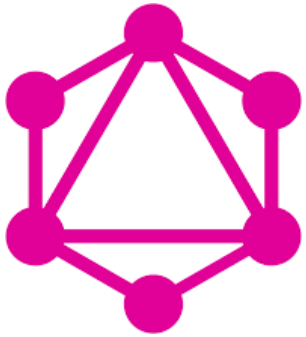


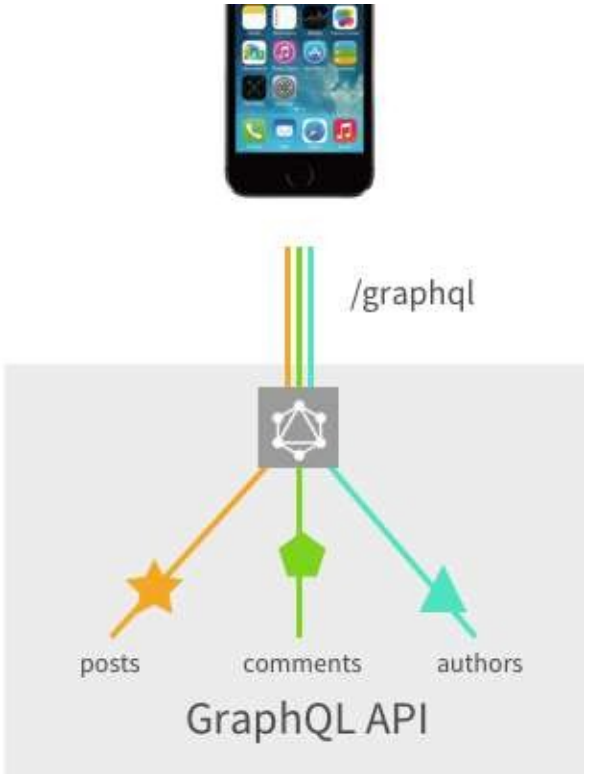
Introducción a GraphQL

¿Qué es GraphQL?

- Query language creado por Facebook en 2012
- Es una interfaz entre el cliente y el servidor para solicitar y manipular datos
- El cliente define la respuesta que quiere obtener del servidor

Principal diferencia con REST

GET
POST
PUT
DELETE
...



REST es bueno :)

Ejemplo: `GET /users/1/friends`

Ventajas:

- Muy descriptivo, estamos solicitando los amigos del usuario con id = 1
- URL path único para un recurso en particular
- Cacheable

Pero tiene limitaciones :(

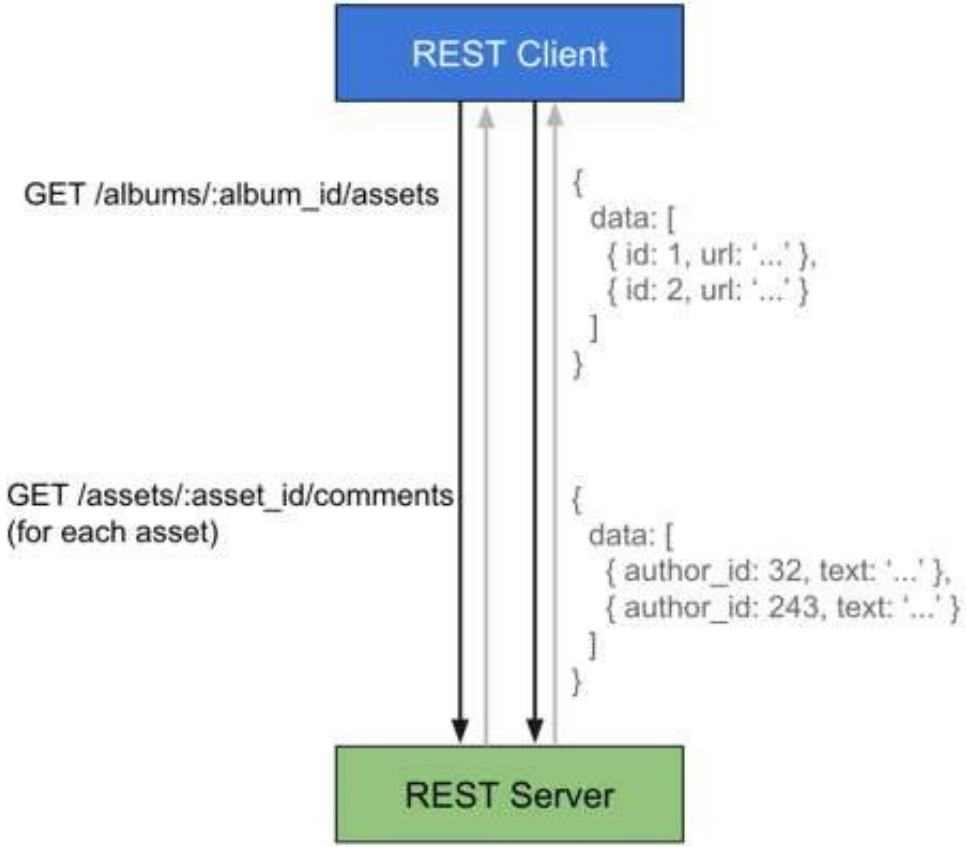
Es común ver cosas así:

