Занятие 17

#system + #analyst

Основы интеграций ИС

Люба Вайгель

Ментор ASAP Education

Давайте вспомним

- Чем веб-приложение отличается от веб-сайта?
- Какие виды архитектуры вы знаете?
- В чем разница между монолитами и микросервисами?
- Какие преимущества и недостатки есть у микросервисной архитектуры?
- Какие способы взаимодействия между микросервисами вы знаете?
- Какие микросервисы можно выделить в приложении для доставки цветов?

План занятия

Интеграции информационных систем

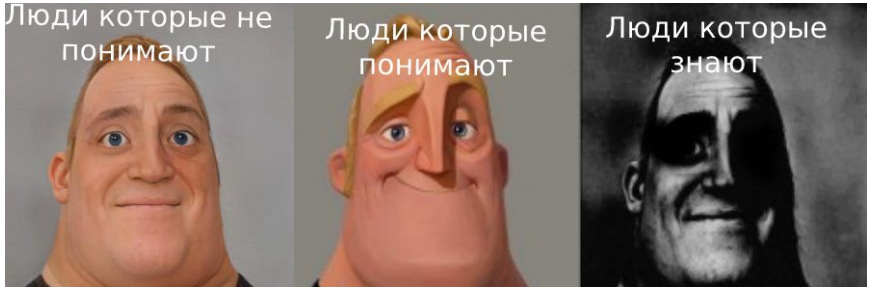
Что это такое, какие виды бывают

Причём здесь аналитик

Роль системного аналитика
в проектировании системных интеграций

Системная интеграция — это объединение отдельных автоматизированных процессов и средств управления, что предполагает не только использование уже работающих систем, но и создание новых

Виды интеграций



1970-е — **Файловый обмен**

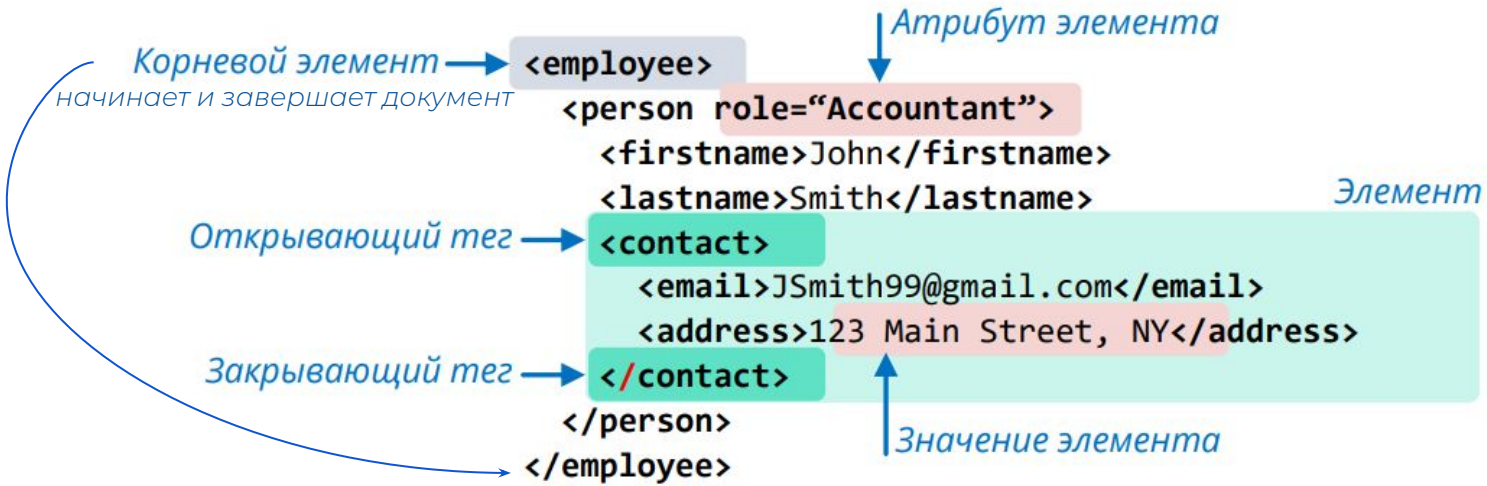
1980-е — **Интеграция через общую БД**

1990-е — **Интеграция через веб-сервисы**
(удалённый вызов процедуры)

2000-е — **Интеграция через обмен сообщениями**
(интеграция с помощью внешнего сервиса через брокеры сообщений)

