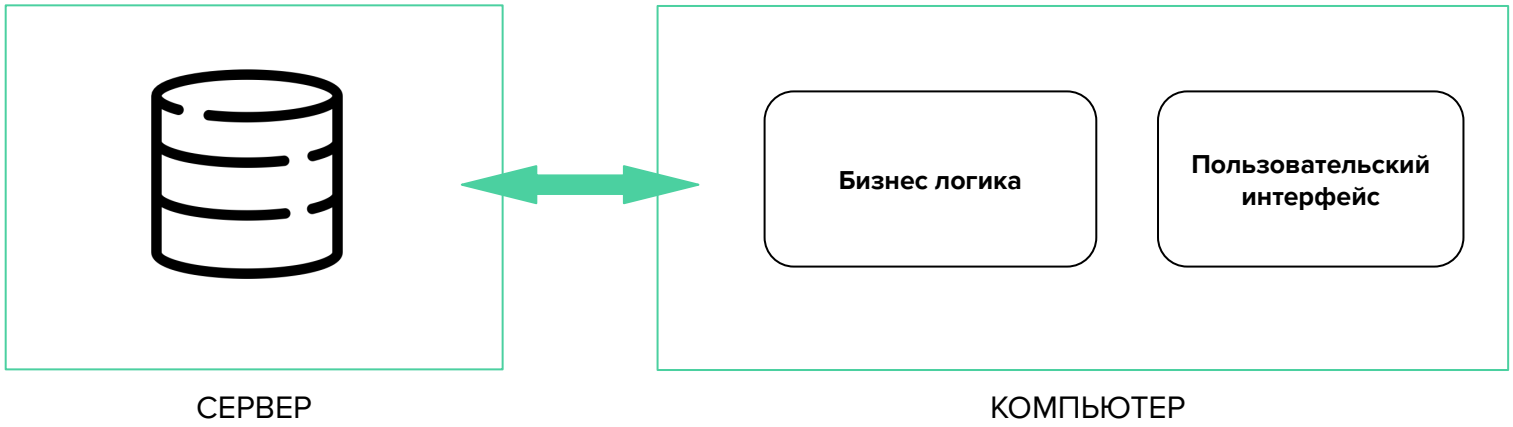
КОМПЬЮТЕР

Хранение информации

- С использованием полной версии любой СУБД
- С использованием встроенной СУБД
- Без использования СУБД (файлы)

Работать в ИС может только один пользователь. Другие пользователи не имеют возможности получить доступ к данным даже для чтения.

Файл-серверные ИС



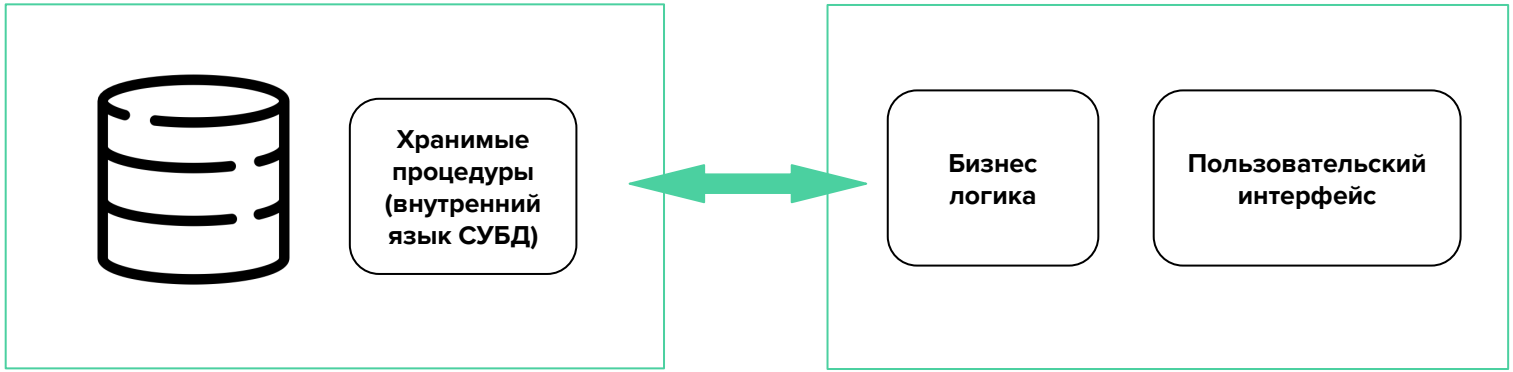
- Сервер выполняет роль только файлового сервера (база данных или файлы).
- Компьютеры пользователей соединены с сервером сетью, поэтому доступ к данным, могут получить несколько пользователей одновременно.
- Однако, кроме функции хранения данных и обеспечения доступа к ним, сервер никаких функций не выполняет.
- Приложения, обрабатывающие данные, находятся на пользовательских компьютерах.

Файл-серверные ИС 

1. Совместный доступ к данным;
2. Единое хранение данных

1. Высокая загруженность сети;
2. Низкая скорость работы;
3. Сложность поддержания непротиворечивости данных, из-за их несогласованной обработки разными пользователями.