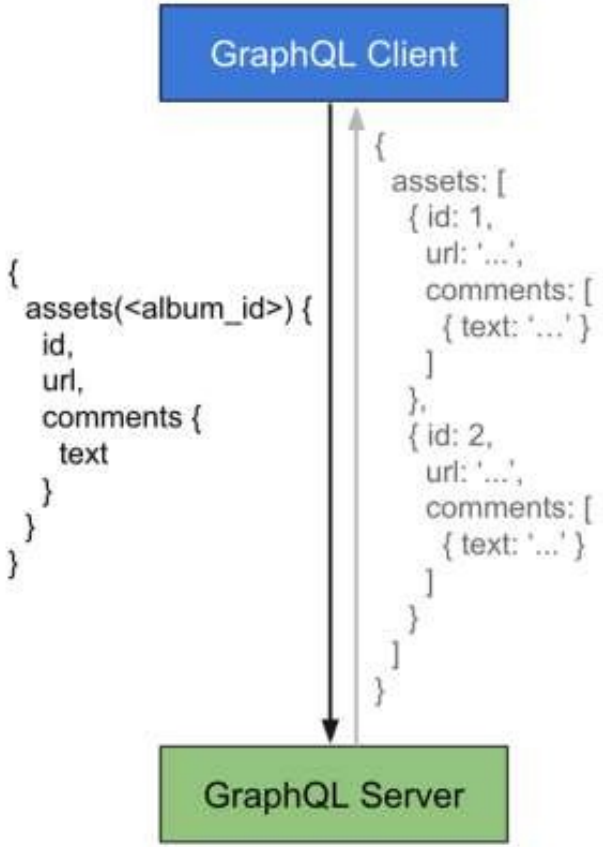
```
GET /users/1/friends/1/dogs/1?include=user.name,dog.age
```

Desventajas:

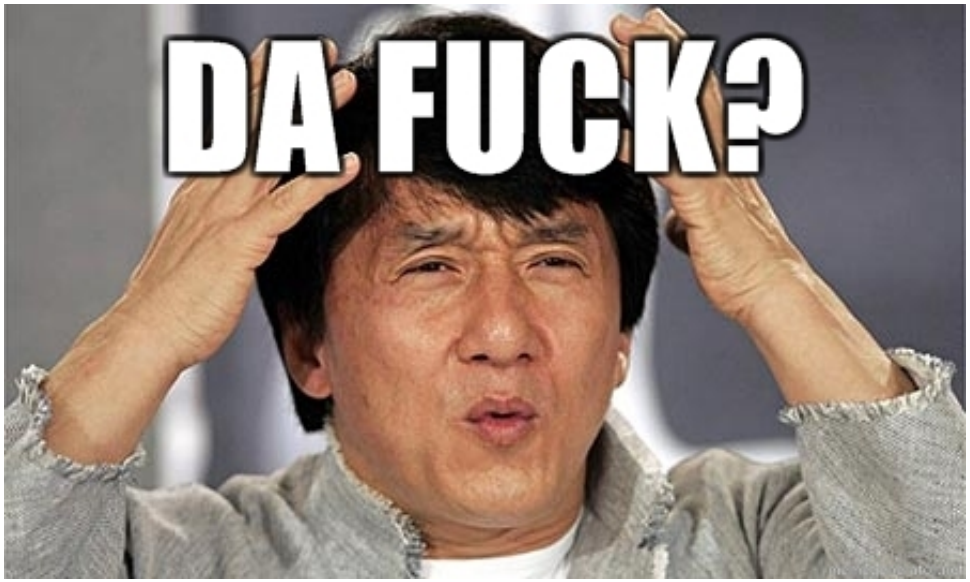
- Complicado de incluir datos en la respuesta
- Complicado de excluir datos de la respuesta
- Multiples request para obtener la información

