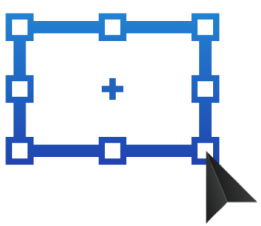
Грабли GraphQL на клиенте

Валерий Бугаков, Evil Martians

Evil Martians



Evil Martians

План

- Почему GraphQL
- GraphQL по частям
- Типичные проблемы

Почему GraphQL

Persons

+name
+birthYear

Films

+title
+releaseDate

Planets

+name
+galaxy

Anakin Skywalker	
Homeworld	Tatooine
Born	41 BBY
Films	
A New Hope	1977
The Empire Strikes Back	1980
Return of the Jedi	1983
Revenge of the Sith	2005

GET — /person/{id}

GET — /planet/{id}

GET — /film/{id}

GET — /person/{id}/card

GET — /person/{id}?include=films,planets

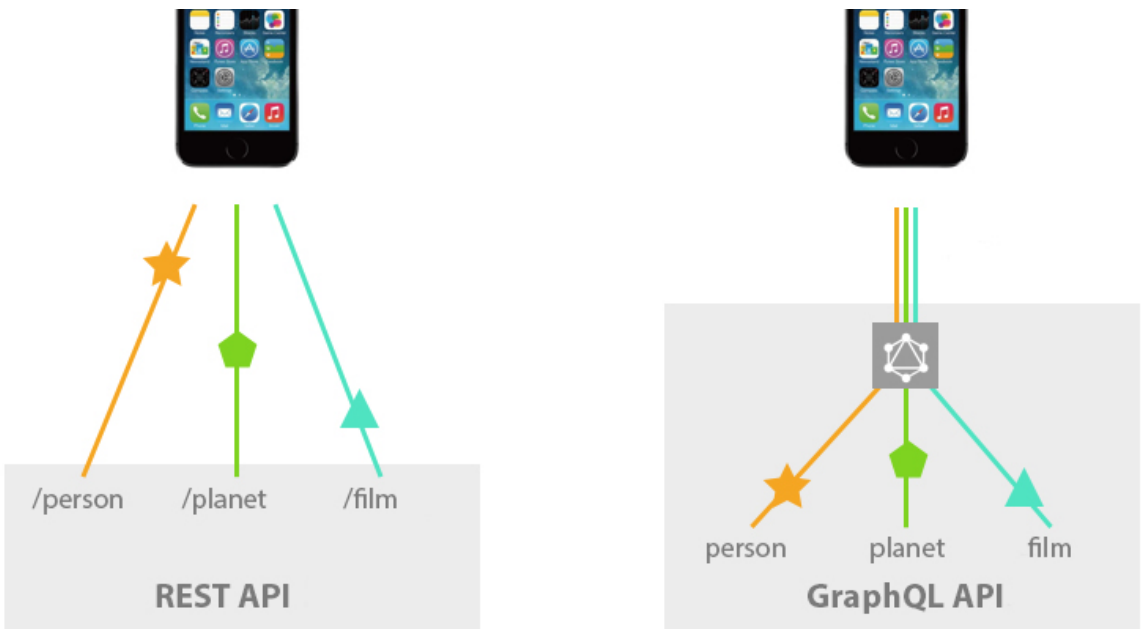
...

Запрос

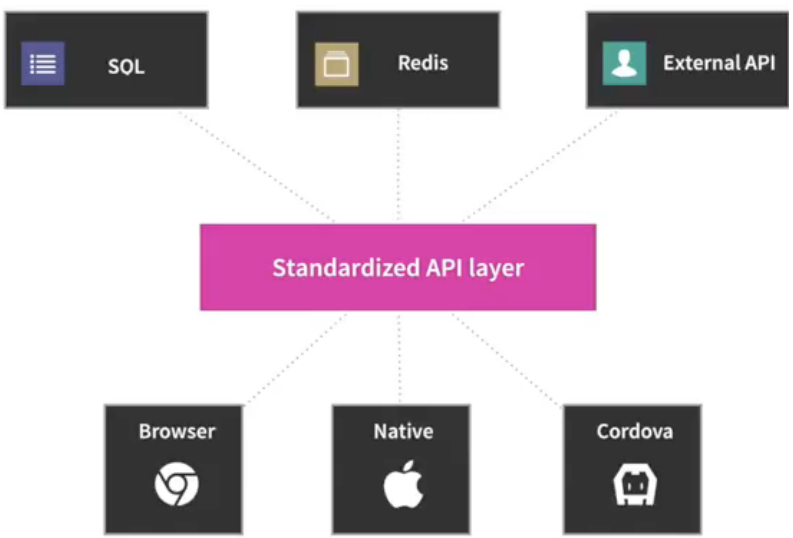
```
person(id: $id) {
  name
  birthYear
  homeworld {
    name
  }
  films {
    title
    releaseDate
  }
}
```

