GraphQL overview

Introduction to GraphQL

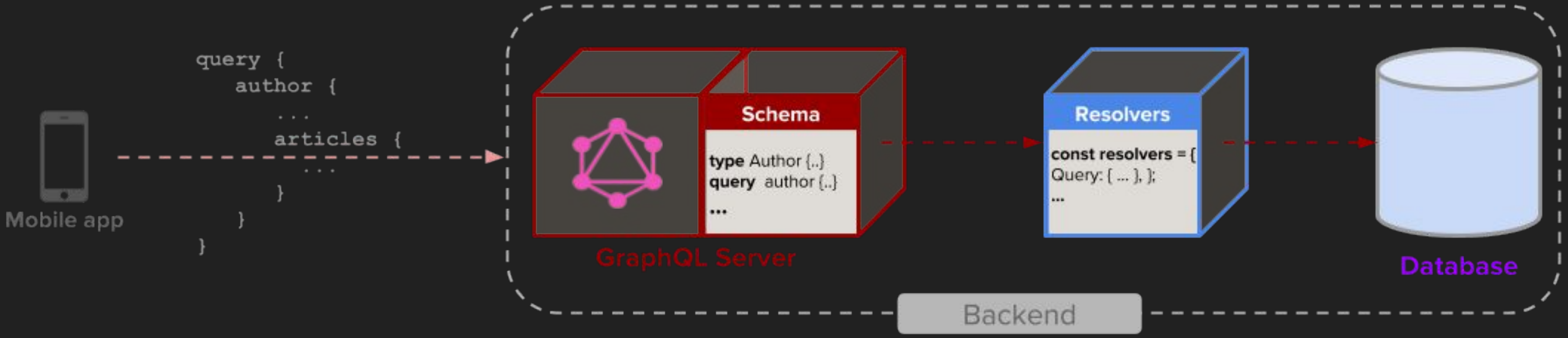
Index

- What’s GraphQL?
  - Basis
  - Uses
  - The foundation
- It worth it?
- What does it mean for my project?
- How could I implement it?
  - Backend
  - Frontend
- Workshop - Apollo server + Angular

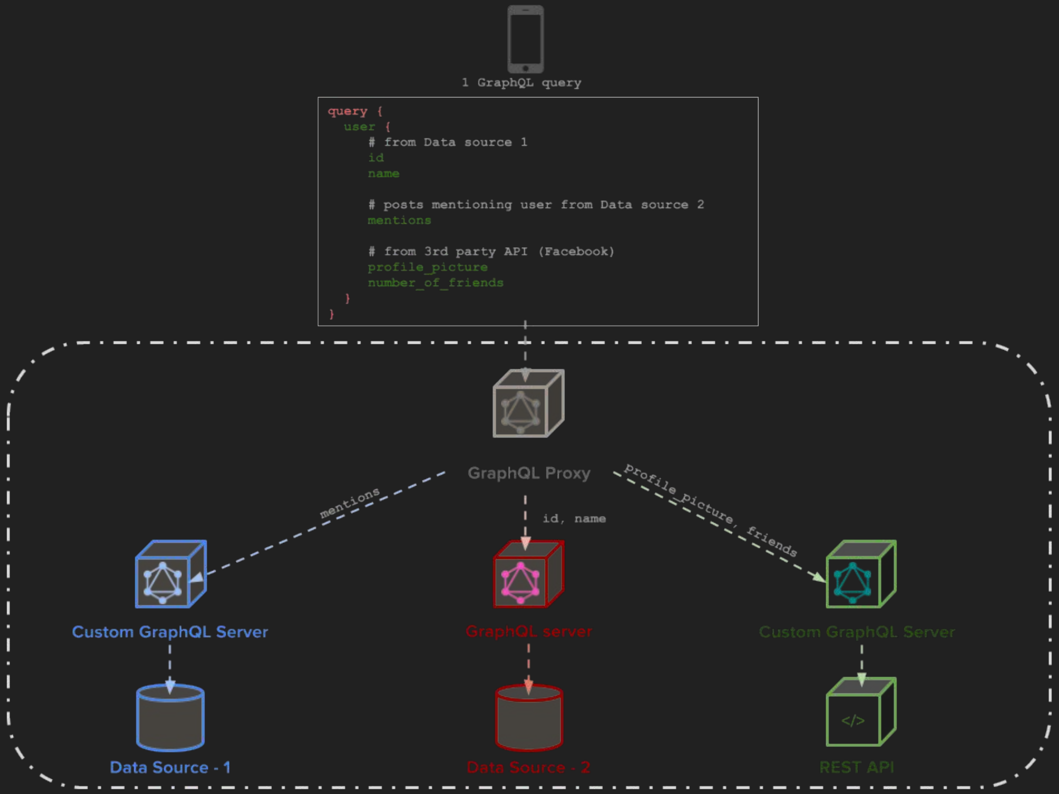
What’s GraphQL?