Клиент-серверная ИС



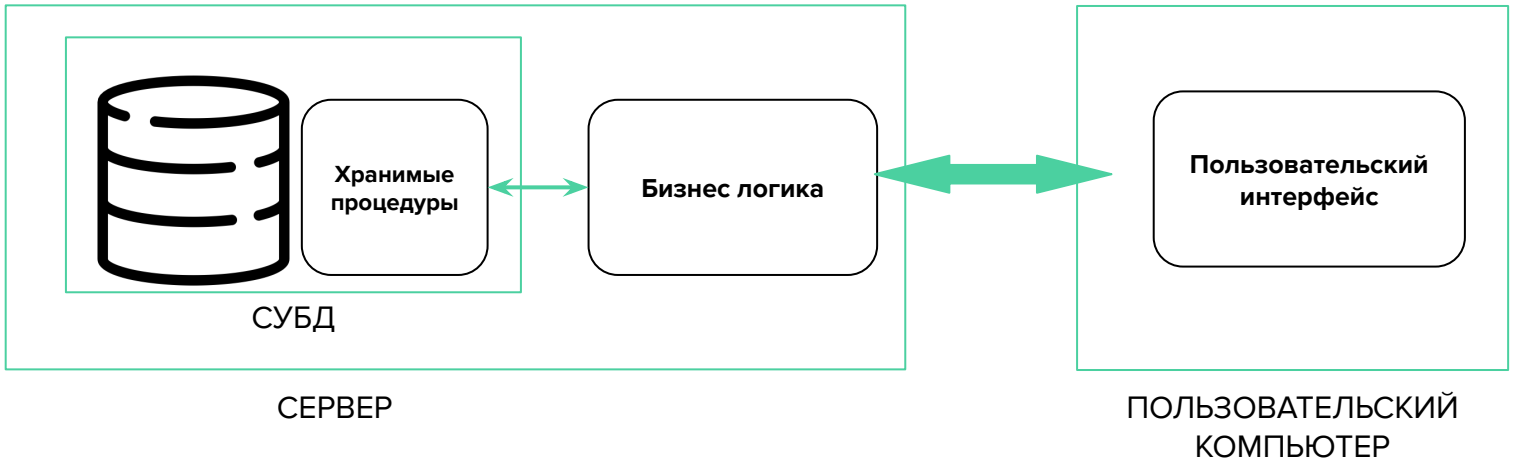
В состав СУБД включили процедурный язык программирования. Начали создавать процедуры для обработки данных, которые можно вызывать повторно. Такие процедуры называются хранимыми процедурами.
Языки хранимых процедур не приспособлены для полноценной реализации бизнес-логики.
Поэтому бизнес-логика в клиент-серверных ИС по-прежнему реализуется на клиентских компьютерах.

Клиент-серверная ИС

1. Совместный доступ к данным;
2. Единое хранение данных
3. Поддержание непротиворечивости данных

1. Слабая защита данных от взломов;
2. Высокие требования к производительности ПК;
3. Обновление клиента при изменении бизнес-логики

Трёхуровневая ИС



Использование серверных языков программирования позволило создать сервер-приложений, на который можно было перенести бизнес-логику.
Использование сервера приложений позволяет максимально разгрузить клиентские компьютеры и сделать обработку данных еще более централизованной, что повышает скорость и надежность ИС.

Трёхуровневая ИС 

1. Совместный доступ к данным;
2. Единое хранение данных
3. Поддержание непротиворечивости данных
4. **Усиление** защиты данных от взломов;
5. **Снижение** требований к производительности ПК;
6. **Отсутствие постоянного обновления клиент**а при изменении бизнес-логики

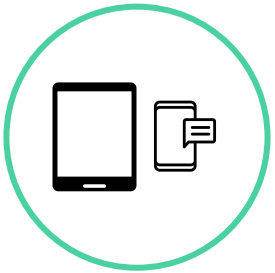
Типы клиентов

Какие вы знаете типы клиентов?

Типы клиентов

Приложение для Операционной системы ПК

Web-клиент



Приложение для мобильных устройств

Типы клиентов

Приложение для Операционной системы ПК

Web-клиент

Мобильный Web-клиент

Приложение для мобильных устройств

Специализированные устройства

Чат-боты

AR/VR

Smart TV

Smart watch

“Антракт”

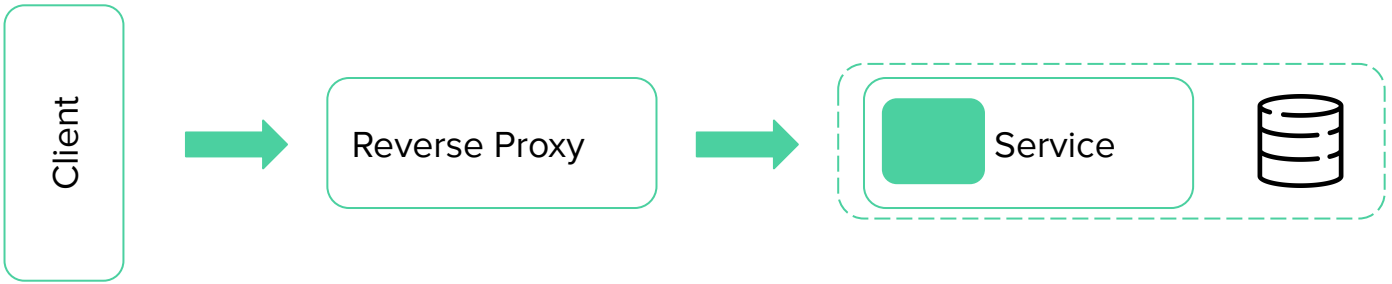
?

Есть вопросы или непонятные термины?

“Монолит”

2

Monolithic Backend



Прокси-сервер (proxy) — промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника между пользователем и целевым сервером, позволяющий клиентам как выполнять косвенные запросы к другим сетевым службам, так и получать ответы.