Ответ

```
"person": {
  "name": "Anakin Skywalker",
  "birthYear": "41.9BBY",
  "planet": {
    "name": "Tatooine"
  },
  "films": [
    {
      "title": "A New Hope",
      "year": 1992
    },
    { "title": "The Empire Strikes Back" },
    { "title": "Return of the Jedi" },
    { "title": "Revenge of the Sith" }
  ]
}
```

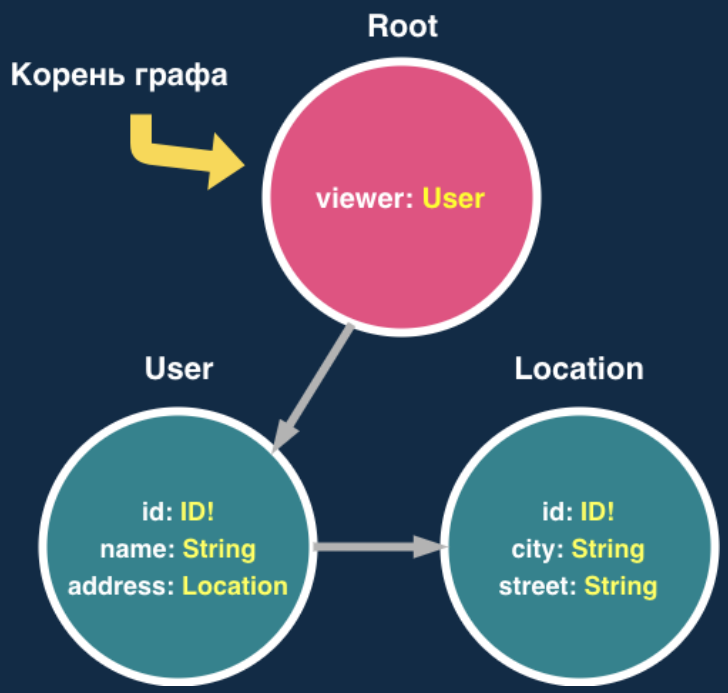

/person
/planet
/film
REST API
person
planet
film
GraphQL API

Thin API layer



GraphQL по частям

- Схема данных
- Резолверы полей
- Язык запросов




```
type Query {
  viewer: User
}

type User {
  id: ID!
  name: String
  address: Location
}

type Location {
  id: ID!
  city: String
  street: String
}
```

На сервере

```
const schema = `
  type Query {
    viewer: User
  }

  type User {
    id: ID!
    name: String
    address: Location
  }

  type Location {
    id: ID!
    city: String
```