- Query and manipulation Language!
- Developed by Facebook in 2012
- Publicly released in 2015
- Under Linux Foundation since November 2018
- Basic operations: type, query, mutation, subscription
- Performance improvements
- Aim of decoupling frontend and backend
- Delegate business logic to frontend

Basis


query {
author {
...
articles {
...
}
}
}
Mobile app
Schema
type Author {..}
query author {..}
...
GraphQL Server
Resolvers
const resolvers = {
Query: { ... }, };
...
Database
Backend

Basis



```
query {
  user {
    # from Data source 1
    id
    name

    # posts mentioning user from Data source 2
    mentions

    # from 3rd party API (Facebook)
    profile_picture
    number_of_friends
  }
}
```

Basis

- Schema Definition SDL

```
type Person {
  name: String!
  age: Int!
  posts: [Post!]!
}
type Post {
  title: String!
  author: Person!
}
```

- Queries

```
{
  allPersons(last: 2) {
    name
  }
}
```

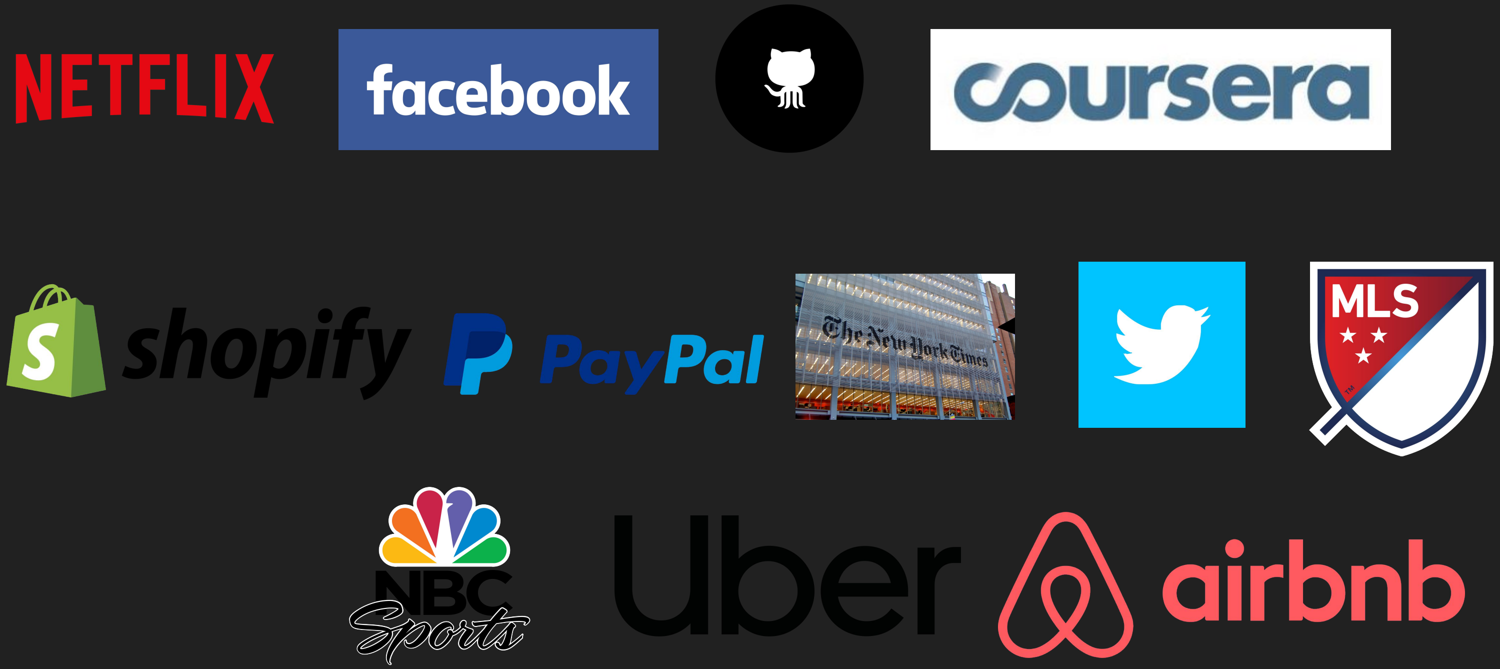
- Mutations

```
mutation {
  createPerson(name: "Bob", age: 36) {
    name
    age
  }
}
```