Обратный прокси (**reverse proxy**) непосредственно взаимодействует лишь с ассоциированными с ним серверами и возвращает ответ только от них

Один код обрабатывает все запросы, выполняет всю бизнес-логику

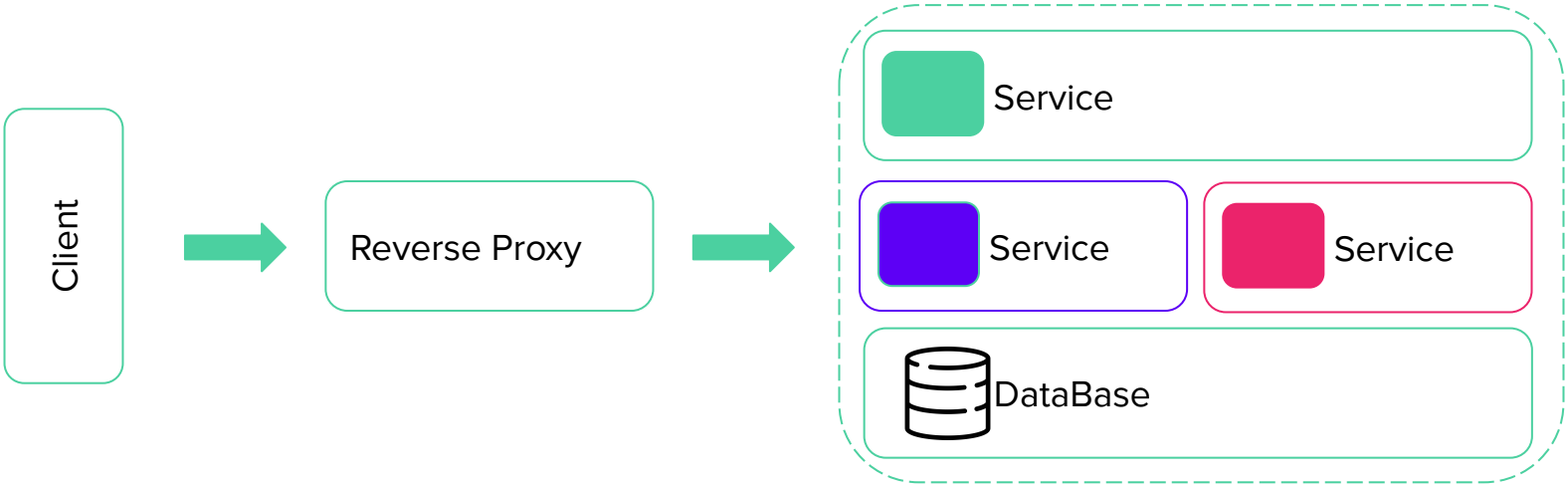
Monolithic Backend

- **высокая скорость разработки**

Идеален для создания MVP

- низкая отказоустойчивость;
- отсутствие горизонтального масштабирования;
- высокая трудоёмкость эксплуатации и поддержки;
- **применение одной технологии или языка;**
- сложность рефакторинга из-за хранения кода в одном месте и большое количество “старого кода”;
- **трудности работы в команде разработчиков;**
- чтобы использовать какой-то функционал повторно, придётся делать рефакторинг.

