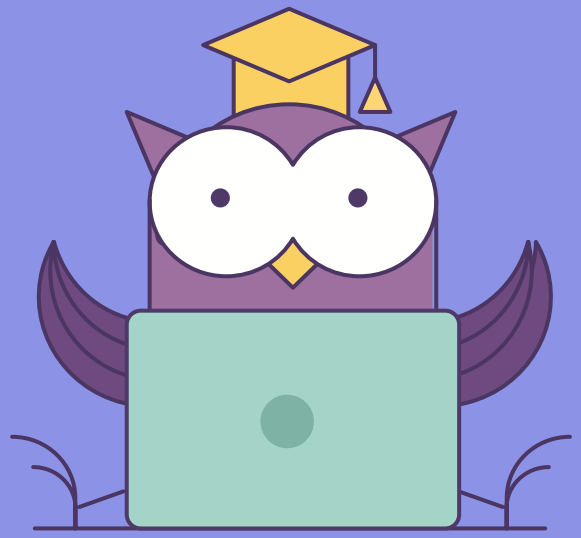
ОНЛАЙН-ОБРАЗОВАНИЕ

Проверить, идет ли запись!

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

GraphQL & gRPC

Архитектор ПО

Жигалко Илья

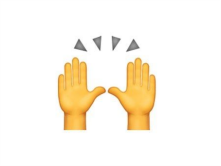
О себе

@izhigalko

Жигалко Илья

- Занимаюсь развитием DevOps в ДИТ Сеть продаж ПАО Сбербанк
- Около 12 лет работаю в IT
- Люблю разбираться в механике

Правила вебинара



Активно участвуем

Задаем вопрос в чат или голосом

slack

Off-topic обсуждаем в Slack #канал группы или #general

Вопросы вижу в чате, могу ответить не сразу

Цели вебинара

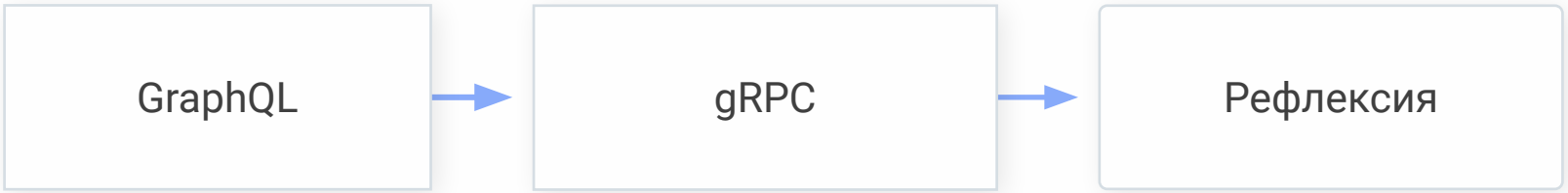
OTUS

После занятия вы сможете:

1. Ответить на вопрос, что такое GraphQL

2. Ответить на вопрос, что такое gRPC

Маршрут вебинара



01

GraphQL

GraphQL - язык запросов

- **НЕ ПРОТОКОЛ**
- Разработан Facebook
- Предоставляет схему запросов
- Позволяет получить все данные в один запрос

Что предлагает:

- Схема запросов

Типы данных

Операции над данными: запросы и мутации

```
schema {
  query: Query
}

type Query {
  books(limit: Int): [BookType]
}

type BookType {
  id: ID!
  title: String!
  description: String!
  author: AuthorType!
}
```

Что предлагает:

- Схема запросов
- Получаем только те данные, что запрашиваем

```
{
  books(limit: 1){
    title
  }
}
```

```
{
  "data": {
    "books": [
      {
        "title": "A Wizard of Earthsea"
      }
    ]
  }
}
```

Что предлагает:

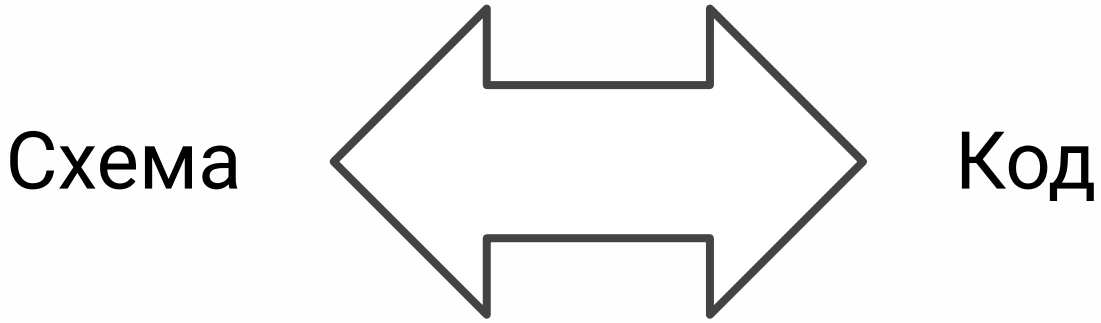
- Схема запросов
- Получаем только те данные, что запрашиваем
- Получаем несколько сущностей в один запрос

```
{
  author(id: 1) {
    id,
    books {
      id
    }
  }
}
```



```
{
  "data": {
    "author": {
      "id": "1",
      "books": [
        {
          "id": "1"
        },
        {
```

