С нуля до GraphQL за 40 минут

Дмитрий Швецов @iamdidev

GraphQL - это приятно и быстро

- Без версионирования
- Документация из коробки
- Контракт между front и back разработчиками
- Работает по любому транспорту
- Работает с любым back-end
- Клиентские и серверные библиотеки облегчают разработку
- Позволяет получить именно те данные, которые нужны (проблема overfetching и underfetching)
- Когда думаешь деревом графа
- Не серебряная пуля

Обычный день разработчика REST API

GET /products

Разные клиенты - разные требования

GET /api/playstation/products

GET /api/mobile/products

Альтернатива

GET /api/products?version=gaming

GET /api/products?version=mobile

Еще альтернатива

GET /api/products?partial=full

GET /api/products?partial=minimal

Обобщение приводит нас к языку запросов

GET /api/products?include=author&fields[products]=name,price

Хорошая новость - с такой проблемой столкнулись другие разработчики

Февраль 2012 началась работа над прототипом GraphQL

Решение проблемы

```
query {
  viewer {
    name
    bio
    company
  }
}
```

```
{
  "data": {
    "viewer": {
      "name": "iamdi",
      "bio": "Positive Full-Stack Web Developer.",
      "company": "Jerry.ai",
    }
  }
}
```

GraphQL публичный релиз

К 2015 году эта же команда разработчиков переработала и опубликовала GraphQL как open source проект в виде спецификации.

Airbnb, Twitter, Medium были первыми компаниями, принявшие GraphQL

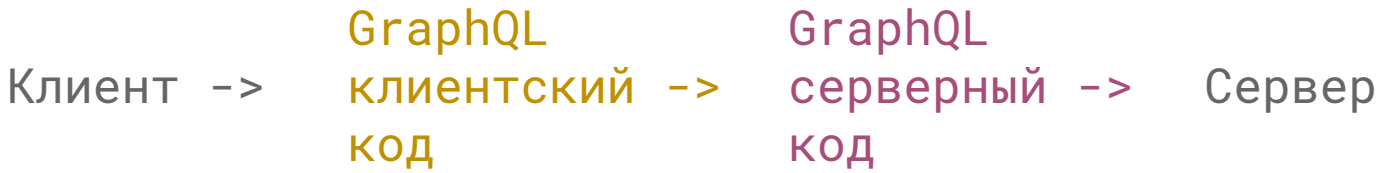
Github, Netflix, Coursera, Shopify, Heroku и многие другие начали использовать GraphQL позже

Что такое GraphQL?

Спецификация, описывающая

- Язык запросов
- Типизацию для API
- Валидацию запросов
- Движок исполнения запросов
- Формат ответа

GraphQL - это слой между клиентом и данными



GraphQL не зависит от

- Транспорта (HTTP, web-sockets, gRPC)
- Языка реализации как back-end, так и front-end
- Источника данных (другие API, базы данных, файлы, вычисления на лету)

Из чего состоит GraphQL сервер и клиент

- Schema
- Types
- Resolvers
- Query
- Mutation
- Subscriptions

GraphQL сервер

Как это выглядит на сервере

Schema (схема данных, тип Query)

```
schema {
  query: Query
  mutation: Mutation
}
```

```
type Query {
  """
  The currently authenticated user
  """
  viewer: User!
}
```

Как это выглядит на сервере

Object Types (тип объектов данных)

```
type User {
  id: ID
  name: String
  login: String!
  createdAt: DateTime!
  isBountyHunter: Boolean!
}
```

Как это выглядит на сервере

Scalar (Скалярные типы данных)

- ID
- String
- Int
- Float
- Boolean
- можно делать свои!

Как это выглядит на сервере

Object Types (тип объектов данных)

```
type User {
  repository(name: String!): Repository
}
```

```
type Repository {
  name: String!
  description: String
}
```

Как это выглядит на сервере

- Wrapper (типы обертки) List и NotNull
- Enum (перечисляемый тип)
- Interface (интерфейс)
- Union (объединение типов)

Как это выглядит на сервере

Schema (схема данных, тип Mutation)

```
schema {
  query: Query
  mutation: Mutation
}


type Mutation {
  “Adds a comment to an issue or pull request”
  addComment(input: AddCommentInput!): AddCommentPayload
}
```

GraphQL клиент

Как это выглядит на клиенте - Query

```
query ViewerQuery {
  viewer {
    name
    bio
  }
}
```

```
{
  "data": {
    "viewer": {
      "name": "iamdi",
      "bio": "Positive Full-Stack Web Developer.",
       ...
```

Как это выглядит на клиенте - Query

```
query ViewerQuery {
  viewer {
    organizations(first: 10) {
      edges {
        node {
          name
        }
      }
    }
  }
}
```

```
{
  "data": {
    "viewer": {
      "organizations": {
        "edges": [
          {
            "node": {
              "name": "Jerry, Inc"
            }
          },
          {
            "node": {
              "name": "Yabeda"
            }
          },
          ...
```