Monolithic Backend - 2

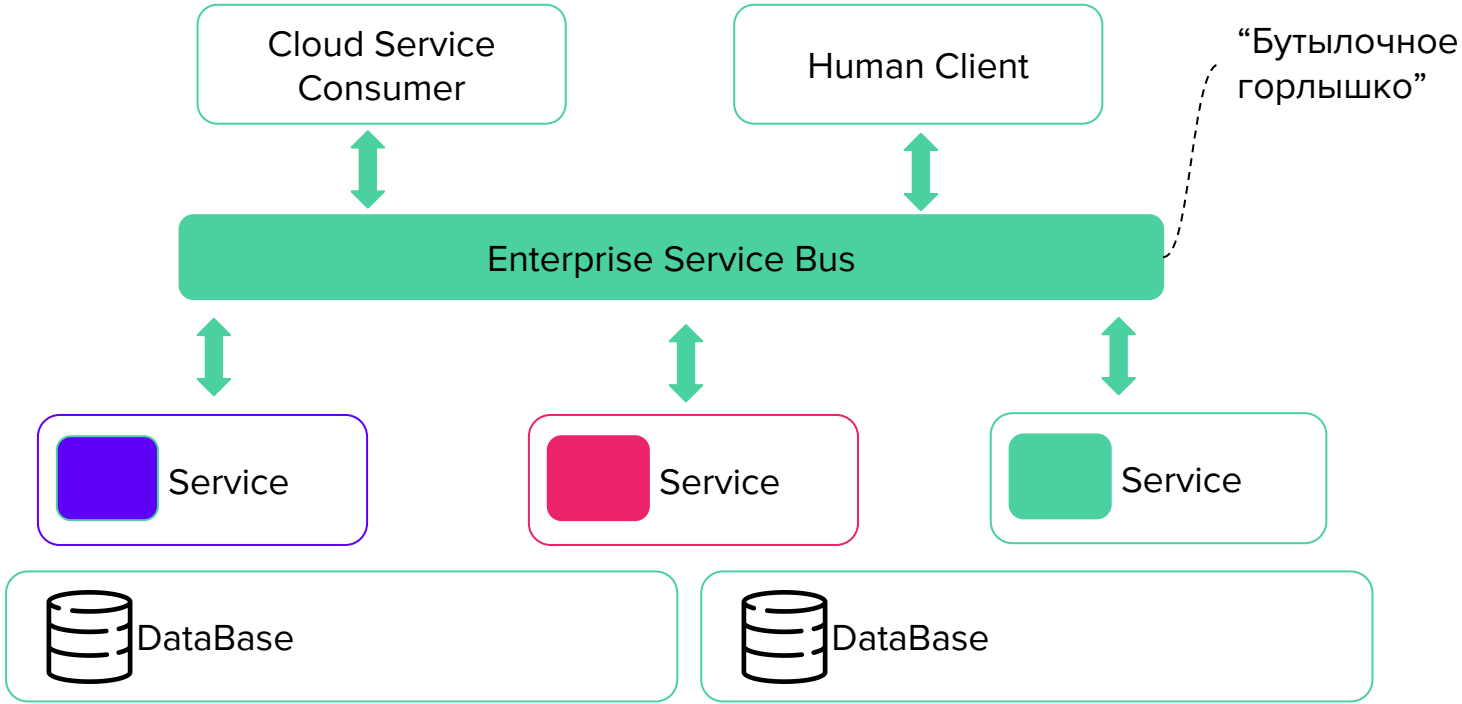


Второй по популярности вид архитектуры – пара монолитов, микс из монолита и сервисов или даже микросервисов. То есть вы сохраняете монолит, а доработки выполняете с использованием современных технологий.

Это частично решает проблемы отказоустойчивости, масштабируемости и одного стека технологий.

Сервисно-ориентированная Архитектура

SOA - Service Oriented Architecture



SOA - Service Oriented Architecture 

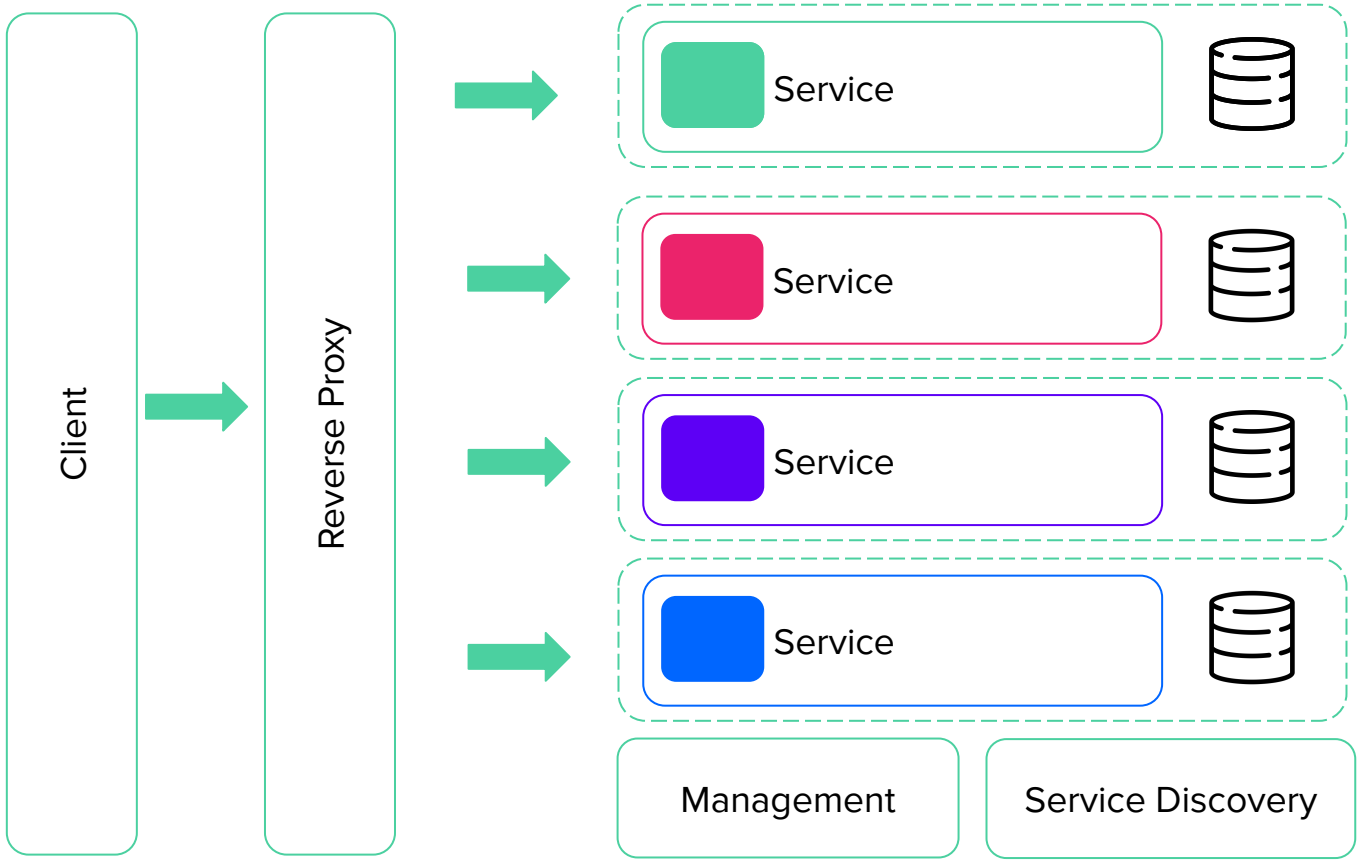
- **Низкая скорость разработки**
- **Сложность поддержки и разработки сервисной шины**

> Сервисная шина предприятия (англ. enterprise service bus, ESB) — связующее программное обеспечение, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами.
> Похожа на прокси, но поддерживает транзакционную модель, содержит элементы бизнес-логики, преобразовывает сервисы

- **отказоустойчивость;**
- **горизонтальное масштабирование;**
- **применение разных технологий или языков**
- высокая трудоёмкость эксплуатации поддержки;
- трудности работы в команде разработчиков;
- чтобы использовать какой-то функционал повторно, придётся делать рефакторинг.