- Subscriptions

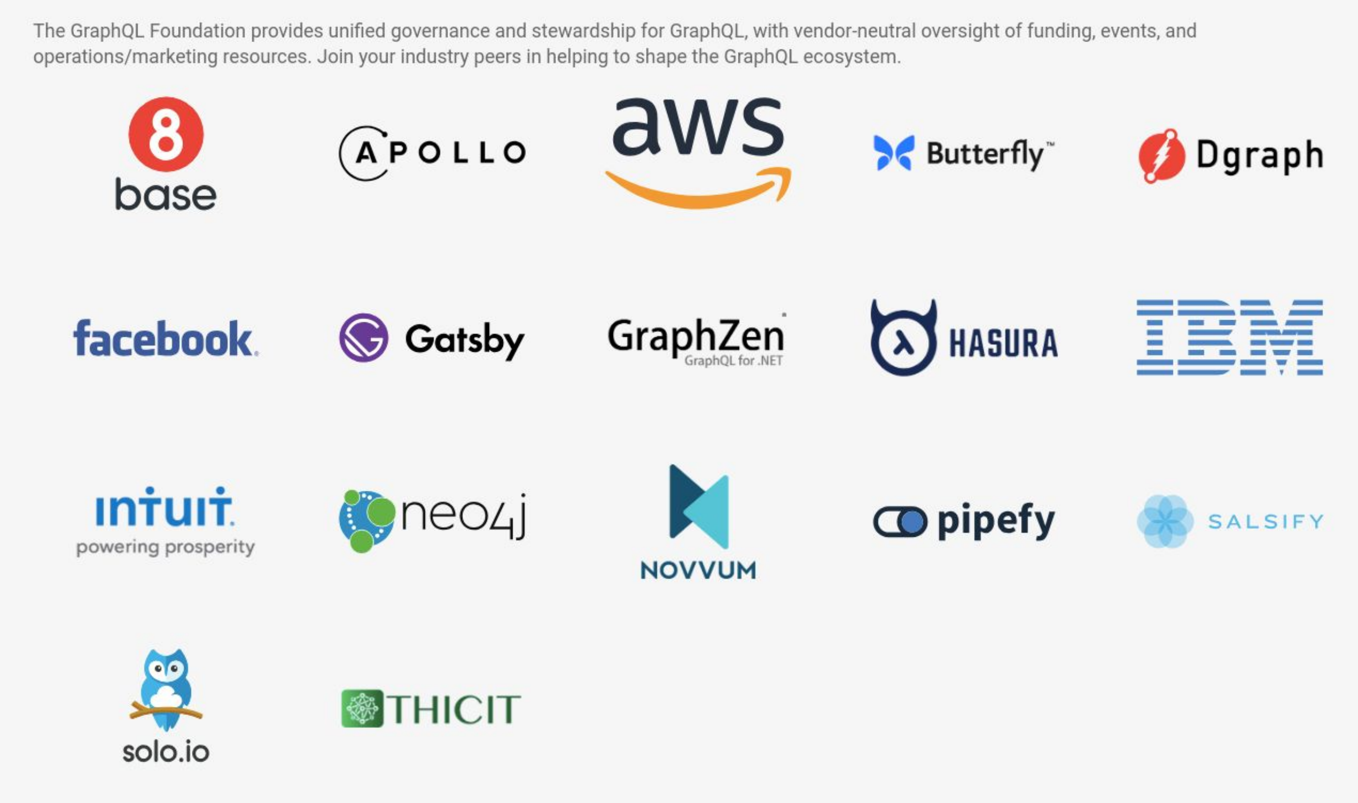
```
subscription {
  newPerson {
    name
    age
  }
}
```

Uses



https://graphql.org/users/

Foundation



The GraphQL Foundation provides unified governance and stewardship for GraphQL, with vendor-neutral oversight of funding, events, and operations/marketing resources. Join your industry peers in helping to shape the GraphQL ecosystem.