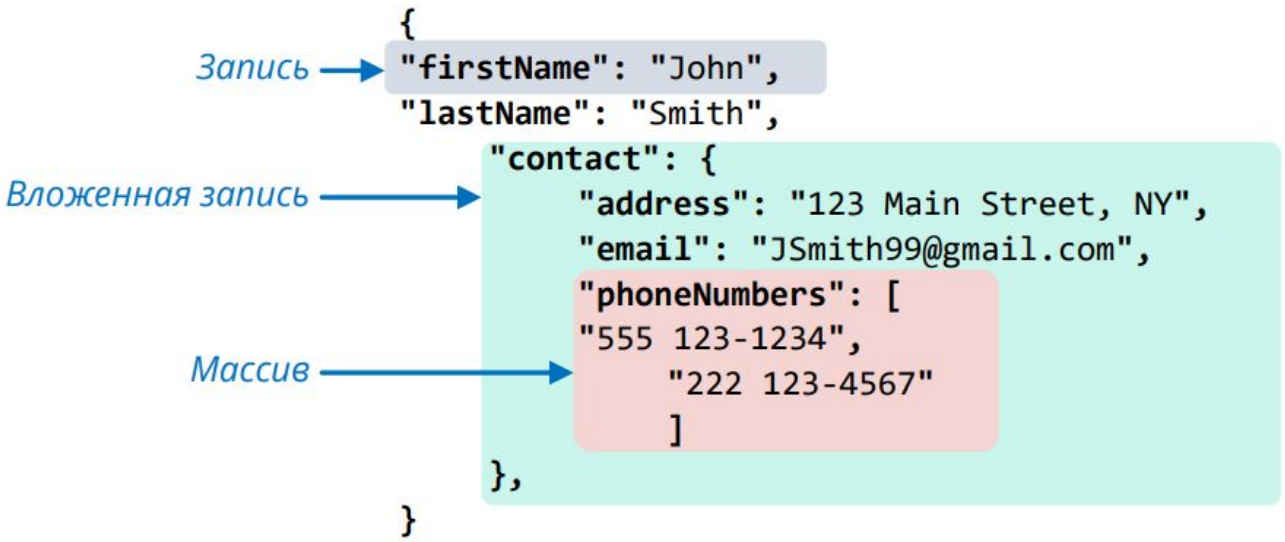
Формат данных: XML

XML (eXtensible Markup Language) – язык с простым формализованным синтаксисом, удобный для чтения как компьютером, так и человеком



Формат данных: JSON

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript



Файловый обмен

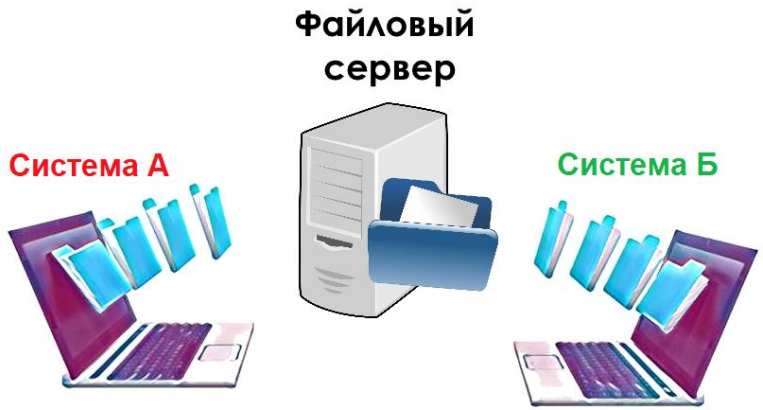
Способ интеграции, при котором данные между системами передаются в виде файлов:

- система-источник размещает на файловом сервере файлы,
- система-приёмник забирает их с файлового сервера и использует для своих нужд

Формат файлов: любой, чаще всего XML

Протокол взаимодействия:

- FTP (File Transfer Protocol)
- FTPS (FTP + SSL)
- SFTP (Secure File Transfer Protocol)



Файловый обмен

asap. education

ПРЕИМУЩЕСТВА

- Просто реализовать и отладить
- Можно использовать любой формат и размер файлов
- Асинхронный обмен
- Легко масштабировать

НЕДОСТАТКИ

- Пакетная, а не потоковая передача данных
- Задержка обработки данных
- Нет транзакционности
- Нужно уделять повышенное внимание безопасности данных
- Многое нужно делать вручную

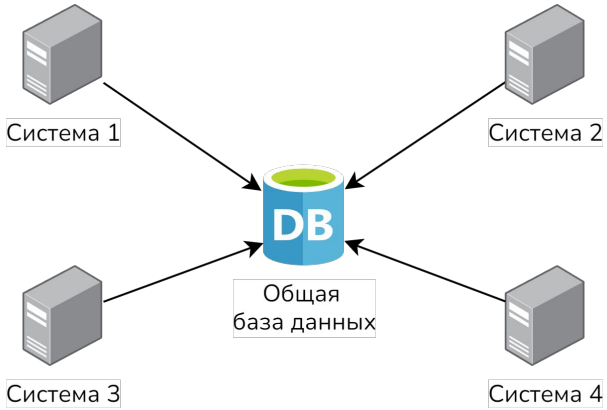
Файловый обмен

Хорошо подходит для интеграции, когда:

- нужно передавать файлы больших объёмов
- нужно передавать агрегированные пакетные данные (например, выгрузка продаж за неделю)
- не нужен обмен данными в реальном времени
- надёжность системы-приёмника низкая

Общая БД

Способ интеграции ИС,
при котором разные системы
обращаются к одному
хранилищу данных



Особенности реализации

- **Нативное подключение к БД** из приложения: сервер и хост БД, учётные данные пользователя, под которым осуществляется доступ прописываются в коде
- **Подключение к БД через DBC** (Database Connectivity) API — интерфейс для подключения и выполнения запросов к базе данных

Общая БД

asap. education

ПРЕИМУЩЕСТВА

- Целостность и непротиворечивость данных: все системы имеют доступ к одному хранилищу и одинаковым данным
- Данные не дублируются
- Просто реализовать

НЕДОСТАТКИ

- БД - единая точка отказа всех систем
- Много внимания нужно уделять масштабированию БД
- Большой объём ручных изменений из-за высокого уровня связанности: все внешние системы должны учитывать все возможные изменения в структуре хранения
- Нужно уделять повышенное внимание безопасности

Общая БД

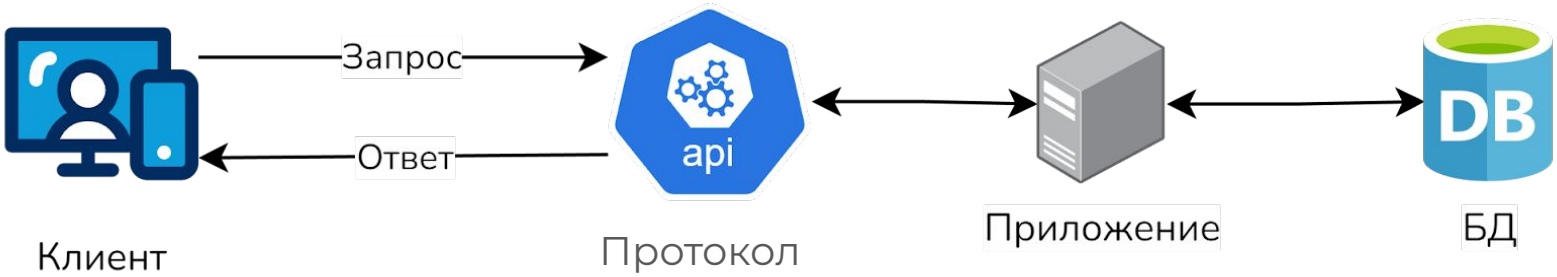
Хорошо подходит для интеграции, когда:

- все интегрируемые ИС внутренние
- интегрируемые ИС относятся к одной доменной области и оперируют одинаковыми данными
- обмен данными происходит по расписанию или эпизодически, обмен частый и в реальном времени
- объём данных не слишком большой (не крупные файлы)
- необходимо сразу видеть изменения, происходящие в ИС

Удалённый вызов процедуры

Это способ интеграции ИС, при котором *клиент* удалённо вызывает функцию или процедуру на сервере по определённому *протоколу* (набору правил).

Сервер в свою очередь выполняет какой-то процесс или формирует набор данных и передаёт ответ клиенту.



Удалённый вызов процедуры

Несмотря на то, что «Удалённый вызов процедур» — наименование конкретной технологии RPC (Remote Call Procedure), под этим термином объединяются различные технологии и подходы:

- **RPC** — Remote Procedure Call;
- **CORBA** — Common Object Request Broker Architecture;
- **SOAP** — Simple Object Access Protocol
- **REST** — Representational State Transfer
- **GraphQL** — Graph Query Language
- **gRPC** — Google Remote Procedure Calling