Микросервисная Архитектура

Micro-Services Architecture

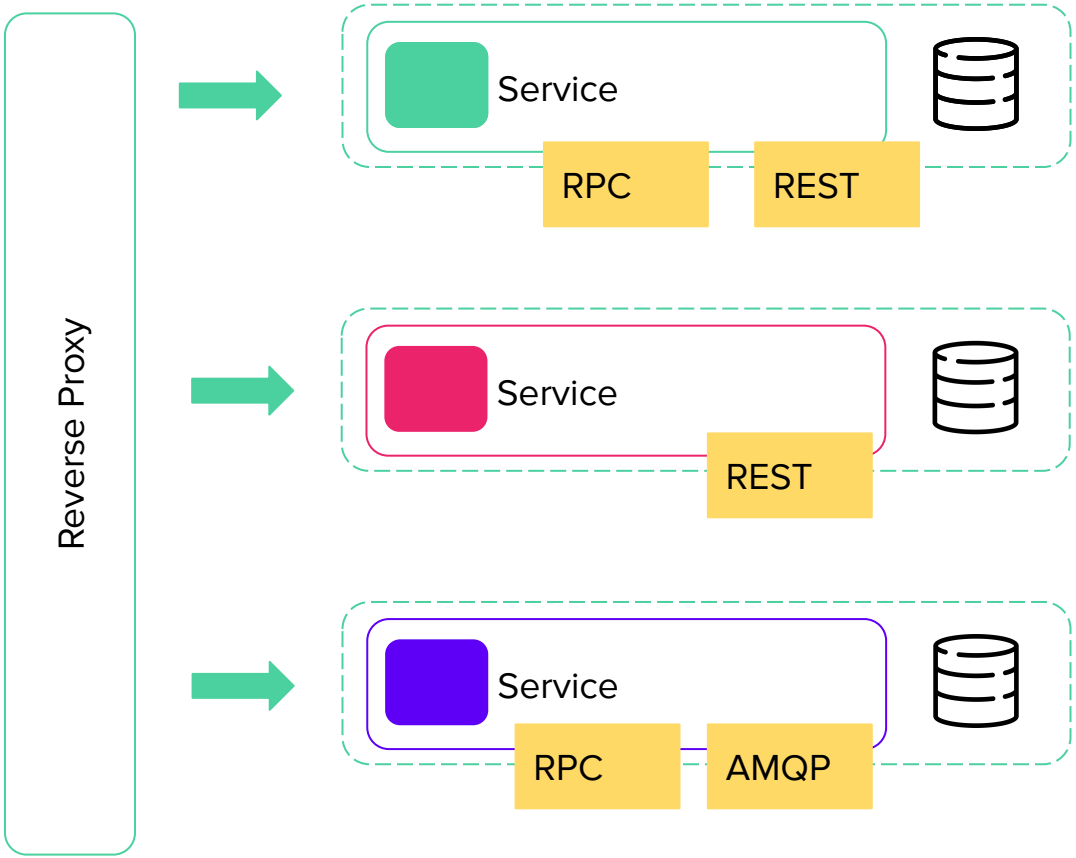


Слабо связанные сервисы взаимодействуют друг с другом для выполнения задач, относящихся к их бизнес-возможностя

Сервисы меньше, чем в монолитной среде.

Но микро — о бизнес-возможностях, а не о размере.