На клиенте

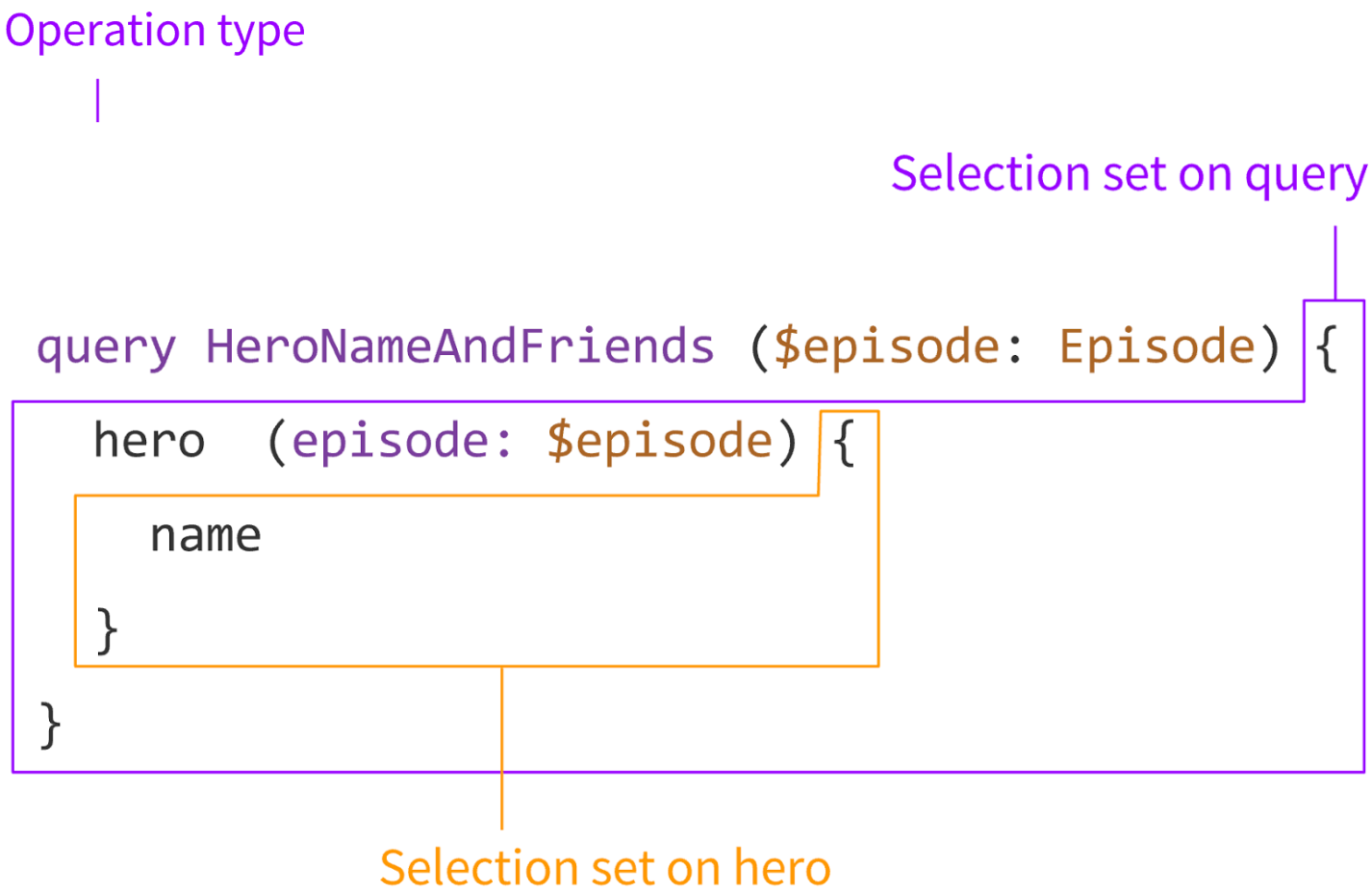
HTTP POST request

```
query {
  viewer {
    name
    address {
      street
    }
  }
}
```

JSON response

```
"data": {
  "viewer": {
    "name": "Anakin Skywalker",
    "address": {
      "street": "Tatooine boulevard"
    }
  }
}
```

Анатомия


Operation type
Selection set on query
query HeroNameAndFriends ($episode: Episode) {
hero (episode: $episode) {
name
}
}
Selection set on hero

Клиентские библиотеки

- Relay
- Apollo
- Graphql-request by Graphcool
- Graphql-client by Shopify
- Locca - not maintained
- ...

Отдельная библиотека для запросов?

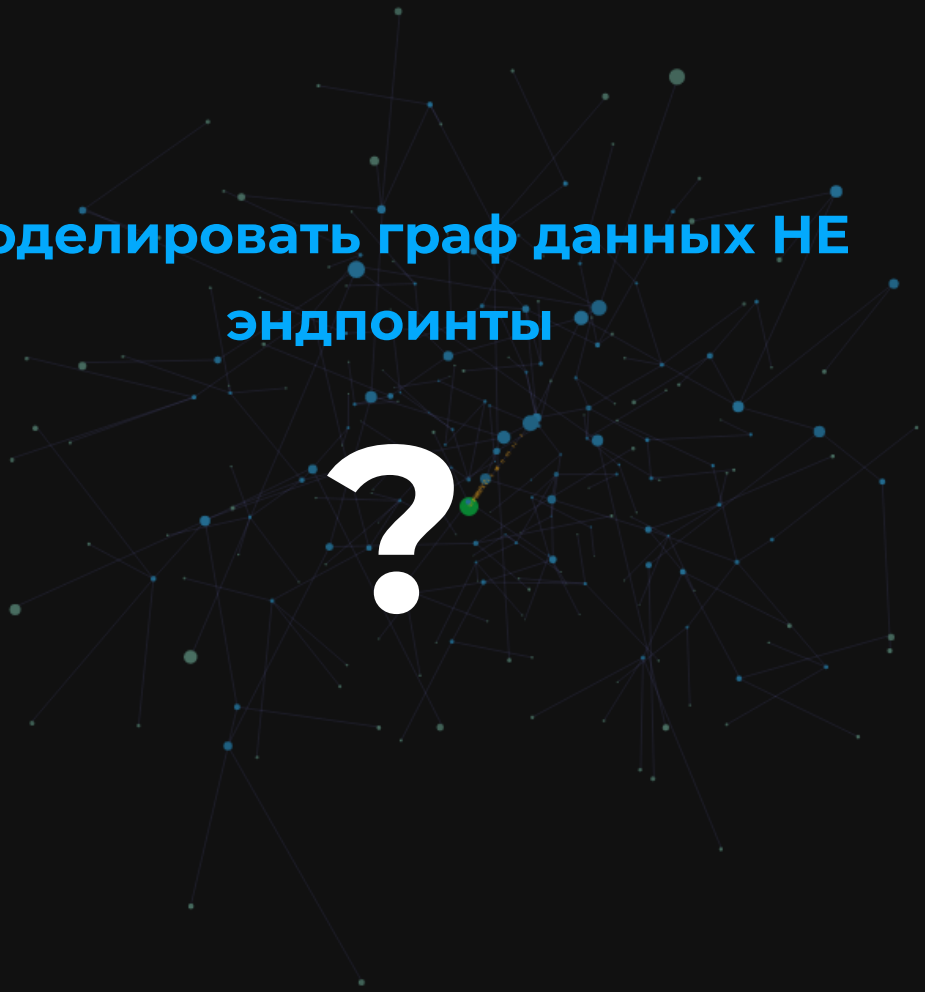
- Подготовить запрос
- Обработать ответ
- Оптимистичные апдейты
- Оффлайн
- Консистентный кэш
- Подписки на обновления
- Удобный интерфейс для паджинации
- Тулзы для оптимизации