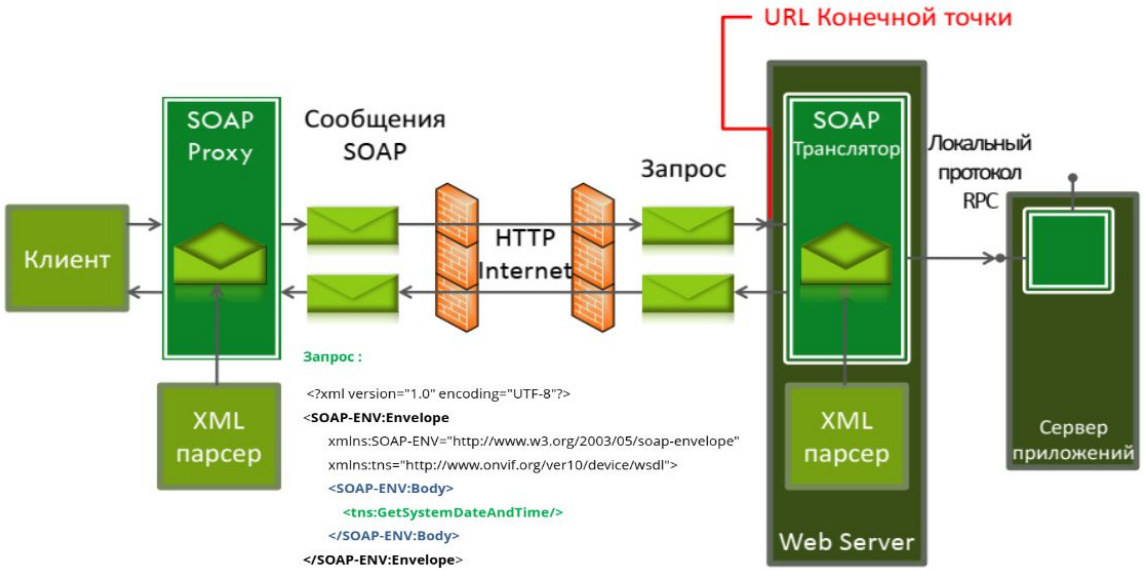
SOAP

SOAP (Simple Object Access Protocol) — протокол взаимодействия веб-сервисов. Протокол имеет набор жёстких правил, без соответствия которым взаимодействие по SOAP невозможно

Транспорт данных:

- HTTP (чаще всего)
- SMTP
- TCP

Формат данных: XML



SOAP

asap. education

ПРЕИМУЩЕСТВА

- Стандартизация протокола
- Поддерживает стандарты безопасности
- Поддерживает гарантированную доставку сообщений
- Можно использовать разные транспортные протоколы
- Хорошо подходит для сложных операций с контролем за состоянием операций

НЕДОСТАТКИ

- Сложная разработка и отладка
- Низкая производительность из-за тяжеловесности XML
- SOAP-запросы и ответы избыточны из-за большого количества метаданных, растёт сетевой трафик
- Из-за строгого контракта сложно вносить изменения
- Ограниченный стек из-за строгой привязки к XML

SOAP

asap. education

Хорошо подходит для интеграции, когда:

- надёжность и безопасность важнее скорости
- строгие контракты важнее высокой частоты изменений
- логика удаленных процедур сложна
- нужны транзакционные операции

Перерыв

REST

REST это архитектурный стиль, набор принципов проектирования архитектуры распределенных веб-приложений

REST не имеет собственных методов и не ограничен никакими протоколами

REST API — применение архитектурного стиля REST к проектированию API

Транспорт данных: HTTP/HTTPs

Формат данных: JSON (чаще всего)

6 принципов REST:

1. Клиент-серверная архитектура
2. Отсутствие состояний
3. Кэширование
4. Единообразие интерфейса
5. Многоуровневая система
6. Код по запросу

REST API

asap. education

ПРЕИМУЩЕСТВА

- Простота реализации
- Универсальность: можно использовать любой формат данных для передачи полезной нагрузки
- Поддерживает кэширование
- Поддерживает разные методы аутентификации

НЕДОСТАТКИ

- Отсутствие стандартизации
- Низкая производительность: каждый запрос требует отдельного HTTP-соединения
- Нет сохранения состояния
- Не подходит для большого объема данных
- Сложно реализовать транзакционные операции над несколькими ресурсами одновременно

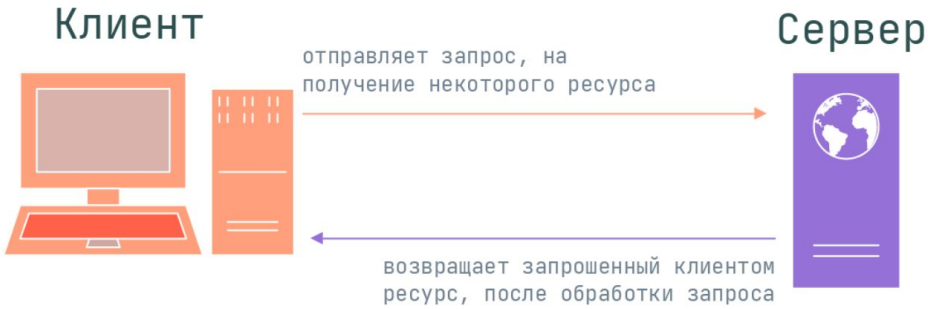
REST API

Хорошо подходит для интеграции, когда:

- нужно сделать быстро и просто
- скорость важнее надёжности и безопасности
- нужно часто вносить изменения в контракты взаимодействия
- бизнес-логика не очень сложная и операции над бизнес-сущностями ограничены набором CRUD-операций
- экономия трафика не важна

HTTP (HyperText Transfer Protocol)