Протоколы



Различные протоколы взаимодействия

RPC

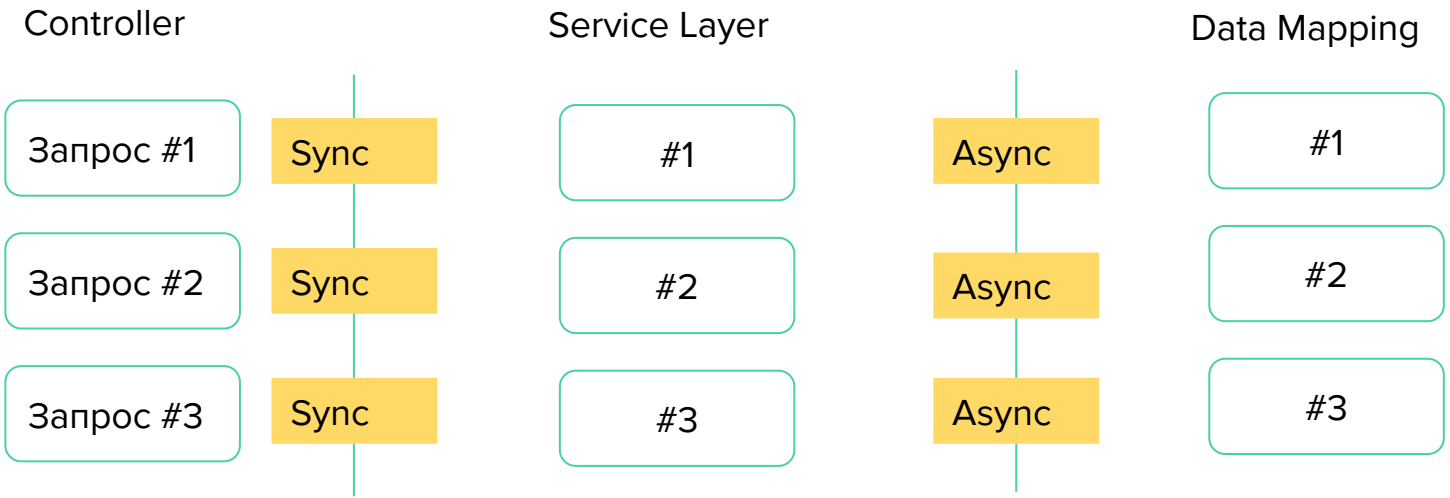
AMQP

Kafka

REST

Stateful Services

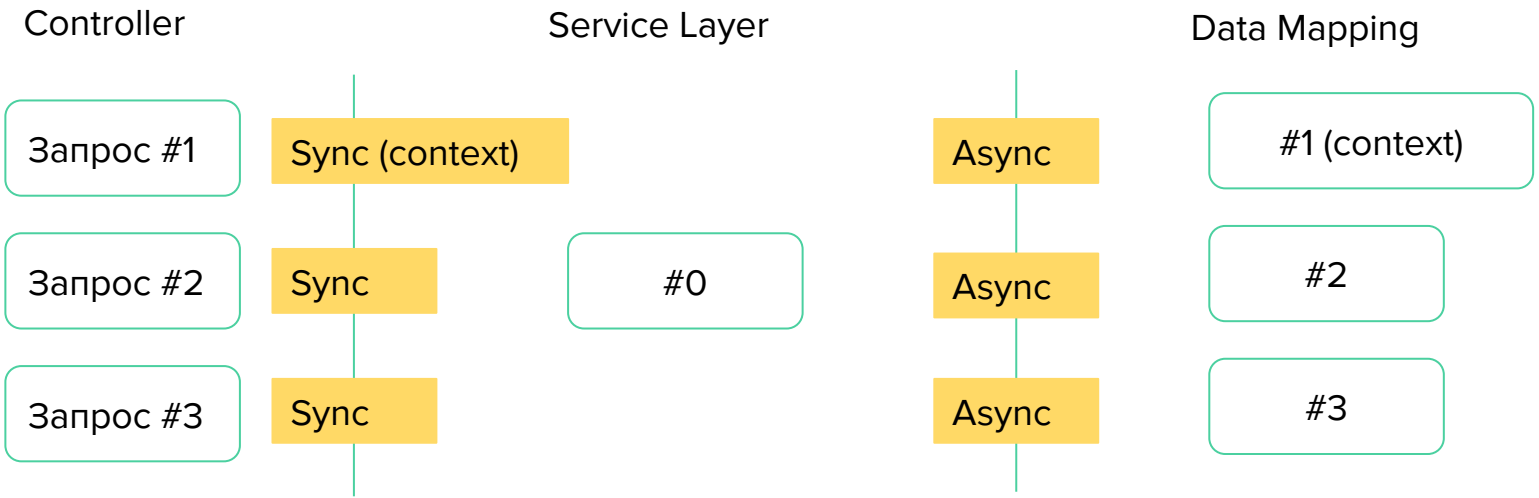
- stateless – не хранит состояние;
- stateful – добавляет объекту класса поля, которые нужны для выполнения запроса



источник

Stateless Services

- stateless – не хранит состояние;
- stateful – добавляет объекту класса поля, которые нужны для выполнения запроса



источник

Micro-Services Architecture

- Модульность повышает **гибкость управления командами** и улучшает шеринг технологий
- Размер сервисов сокращает сроки ожидания между разработкой различной функциональности и упрощает тестирование
- Нет привязки к технологии, используемой в других сервисах. **Возможность использовать лучшие технологии.**
- Лёгкий рефакторинг
- **Отказоустойчивость**
- **Масштабирование**
- Возможность независимого и быстрого развертывания выгодны для **непрерывной интеграции** (CI) и **непрерывного развёртывания** (CD)

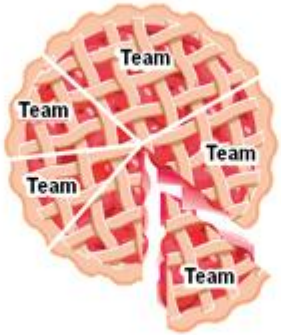
- Сложность реализации взаимодействия между сервисами
- Сложность интеграционного тестирования
- Трудоёмкость эксплуатации
- **Сложность в управлении транзакциями**
- **Низкая скорость разработки**

Monolithic vs SOA vs Microservices

до 1990
тесная связь

Каждое изменение в системе требует полной пересборки всей системы и проверки всех процессов