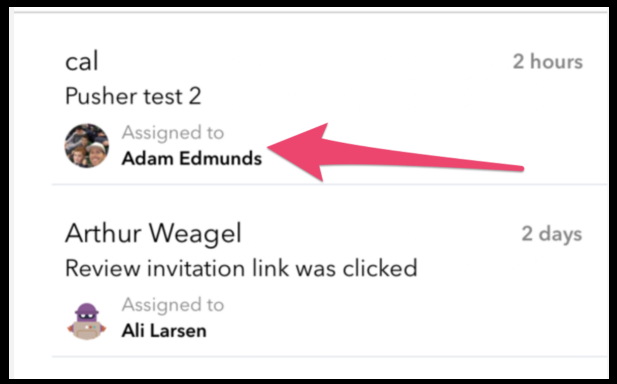
Где лежат грабли?

- Описание графа данных
- Подключение клиента к приложению
- Дебаг ошибок внутри клиента

1. Граф данных

Моделировать граф данных НЕ эндпоинты






```
conversation {
  ...
  moderator {
    date
    firstName
    lastName
    avatarUrl
  }
}
```

```
conversation {
  ...
  moderator {
    date
    user {
      id
      firstName
      lastName
      avatarUrl(size: 80)
    }
  }
}
```

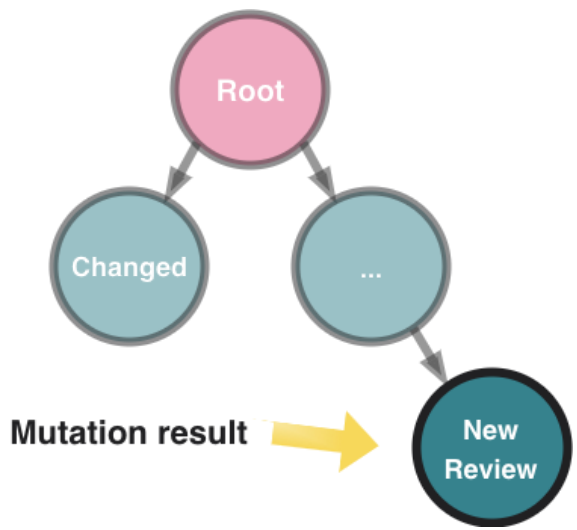
Фичи отдельно от данных

```
query searchUsers($q: String!) {
  searchUsers(query: $q) {
    count
    edges {
      node {
        name
        reviews
      }
    }
  }
}
```

```
query search($q: String!) {
  search(query: $q) {
    count
    edges {
      node {
        ... on User {
          name
          reviews
        }
        ... on Film {
          title
          releaseDate
        }
      }
    }
  }
}
```

Доступность данных

```
mutation createReview($review: String!) {
  createReview(review: $review) {
    id
    stars
    commentary
    root {
      changed
    }
  }
}
```