Как это выглядит на клиенте - Variables

```
query ViewerQuery(
  $avatarSize: Int = 200
  $orgFirst: Int!
) {
  viewer {
    avatarUrl(size: $avatarSize)
    organizations(first: $orgFirst) {
      edges {
        node {
          name
        }
      }
    }
  }
}
```

```
{
  “avatarSize”: 250,
  “orgFirst”: 10
}
```

Как это выглядит на клиенте - Mutation

```
mutation LetsCreateNewRepo {
  createRepository(input: {
    name: "graphql-tryout",
    description: "WIP",
    visibility: PUBLIC
  }) {
    repository {
      name
      description
    }
  }
}
```

```
{
  "data": {
    "createRepository": {
      "repository": {
        "name": "graphql-tryout",
        "description": "WIP"
      }
    }
  }
}
```

Как это выглядит на клиенте - остальное

- Connection
- Fragments
- Inline Fragments
- Aliases
- Directives
- Introspection
- Subscriptions

Попробуйте сами

Большинство примеров клиентских запросов можно выполнить самостоятельно на https://developer.github.com/v4/explorer/

apollo-client

- Декларативность, с вас запрос, с apollo - данные
- Кеширование, если данные уже есть, зачем беспокоить сервер?
- Оптимистичное обновление
- Инструменты для TypeScript, Chrome DevTools и VS Code
- Быстро развивается
- Упрощает работу с мутациями и подписками
- Сильное сообщество

relay modern

- Декларативность, с вас запрос, с relay - данные
- Кеширование, если данные уже есть. зачем беспокоить сервер?
- Описание данных находятся рядом с компонентами
- Оптимистичное обновление
- Наличие четких требований к схеме и написании запросов
- Этап сборки (компиляции)
- Упрощает работу с мутациями и подписками

Сравнение клиентов

apollo-client

- большое сообщество
- проще начать
- не зависит от UI фреймворка
- свободная структура кода
- медленнее
- ошибки при исполнении
- маскирование возможно
- обновление хранилища сложнее
- документация лучше
- большой размер

relay modern

- facebook driven
- сложнее начать
- сделан для react
- определенная структура кода
- быстрее
- отлов ошибок при компиляции
- маскирование атрибутов
- обновление хранилища проще
- скудная документация
- меньший размер

Другие клиенты

- urlq
- graphql-request
- micro-graphql-react
- AWS amplify

Больше клиентов https://github.com/chentsulin/awesome-graphql#clients

Сервисы

- AWS AppSync
- Prisma
- Apollo platform

Больше сервисов https://github.com/chentsulin/awesome-graphql#services

Инструменты

- graphiql браузерное IDE для тестирование GraphQL endpoint
- andev-software/graphql-ide и graphiql-app приложение IDE
- insomnia приложение для отправки запросов
- Macroz/GraphQLviz и sheerun/graphqlviz визуализация схемы
- Ghirro/graphql-network Chrome расширение
- 2fd/graphdoc генератор файла документации
- relay-devtools Chrome расширение
- apollo-client-devtools Chrome расширение
- apollographql.vscode-apollo VS Code расширение

Больше инструментов https://github.com/chentsulin/awesome-graphql#tools

Недостатки GraphQL по сравнению с REST

- Нет HTTP кеширование
- Безопасность
- Обработка ошибок, ответ всегда 200
- В ответ только JSON
- Нет работы с файлами

Используйте GraphQL

- Общаетесь с самим собой, свой front со своим back
- Нет возможности делать REST (вместо недо-REST API)
- Короткие проекты
- Постоянно меняющиеся требования
- При простом доступе к данным без серверного кэширования
- Когда developers experience - это фича

Используйте REST

- Долгоиграющие проекты
- Нужно отдавать разный контент (HTML, JSON, XML)
- Сложная аутентификация, авторизация, лимиты на запросы
- API должно отдавать медиа типы
- Когда масштабируемость - это фича

Спасибо

https://iamdi.dev

helloiamdi@gmail.com

@iamdidev Twitter, Github, Instagram

Скачать слайды https://iamdi.dev/slides/graphql.pdf

Ресурсы

https://www.howtographql.com/ GraphQL Tutorials

https://www.youtube.com/channel/UC0pEW_GOrMJ23l8QcrGdKSw Apollo youtube channel

https://www.youtube.com/channel/UCptAHlN1gdwD89tFM3ENb6w Prisma youtube channel

https://graphql.github.io/graphql-spec/ GraphQL specification

https://graphql.org/community/ Доклады, блоги, IRC. соц. сети посвященные GraphQL

https://github.com/APIs-guru/graphql-apis Открытые GraphQL API

https://www.graphqlhub.com/ GraphiQL с доступом на github, twitter, reddit и др.

https://developer.github.com/v4/explorer/ Github API использованый на сладах

Ресурсы (продолжение)

https://github.com/relay-tools Relay Tools репозиторий

https://github.com/sibelius/relay-modern-course курс из слайдов по relay modern

https://foundation.graphql.org/ GraphQL foundation сайт

https://github.com/chentsulin/awesome-graphql список других потрясающих ресурсов о GraphQL