2000е
более слабая связь



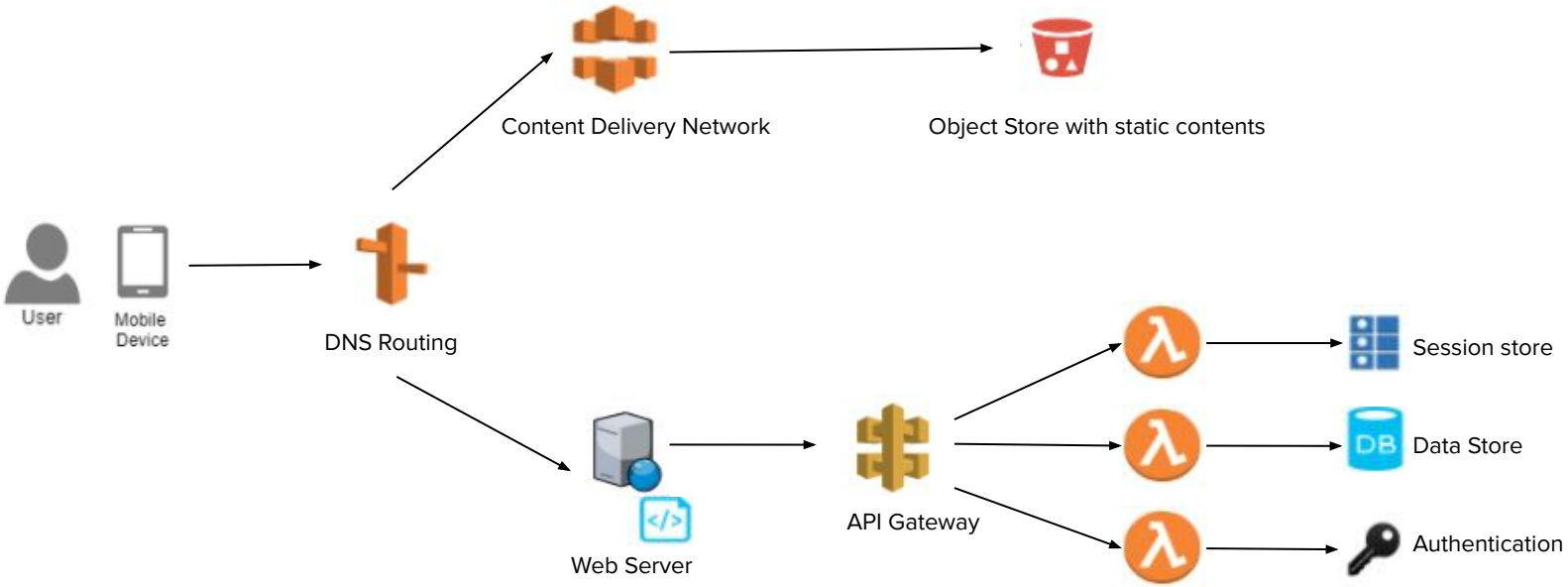
Отдельные элементы SOA более автономные, но должны координироваться общим интерфейсом

2010е
Отсутствие связи

Каждый сервис может поставляться без изменений в других сервисах. Позволяет внедрить **continuous delivery**

Домашнее задание: сделать сравнение достоинств и недостатков трёх видов Архитектур

Serverless



Бессерверные приложения — это управляемые событиями облачные системы, в которых разработка приложений основывается исключительно на сочетании сторонних сервисов, логики на стороне клиента и удаленных вызовов процедур, размещаемых в облаке («Функции как услуга»).

Serverless 

- Модульность повышает **гибкость управления командами** и улучшает шеринг технологий
- Размер сервисов сокращает сроки ожидания между разработкой различной функциональности и упрощает тестирование
- **Нет привязки к технологии**, используемой в других сервисах. **Возможность использовать лучшие технологии.**
- Лёгкий рефакторинг
- **Отказоустойчивость**
- **Масштабирование**
- Возможность независимого и быстрого развертывания выгодны для **непрерывной интеграции** (CI) и **непрерывного развёртывания** (CD)

- **Простота эксплуатации**
- **Сокращение сроков разработки**
- Сложность интеграционного тестирования
- Подходит только для stateless сервисов
- Сложность реализации взаимодействия между сервисами
- Сложность в управлении транзакциями

Кейсы 

Monolit	SOA	Micro-Services	Serverless
Приложения простые/ моно-функциональные MVP (для проверки гипотез, с низкой пользовательской нагрузкой)	Сложные корпоративные системы Часто применяется в банках и промышленных организациях	Многофункциональные , высоконагруженные системы с высокими требованиями к производительности Часто подвергаемые изменениями из-за гибкости внешних обстоятельств	При приложений или функций, которые часто воспроизводят одно и тоже действие: выгрузка файлов, расчёт маршрутов, обработка фото (применение фильтров или

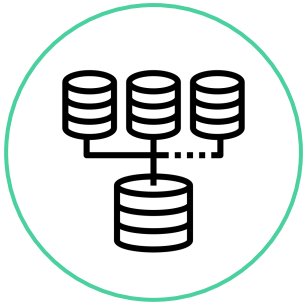
Интеграция

3

Типы интеграций

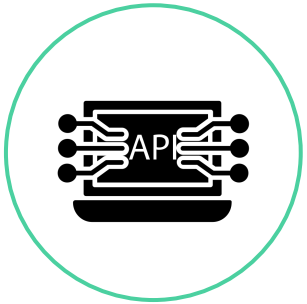
Файловое взаимодействие

Быстрая реализация на источнике, сложная реализация и обработка данных на приёмнике
ещё сложнее поддерживать целостность данных;
изменения одного сервиса влечет к изменению всех



Взаимодействие через Базу данных

Быстро реализовывать
сложно поддерживать целостность данных;
долгий процесс передачи данных;
изменения одного сервиса влечет к изменению всех



Сервисное (API) взаимодействие

Долго реализовывать
Легче поддерживать целостность данных;
При сохранении контракта взаимодействия - сервисы независимы;

Брокеры сообщений для взаимодействия через API

Распространённые брокеры сообщений

более производительная, но сложнее в эксплуатации

более простая, но легче

чаще используется .net разработке

подходит для работы с BigData и IoT

можно использовать в облаке AWS

Хранилище структур данных в памяти с открытым исходным кодом, используемое в качестве базы данных, кеша и брокера сообщений.

Риски при проектировании Архитектуры ИС

6

Риски проектирования ИС

Риски проектирования при разработке системы

Изобретение «велосипеда»

Сложность архитектуры программного обеспечения

Неоптимальный выбор языка программирования и других технологий и библиотек

Неоптимальный выбор структур данных

Неправильная структура базы данных

Технические риски, которые реализуются в виде отказов, простоев, потерь данных;

Риски бизнес-потерь, связанные с многообразием и неопределенностью бизнес-процессов.
Бизнес-процессы систематически изменяются, а это, в свою очередь, требует своевременного изменения и ИС.