NEVER FORGET WHO YOU ARE

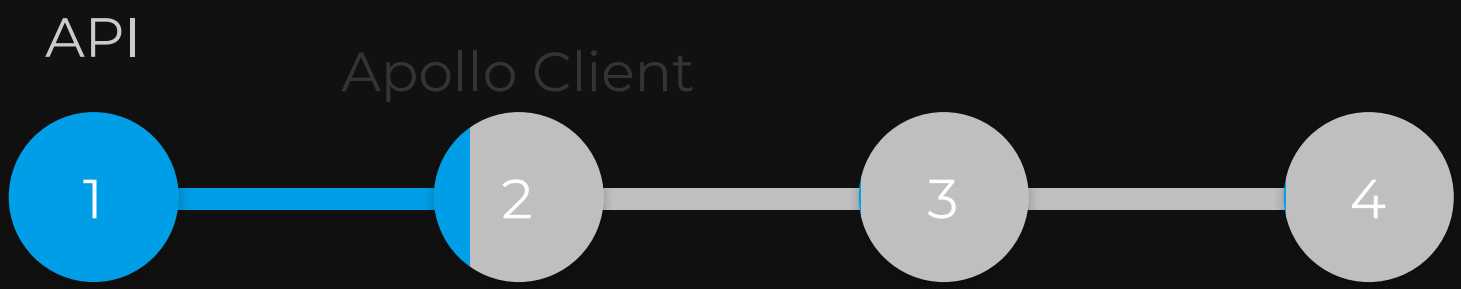
```
# Authentication context
query {
  viewer {
    myFilms {
      ...
    }
  }
}

# Field path context
query {
  university {
    lesson {
      students
    }
  }
}

query {
  day {
    lesson {
      students
    }
  }
}
```

2. Интеграция клиента

Apollo Client setup

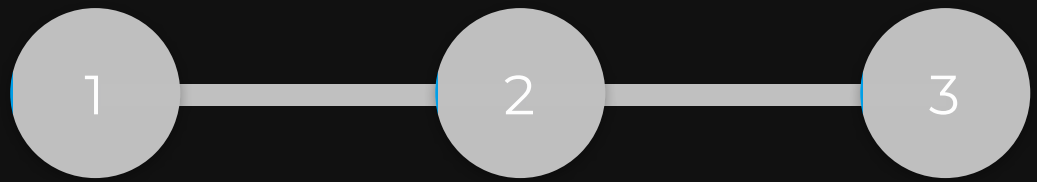
```
1. import { ApolloClient } from 'apollo-client';
2. import { HttpLink } from 'apollo-link-http';
3. import { InMemoryCache } from 'apollo-cache-inmemory';
4.
5. const client = new ApolloClient({
6.   link: new HttpLink({ uri: '/graphql' }),
7.   cache: new InMemoryCache()
8. });
9.
```

Этапы загрузки данных



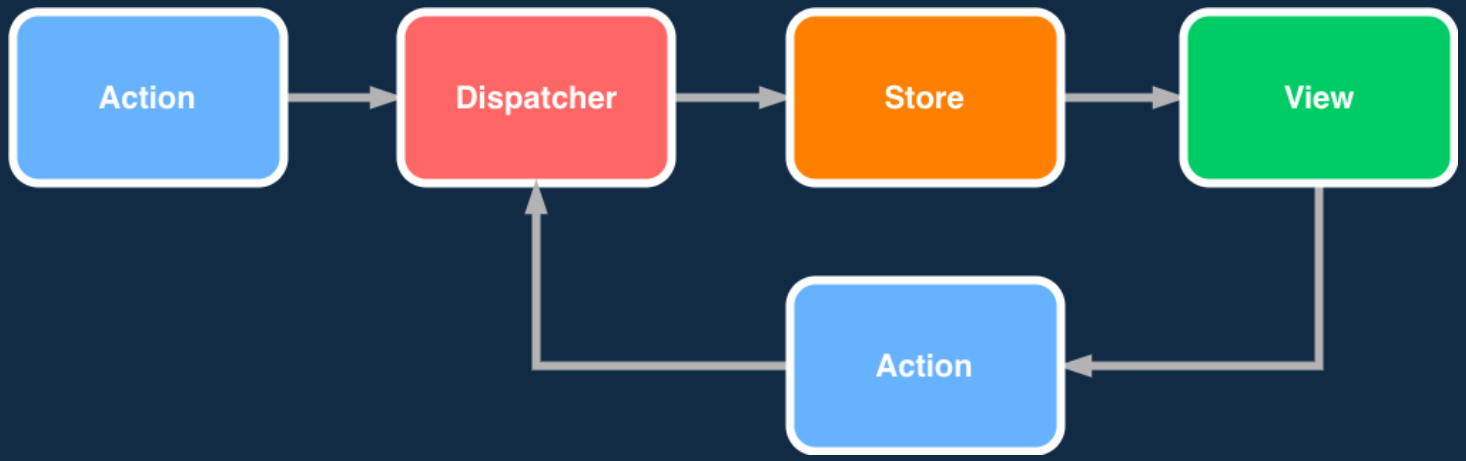

But....but
Why?