Протокол прикладного уровня, используемый для доступа к ресурсам Интернета



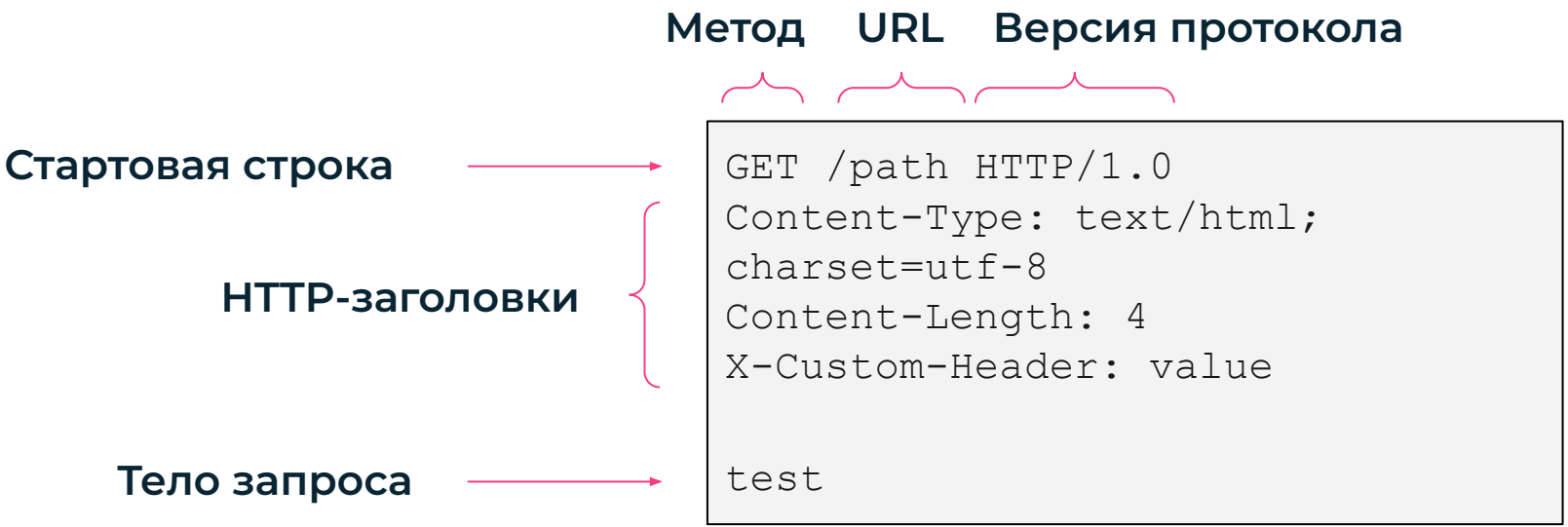
Данные между клиентом и сервером передаются через **HTTP-сообщения**:

- **Запросы** (HTTP Requests) — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Основой запроса является HTTP-заголовок
- **Ответы** (HTTP Responses) — сообщения, которые сервер отправляет *в ответ* на клиентский запрос

HTTP-сообщение: структура

- **Стартовая строка** (start line) — используется для описания версии используемого протокола и другой информации — вроде запрашиваемого ресурса или кода ответа
- **HTTP-заголовки** (HTTP Headers) — несколько строчек текста в определенном формате, которые либо уточняют запрос, либо описывают содержимое *тела* сообщения
- Опциональное **тело сообщения**, которое содержит данные, связанные с запросом, либо документ (например HTML-страницу), передаваемый в ответе

HTTP-запрос


Метод
URL
Версия протокола
Стартовая строка
HTTP-заголовки
Тело запроса
GET /path HTTP/1.0
Content-Type: text/html;
charset=utf-8
Content-Length: 4
X-Custom-Header: value
test

Методы HTTP

Каждый HTTP-запрос состоит из метода, который указывает действие, которое должно быть выполнено для указанного ресурса

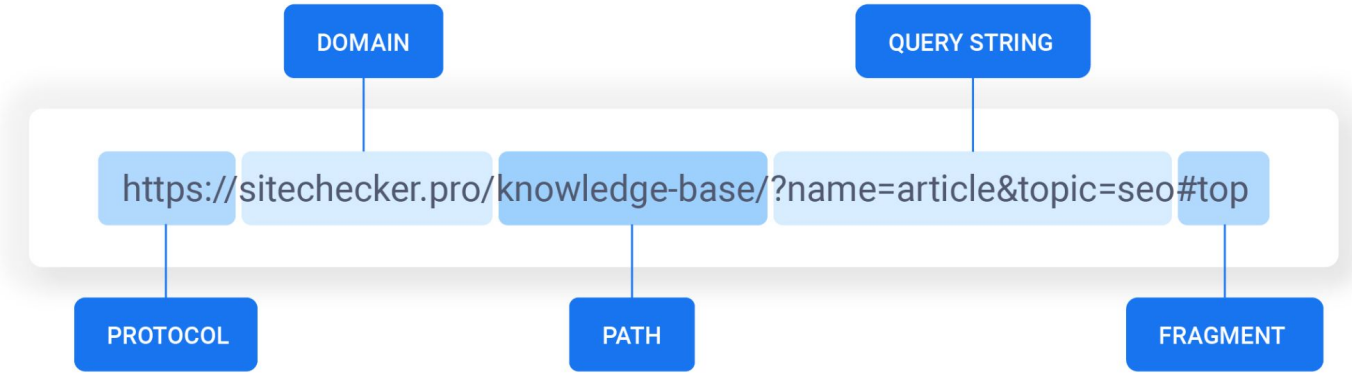
Основные методы:

- **GET** — получение ресурса
- **POST** — создание ресурса
- **PUT / PATCH**— полное / частичное обновление ресурса
- **DELETE** — удаление ресурса

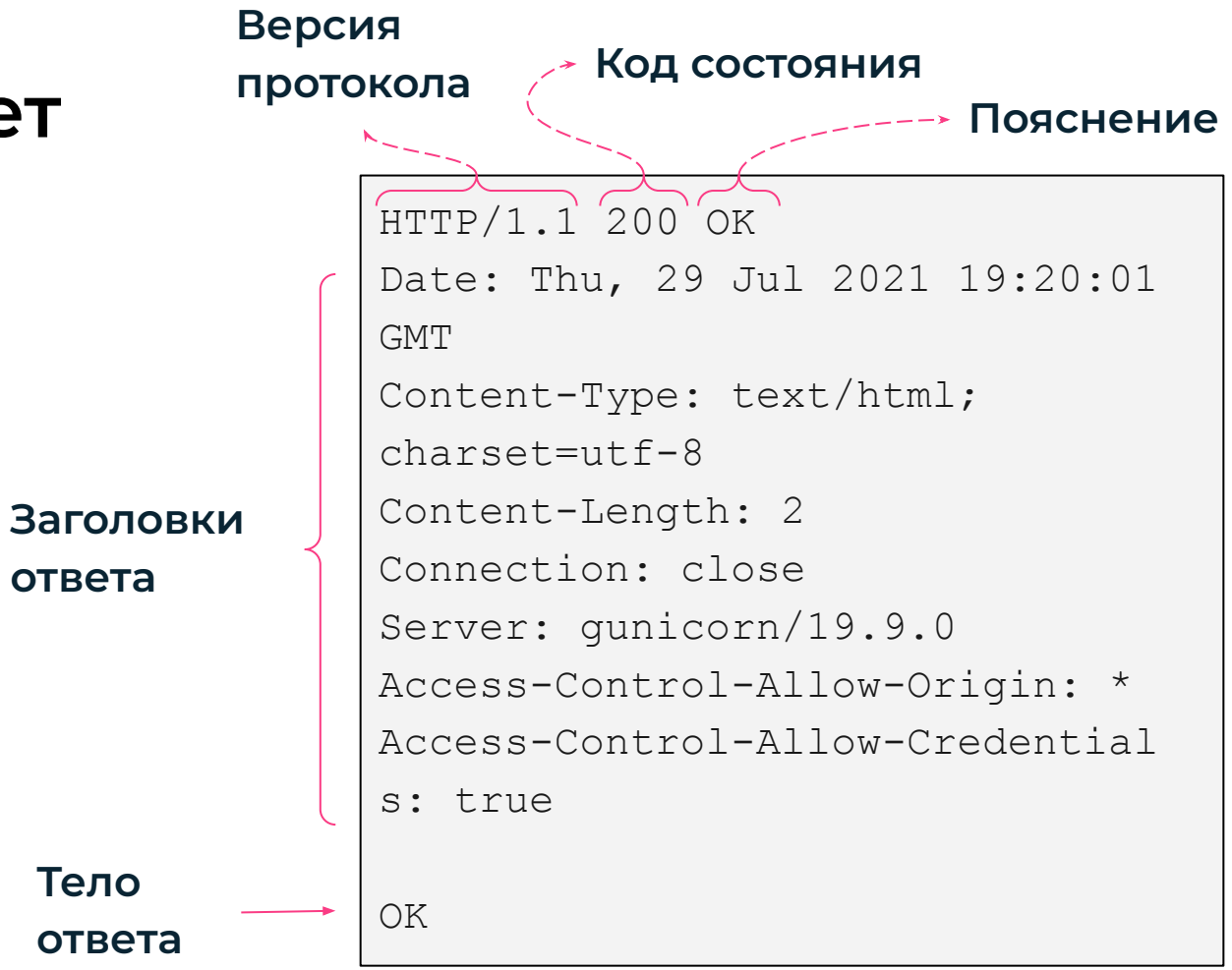
Свойства методов HTTP

- **Идемпотентные методы** - это методы, которые либо не изменяют состояние в базе данных, либо изменяют состояние только при первом запросе.
  В случае повторной отправки идентичного запроса, состояние в базе данных не изменяется (**GET, PUT, DELETE, HEAD и OPTIONS**)
- **Безопасные методы** - это методы, которые не изменяют состояние в базе данных (read only методы). Все безопасные методы также являются идемпотентными (**GET, HEAD и OPTIONS**)
- Безопасные методы не меняют состояние базы данных, в то время как идемпотентные методы могут внести изменения при первом запросе, но последующие идентичные запросы уже не будут менять состояние в базе данных