It worth it?

Pros

- Multiple clients for same API
- Data fetching on demand
- Project lifecycle
- Isolation
- Client oriented
- Caching

Cons

- Query performance
- Learning curve
- Building queries
- Client oriented
- Caching

What it means to my project?

- GraphQL SDL is not data modeling
- Frontend team will receive tons of business logic
- Learning curve
- Ready to grow
- Select stack carefully
- Full stack developers are unicorns

How could I implement it?

Backend:

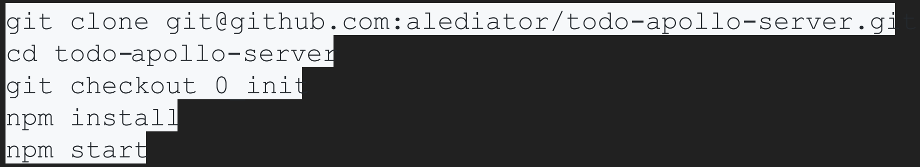
- Apollo Server
- AppSync
- Hasura
- NodeJS (Express)
- API platform
- Prisma

Frontend:

- Apollo Client
- AppSync
- Prisma

Getting started with Apollo server

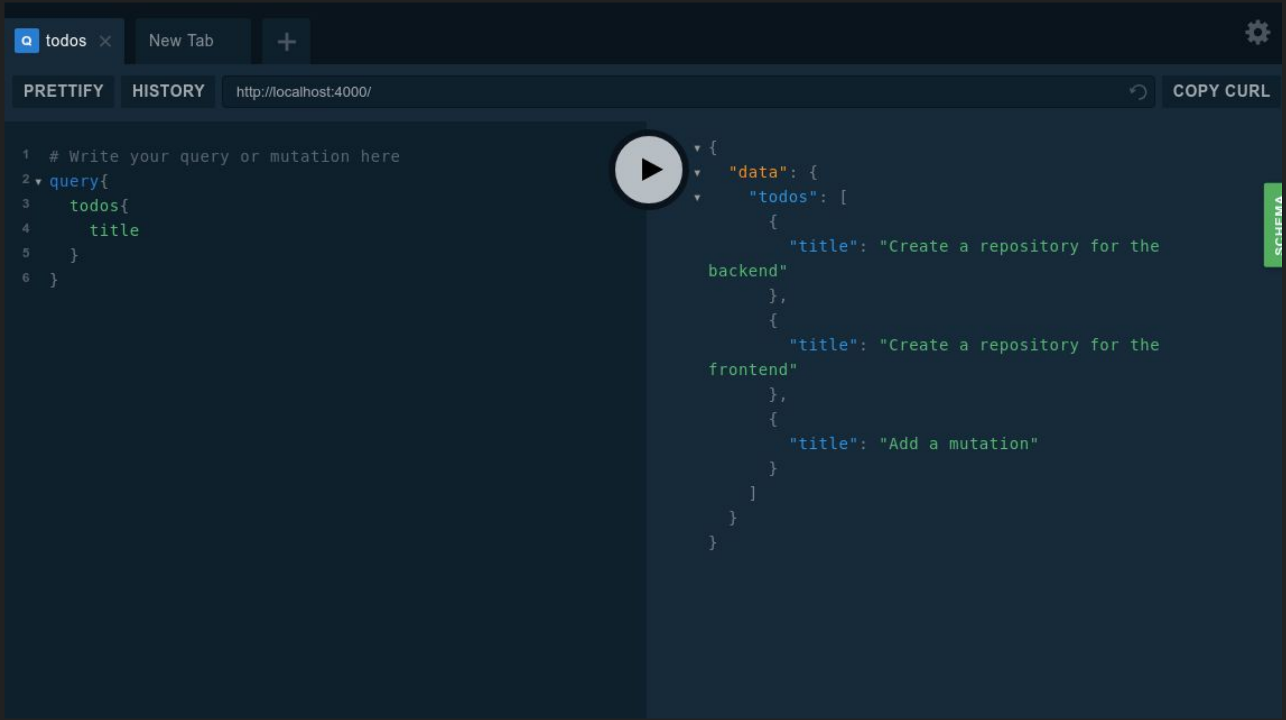
https://github.com/alediator/todo-apollo-server
https://todo-apollo-server.herokuapp.com/graphql
https://github.com/bsh314/todo-apollo-client
http://todo-apollo-client.herokuapp.com/



```
git clone git@github.com:alediator/todo-apollo-server.git
cd todo-apollo-server
git checkout 0_init
npm install
npm start
```