Этапы загрузки данных



React Apollo

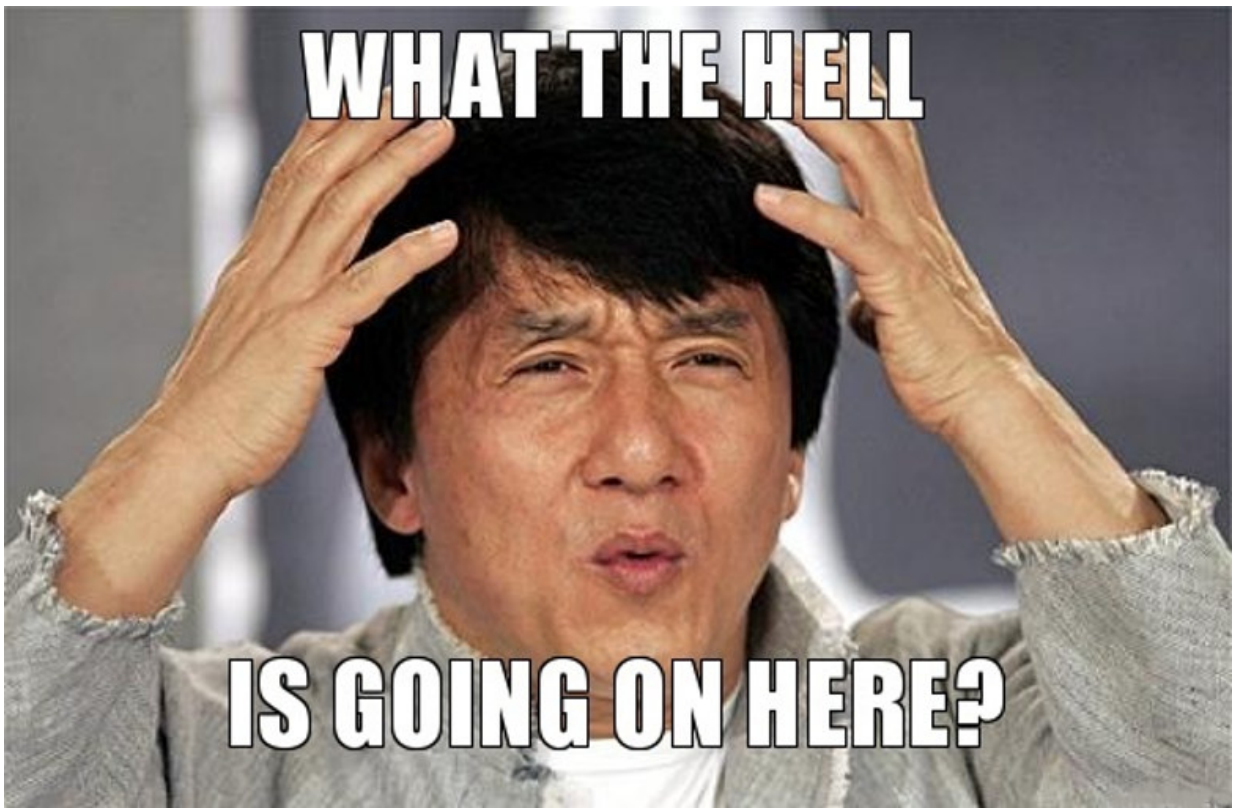
```
1. import { ApolloProvider } from 'react-apollo'
2. import client from './client'
3.
4. ReactDOM.render(
5.   <ApolloProvider client={client}>
6.     <MyAppComponent />
7.   </ApolloProvider>,
8.   document.getElementById('root')
9. )
10.
11. export default graphql(todosQuery, {
12.   options: {
13.     variables: {
14.       status: DONE
```


Action
Dispatcher
Store
View
Action

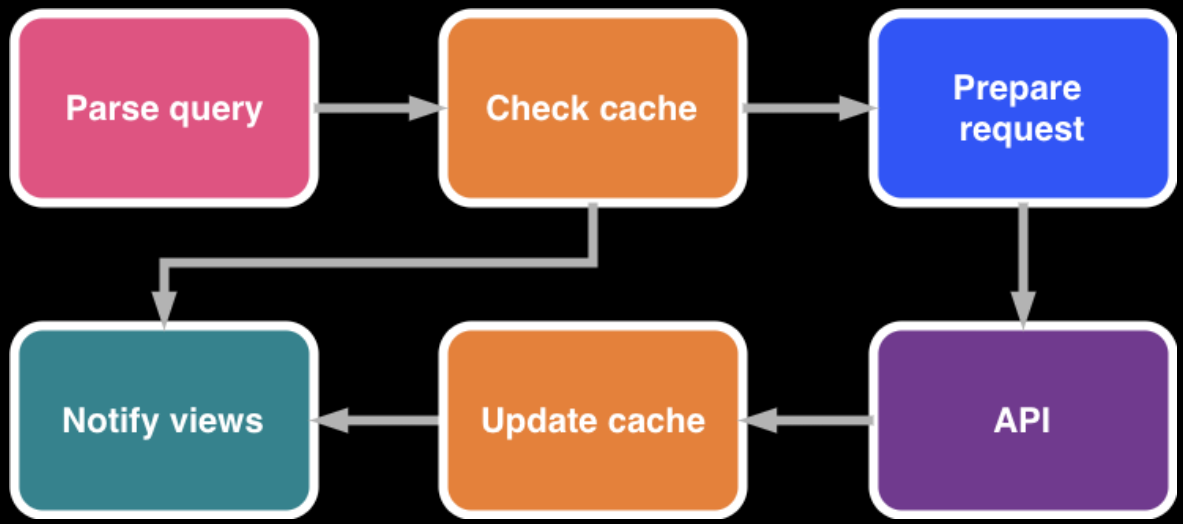
GraphQL everything

```
const MIXED_QUERY = gql`
  query UserData($id: ID!) {
    // from a rest endpoint
    articles @rest(route: '/articles') @type(type: '[Article]') {
      title
      author
    }
    // from the graphql server
    user(id: $id) {
      firstName
    }
    // from local state
    network @client {
      isConnected
    }
  }
`;
```