Риски бизнес-потерь, возникающие при эксплуатации системы (бизнес-риски).
Например, отсутствие реакции или слишком большое время реакции на действия пользователя приводит к его отказу от использования данной системы

Итоги занятия

1. **К вопросу проектирования нужно подходить очень аккуратно!**

Итоги занятия

1

К вопросу проектирования нужно подходить очень аккуратно!

2

От хорошо проделанной работы системного аналитика зависит на сколько корректно бизнес-процессы будут разбиты на сервисы, какой набор технологий и способы интеграции будут выбраны. А отсюда и на работоспособность конечной системы и её отказоустойчивость.

Итоги занятия

1. **К вопросу проектирования нужно подходить очень аккуратно!**

2. От хорошо проделанной работы системного аналитика зависит на сколько корректно бизнес-процессы будут разбиты на сервисы, какой набор технологий и способы интеграции будут выбраны. А отсюда и на работоспособность конечной системы и её отказоустойчивость.

3. СА выполняет ключевую роль по формированию Архитектуры ИС

Итоги занятия

1. **К вопросу проектирования нужно подходить очень аккуратно!**

2. От хорошо проделанной работы системного аналитика зависит на сколько корректно бизнес-процессы будут разбиты на сервисы, какой набор технологий и способы интеграции будут выбраны. А отсюда и на работоспособность конечной системы и её отказоустойчивость.

3. СА выполняет ключевую роль по формированию Архитектуры ИС

4. В небольших компаниях или на небольших проектах СА может выполнять и значительно бОльшую часть по проектированию системы

Итоги занятия

1. **К вопросу проектирования нужно подходить очень аккуратно!**

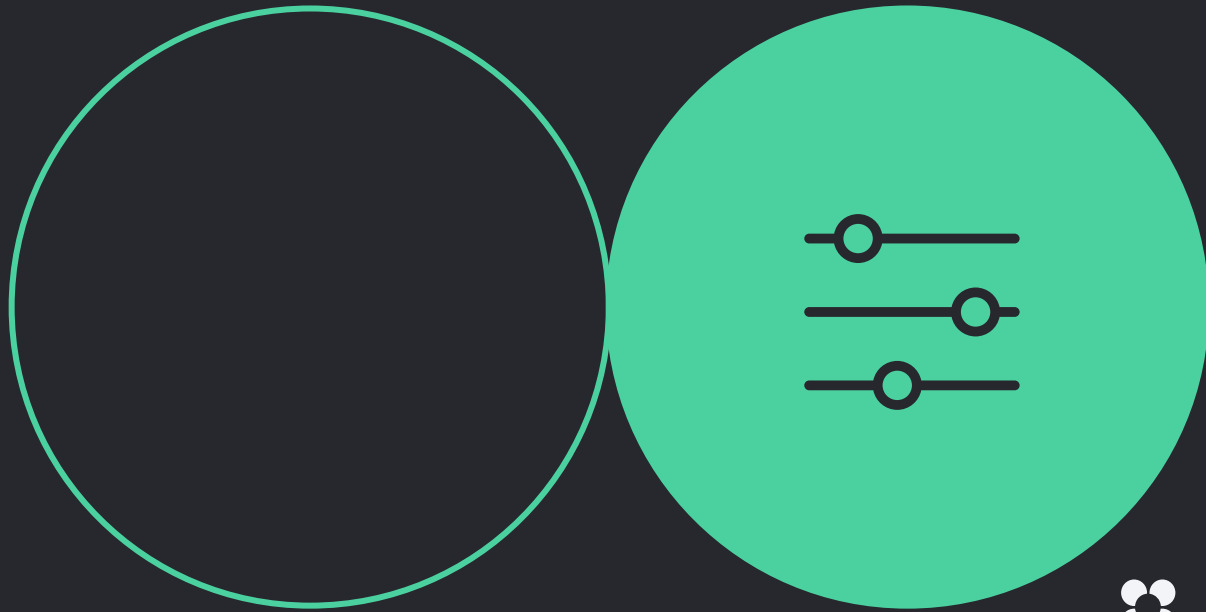
2. От хорошо проделанной работы системного аналитика зависит на сколько корректно бизнес-процессы будут разбиты на сервисы, какой набор технологий и способы интеграции будут выбраны. А отсюда и на работоспособность конечной системы и её отказоустойчивость.

3. СА выполняет ключевую роль по формированию Архитектуры ИС

4. В небольших компаниях или на небольших проектах СА может выполнять и значительно бОльшую часть по проектированию системы

5. Нет плохой архитектуры - есть неподходящие архитектуры, отсюда возникает большинство неработающих/плохо-функционирующих систем

Дополнительные ресурсы



Источники

- Максим Смирнов https://mxsmirnov.com
  Канал «Архитектура ИС» в Telegram: https://t.me/it_arch
- https://proglib.io/u/kapo4ka/posts (статья про микро-сервисы)
- https://www.sumologic.com/blog/serverless-aws/ (статья про serverless)
- “Migrating to Cloud-Native Application Architectures” Мэтт Стайн
- “Распределенные вычислительные системы. Лекции” Бурдаков А.В.
- “Распределенные вычислительные системы” Танненбаум
- ДеМарко Т., Листер Т. Вальсируя с медведями: управление рисками в проектах по разработке программного обеспечения
- ГОСТ Р 57100-2016 Системная и программная инженерия. Описание архитектуры
- ГОСТ Р ИСО/МЭК 18384-1-2017 Информационные технологии (ИТ)
- EIP Tutorial
  https://warren2lynch.medium.com/enterprise-integration-patterns-eip-tutorial-f6d7134f67ae
- DFD Tutorial
  https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/

Архитектура ИС

rigataullina@gmail.com

fb.com/rigataullina

Регина Гатауллина
Product manager Bookmate