Queries

https://github.com/alediator/todo-apollo-server/tree/0_init

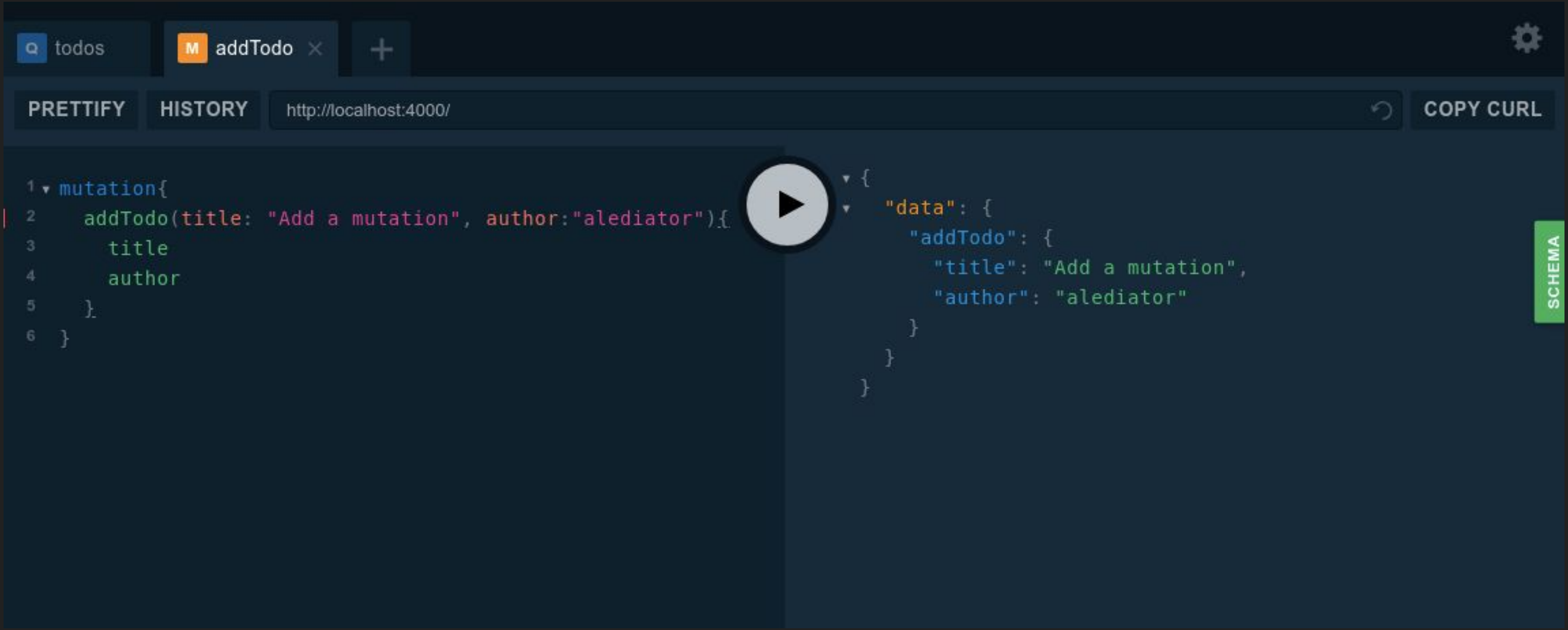


```
# Write your query or mutation here
query{
  todos{
    title
  }
}
```

```
{
  "data": {
    "todos": [
      {
        "title": "Create a repository for the
backend"
      },
      {
        "title": "Create a repository for the
frontend"
      },
      {
        "title": "Add a mutation"
      }
    ]
  }
}
```