3. Внутри клиента


WHAT THE HELL
IS GOING ON HERE?

По шагам


Parse query
Check cache
Prepare request
Notify views
Update cache
API

Devtools!

- GraphiQL

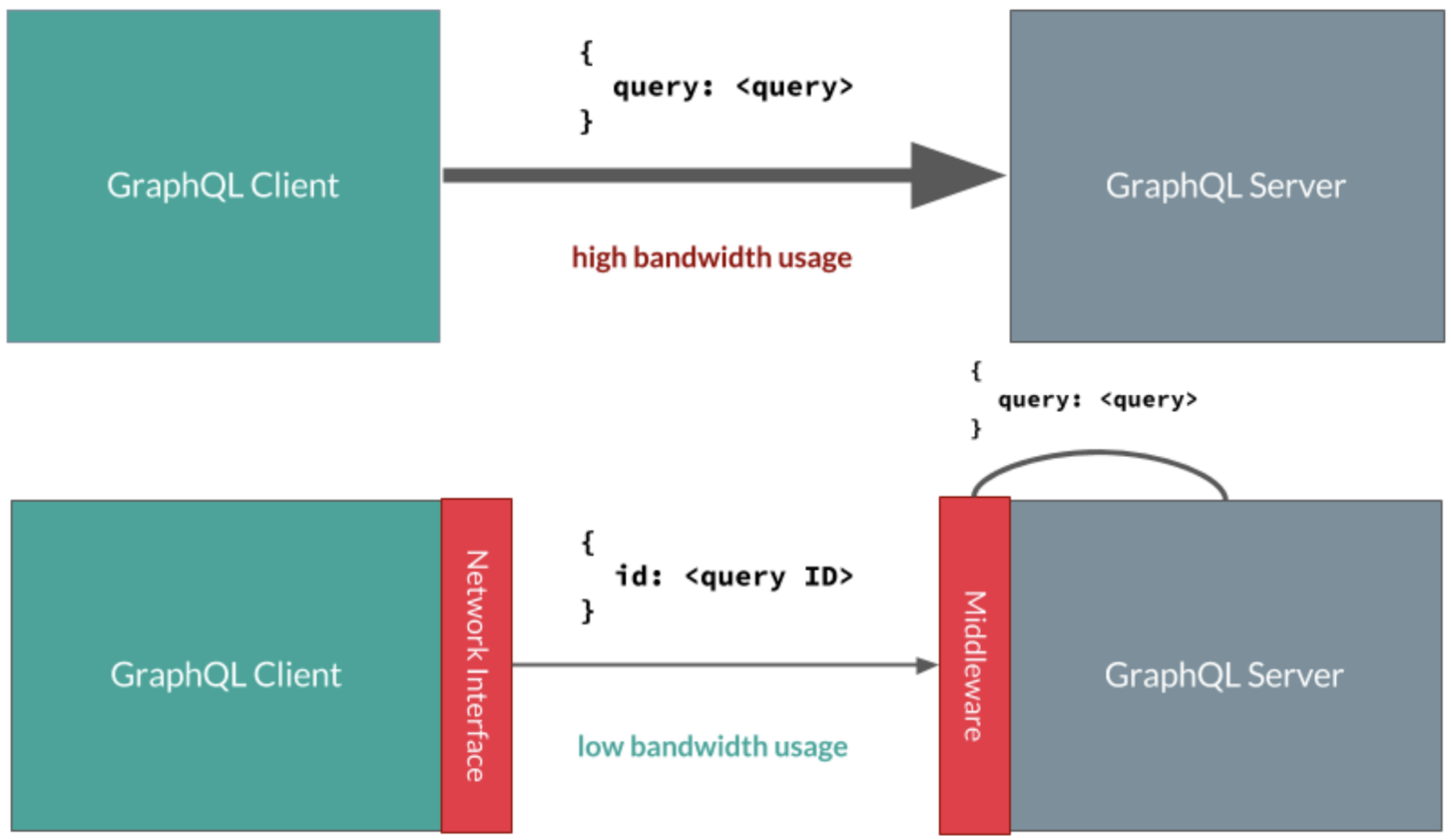
Devtools!