Реализация



- Доступна реализация под различные ЯП - https://graphql.org/code/
- В качестве транспортного протокола используется HTTP
- В качестве формата данных используется JSON

Проблемы в использовании:

- Сложность в разработке бекенда

Select N + 1

Проблемы в использовании:

- Сложность в разработке бекенда
- Сложность мониторинга

Всегда отвечает 200

Один эндпоинт

Проблемы в использовании:

- Сложность в разработке бекенда
- Сложность мониторинга
- Безопасность использования

Открытая схема

Зацикленные запросы

Произвольные запросы

```
{
  books {
    author {
      books {
        author {
          books {
            title
          }
        }
      }
    }
  }
}
```

GraphQL + Frontend

В основном используется как:

- Backend for frontend
- API для внешних пользователей

GraphQL + Backend

В основном не используется по причинам:

- Предпочтительна конкретика операций и их результата
- Отсутствие преимуществ от свободы

GraphQL vs REST?

Вопросы, которые задаём при выборе:

- Необходимо получать множество данных с несколькими уровнями зависимостей? Или можно обойтись простыми запросами?
- Есть ограничение в пропускной способности сети между сервисом и потребителем?
- Критична ли ошибка в консистентности отдаваемых сервисом данных или ошибка в типах данных?

02

gRPC

gRPC - фреймворк

- **НЕ ПРОТОКОЛ**
- Основан на разработках Google - Stubby
- Использует схема и кодогенерацию по ней
- Нет необходимости реализовывать транспорт

Что предлагает:

- Возможность описания схемы данных
- Возможность кодогенерации на стороне сервера и клиента

RPC

```
syntax = "proto3";

import "google/protobuf/timestamp.proto";

service BooksService {
  rpc GetBooks(Empty) returns (stream Book);
}

message Empty {
}

message Author {
  string name = 1;
  google.protobuf.Timestamp birthday = 2;
}

message Book {
  string title = 1;
  string description = 2;
  Author author = 3;
}
```

Что предлагает:

- Возможность описания схемы данных (IDL)
- Возможность кодогенерации на стороне сервера и клиента
- Возможность использовать потоки

```
syntax = "proto3";

import "google/protobuf/timestamp.proto";

service BooksService {
  rpc GetBooks(Empty) returns (stream Book);
}

message Empty {
}

message Author {
  string name = 1;
  google.protobuf.Timestamp birthday = 2;
}

message Book {
  string title = 1;
  string description = 2;
  Author author = 3;
}
```

Реализация

- Доступна реализация под различные ЯП - https://grpc.io/docs/languages/
- В качестве транспортного протокола используется HTTP2
- В качестве формата данных используется Protobuf

Проблемы в использовании:

Транспорт реализуется библиотекой

- Сложность Runtime

Проблемы в использовании:

Держит соединение

- Сложность Runtime
- Сложность балансировки

Проблемы в использовании:

Бинарный протокол

- Сложность Runtime
- Сложность балансировки
- Сложность отладки

gRPC + Backend

Часто используется, но нужно учесть, что:

- Существуют более легкие способы интеграции

На что стоит обратить внимание при выборе:

- Важно ли наличие схемы сервиса и идентичности реализации серверной и клиентской части?
- Есть ограничение в пропускной способности сети между сервисом и потребителем?

gRPC + Frontend

В основном не используют, потому что:

- Невозможно использовать напрямую. Нужен гейт для преобразования, например grpc-web.
- Сложность отладки

Рефлексия

OTUS

После занятия вы сможете:

1. Ответить на вопрос, что такое GraphQL

2. Ответить на вопрос, что такое gRPC

Список материалов для изучения

- https://graphql.org/
- https://engineering.fb.com/2015/09/14/core-data/graphql-a-data-query-language/
- https://stablekernel.com/article/advantages-and-disadvantages-of-graphql/

Заполните, пожалуйста, опрос о занятии: https://otus.ru/polls/27727/

Спасибо
за внимание!