Структура URL



HTTP-ответ



asap. education

Коды состояния

asap. education

2xx Success

- 200 Success / OK

3xx Redirection

- 301 Permanent Redirect
- 302 Temporary Redirect
- 304 Not Modified

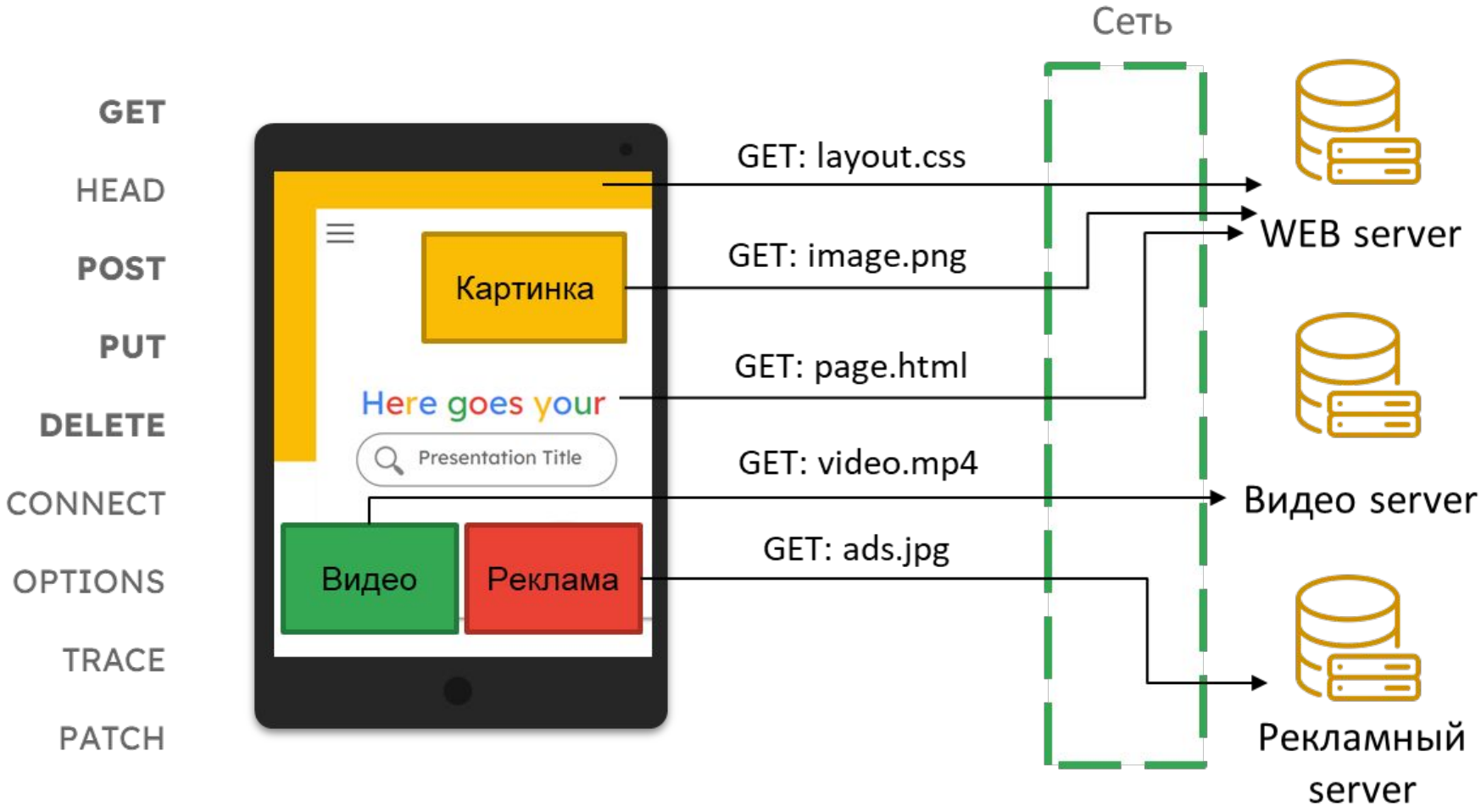
4xx Client Error

- 401 Unauthorized Error
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed

5xx Server Error

- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout

Все коды состояния с пояснениями здесь


GET
HEAD
POST
PUT
DELETE
CONNECT
OPTIONS
TRACE
PATCH
Сеть
Картинка
Here goes your
Presentation Title
Видео
Реклама
GET: layout.css
GET: image.png
GET: page.html
GET: video.mp4
GET: ads.jpg
WEB server
Видео server
Рекламный server