Mutations

https://github.com/alediator/todo-apollo-server/tree/2_mutation



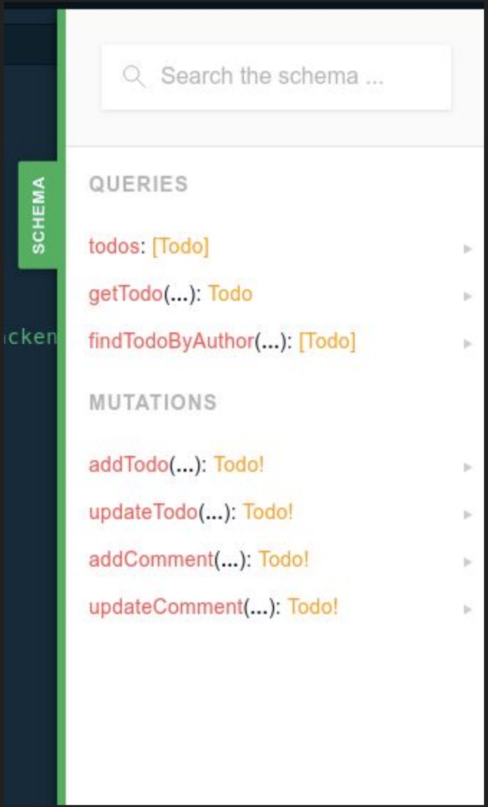
```
mutation{
  addTodo(title: "Add a mutation", author:"alediator"){
    title
    author
  }
}
```

```
{
  "data": {
    "addTodo": {
      "title": "Add a mutation",
      "author": "alediator"
    }
  }
}
```

Subscriptions - preface

https://github.com/alediator/todo-apollo-server/tree/4_comments

- Add comments to a TODO
- Update TODO
- Get a TODO by id
- Get comments by author
- Update a comment



QUERIES

todos: [Todo]

getTodo(...): Todo

findTodoByAuthor(...): [Todo]

MUTATIONS

addTodo(...): Todo!

updateTodo(...): Todo!

addComment(...): Todo!

updateComment(...): Todo!

Subscriptions

https://github.com/alediator/todo-apollo-server/tree/5_subscriptions

- Follow instructions in https://github.com/apollographql/graphql-subscriptions
- You need to use async
- Remember that isn’t magic
  - Offline
  - Idle time
- It is based on sockets

Questions?

Reference

- https://www.howtographql.com/basics/2-core-concepts/
- https://graphql.org/users/
- https://www.graphql.com/case-studies/
- https://stackshare.io/posts/companies-using-graphql-in-production-2018
- https://medium.com/netflix-techblog/our-learnings-from-adopting-graphql-f099de39ae5f
- https://about.sourcegraph.com/graphql/graphql-at-twitte
- https://blog.logrocket.com/5-reasons-you-shouldnt-be-using-graphql-61c7846e7ed3
- https://labs.mlssoccer.com/implementing-graphql-at-major-league-soccer-ff0a002b20ca
- https://medium.com/@mbardauskas/graphql-is-it-worth-the-trouble-789a20c375ff
- https://www.youtube.com/watch?v=OQTnXNCDywA
- https://scotch.io/tutorials/graphql-the-good-and-the-bad
- https://artsy.github.io/blog/2018/05/08/is-graphql-the-future/
- https://gql.foundation/members/
- https://apievangelist.com/2018/04/16/graphql-thoughts-after-almost-two-years/
- https://dev.to/sadarshannaiynar/graphql-or-rest-what-should-i-use-38mj
- https://www.programmableweb.com/news/top-5-things-to-remember-when-adding-graphql-backend/analysis/2018/10/01
- https://medium.com/@mohamedaymen.ourabi11/creating-a-simple-crud-app-with-nodejs-graphql-and-mongodb-docker-eeb22d44925b
- https://api-platform.com/
- https://hackernoon.com/graphql-apis-for-backend-devs-bba579e72eba