- GraphiQL
- Eslint-plugin-graphql
- Apollo-codegen

Парсили парсили...

__typename

```
union SearchResult = Human | Starship

type Search {
  resultCount: Int!
  results: [SearchResult]!
}

{
  search(query: "an") {
    ... on Human {
      name
      height
    }
    ... on Starship {
```


GraphQL Client
{
query: <query>
}
high bandwidth usage
GraphQL Server
{
query: <query>
}
GraphQL Client
Network Interface
{
id: <query ID>
}
low bandwidth usage
Middleware
GraphQL Server

Persisted queries

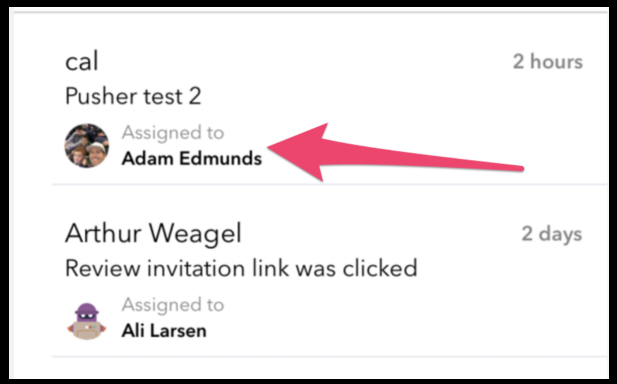
- Добавить build step – **persistgraphql**
- Добавить **middleware** на сервер

Проверка кэша

Сохранение результата

```
// JSON response

"locations": {
  "pageNumber": 1,
  "resources": [
    {
      "id": 7020690,
      "address": "1400 Bungie Ave",
      "__typename": "Location",
    }
    ...
  ]
}
```

```
// key = __typename + objectId

"store": {
  "$ROOT_QUERY:locations": {
    "pageNumber": 1,
    "resources": [
      {
        "type": "id",
        "id": "Location:7020690"
      },
      ...
    ],
  },
  "$Location:7020690": {
    "address": "1400 Bungie Ave",
  },
  ...
}
```




```
conversation {
  ...
  moderator {
    date
    firstName
    lastName
    avatarUrl
  }
}
```

```
conversation {
  ...
  moderator {
    date
    user {
      id
      firstName
      lastName
      avatarUrl(size: 80)
    }
  }
}
```

Если мы не контролируем данные

```
const cache = new InMemoryCache({
  dataIdFromObject: object => {
    switch (object.__typename) {
      // Custom ids
      case 'bar': return object.blah
      case 'foo': return object.foo + object.bar

      // Default fallback
      default: return object.id || object._id
    }
  }
})
```

Если мы не контролируем данные

```
const options = {
  update: (cache, { data: { todo } }) => {
    const data = cache.readQuery({ todoQuery })

    data.todos.push(todo)

    cache.writeQuery({ todoQuery, data })
  },
}
```

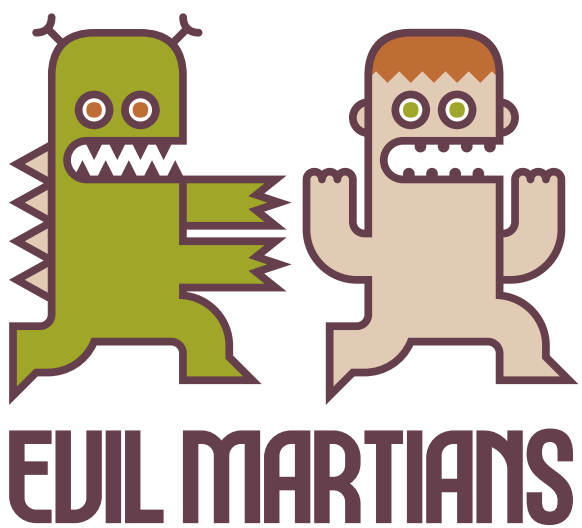
Итого

- Моделируйте граф данных
- Используйте все фичи клиентов
- Следите за кэшем
- ❤️ GraphQL

Вопросы?

valerybugakov

valerybugakov



evilmartians

http://evl.ms