Примеры открытых API

Российские государственные интернет-сервисы

- Цифровой профиль гражданина РФ
- Процедура интеграции с ЕСИА государственных и муниципальных организаций
- Вход на сайт через Госуслуги (ЕСИА)
- API ФИАС (КЛАДР)

Популярные интернет-сервисы

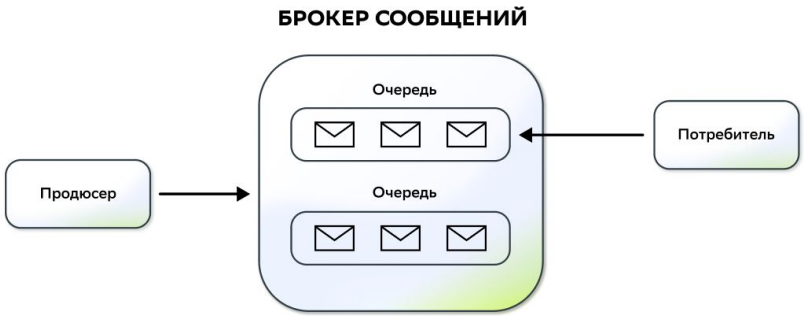
- Public APIs @ Swagger Hub
- Sunset and sunrise times API
- API ЮKassa
- API Dadata
- JIRA Server platform REST API
- Trello REST API Reference
- YouTube API Reference

Обмен сообщениями

Очереди предоставляют буфер для временного хранения сообщений и конечные точки, которые позволяют подключаться к очереди для отправки и получения сообщений в ***асинхронном*** режиме

В сообщениях могут содержаться запросы, ответы, ошибки и иные данные, передаваемые между программными компонентами

Обмен сообщениями



метод Pull

периодический опрос очереди получателем по поводу наличия новых сообщений

метод Push

отправка уведомления получателю в момент прихода сообщения

Обмен сообщениями

asap. education

ПРЕИМУЩЕСТВА

- Отделение логически независимых компонентов друг от друга
- Улучшение масштабируемости
- Балансировка нагрузки
- Повышение надежности
- Безопасность

НЕДОСТАТКИ

- Непростая для освоения система
- Чем больше очередей, тем сложнее отладка
- Нужно выбирать стратегию отправки сообщений, а они все сложные для реализации и имеют свои недостатки: придется идти на компромиссы

Обмен сообщениями

Хорошо подходит для интеграции, когда:

- В рамках системы есть действия, которые требуют для выполнения много времени и ресурсов и не требуют немедленного результата
- Система достаточно сложна и состоит из отдельных сервисов - для их координации можно использовать брокер сообщений
- Какое-то действие системы состоит из отдельных этапов, каждый из которых выполняется отдельным элементом системы, то в этом случае брокер сообщений может выступить в роли своеобразной «доски уведомлений»

Перерыв

Причём здесь аналитик

Аналитик и этапы разработки

- **Проектирование**: аналитик *собирает требования* к продукту или функциональности, *продумывает реализацию*, в том числе интеграции между системами, *описывает решение*
- **Разработка**: аналитик *консультирует* разработчиков по возникающим вопросам, принимает решения при возникновении проблем
- **Тестирование**: аналитик *консультирует* тестировщиков по возникающим вопросам, иногда аналитика привлекают к проектированию тест-кейсов

Почему аналитик проектирует интеграции

- Знает **бизнес контекст** и проектирует интеграцию, находя компромисс между удобством для пользователя и для разработки
- Может спроектировать **единое решение** для разных платформ (например, iOS и Android), учитывая их особенности
- Как правило, лучше **коммуницирует** с людьми и умеет договариваться
- Чем больше людей привлечено к проектированию и валидации результата, тем меньше шанс что-то упустить

Как проектировать

- Определить основную **цель** интеграции – какую задачу предстоит решить
- Максимально детально расписать все возможные **сценарии интеграционного бизнес-процесса**
- Выбрать **способ интеграционного взаимодействия**
- Определить **данные**, которыми необходимо обмениваться: что вы передаете, что принимаете, что из этого будет обязательным по спецификации и в каком формате данные будут переданы
- Досконально изучить **документацию интегрируемой системы**

Вопросы

- Какие виды интеграций вы знаете?
- Что такое API? Какую функцию выполняет?
- В чем разница между SOAP и REST?
- Что такое JSON и XML? Для чего они нужны и что в них описано?
- Какие методы HTTP вы знаете?
- Какие коды состояния HTTP вы знаете?
- В чем разница между методами PUT и PATCH?
- Что такое идемпотентность? Какие методы HTTP являются идемпотентными?
- Назовите принципы REST
- Расскажите принцип работы брокеров сообщений