GraphQL al rescate :)





Con graphQL puedes hacer esto :)




```
movies {
  id
  actors {
    name
    movies {
        id
        actors {
          name
          movies {
              ....
          }
      }
  }
}
```

Schemas (Describir los datos que manejaremos)

- Describir nuestros modelos
- Relaciones entre los modelos
- Exponer sus campos / datos

Types

- Int
- Float
- String
- Boolean
- Object

- Int!
- Float!
- String!
- Boolean!
- Object!

- [Int]
- [Float]
- [String]
- [Boolean]
- [Object]

- [Int]!
- [Float]!
- [String]!
- [Boolean]!
- [Object]!

Queries y Mutations

Queries = Leer datos del servidor (normalmente extraidos de una db)

Mutations = Crear / modificar / borrar datos en el servidor

```
query {
  getMovies {
    name
    year
    score
    actors {
      name
      age
      country
    }
    comments {
      user
      commentary
      timestamp
    }
  }
}
```

```
mutation {
  addClient(
    integration_name: "test",
    grant_type: "client_credentials",
    permissions: ["read"],
  ){
    id
    integration_url
    integration_name
    client_secret
    redirect_uri
    user_id
    grant_type
    permissions
  }
}
```

Ejemplo: Película, actores y comentarios

```
{
  movie {
    name
    score
    year
    actors {
      name
      age
      country
    }
    comments {
      user
      timestamp
      commentary
    }
  }
}
```

Películas

```
import actorQL from './actorQL';
import commentQL from './commentQL';

const movieQL = new GraphQLObjectType({
  name: 'movieQL',
  description: 'This is a movie QL object',
  fields: () => {…
});

export default movieQL;
```

```
fields: () => {
  return {
    name: {
      type: GraphQLString,
      resolve(movie) {
        return movie.name;
      }
    },
    score: {…
    year: {…
    actors: {
      type: new GraphQLList(actorQL),
      resolve(movie) {
        return movie.actors;
      }
    },
    comments: {
      type: new GraphQLList(commentQL),
      resolve(movie) {
        return movie.comments;
      }
    },
  }
}
```

Actores

```
const actorQL = new GraphQLObjectType({
  name: 'actorQL',
  description: 'This is an actor QL object',
  fields: () => {
    return {
      name: {
        type: GraphQLString,
        resolve(actor) {
          return actor.name;
        }
      },
      age: {
        type: GraphQLInt,
        resolve(actor) {
          return actor.age;
        }
      },
      country: {
        type: GraphQLString,
        resolve(actor) {
          return actor.country;
        }
      },
    }
  }
});

export default actorQL;
```

Comentarios

```
const commentQL = new GraphQLObjectType({
  name: 'commentQL',
  description: 'This is a comment QL object',
  fields: () => {
    return {
      user: {
        type: GraphQLString,
        resolve(comment) {
          return comment.name;
        }
      },
      timestamp: {
        type: GraphQLInt,
        resolve(comment) {
          return comment.timestamp;
        }
      },
      commentary: {
        type: GraphQLString,
        resolve(comment) {
          return comment.commentary;
        }
      },
    }
  }
});

export default commentQL;
```

Query

```
import movieQL from './movieQL';
import { movies } from '../data';

const query = new GraphQLObjectType({
    name: 'Query',
    description: 'This is the root Query',
    fields: () => {
      return Object.assign({
        getMovies: {
          type: new GraphQLList(movieQL),
          args: {},
          resolve(root, args, request) {
            // do some db queries
            return movies;
          }
        },
      });
    },
});

export default query;
```

```
query {
  getMovies {
    name
    year
    score
    actors {
      name
      age
      country
    }
    comments {
      user
      commentary
      timestamp
    }
  }
}
```

Mutations

```
const mutation = new GraphQLObjectType({
  name: 'Mutation',
  description: 'This is the root Mutation',
  fields: () => {
    return Object.assign({
      createMovie: {
        type: movieQL,
        args: {
          name: {…
          year: {…
          score: {…
          actors: {
            type: new GraphQLList(actorInputQL),
          },
          comments: {
            type: new GraphQLList(commentInputQL),
          },
        },
        resolve(root, args, request) {
          // do something here
          return args;
        },
      },
    });
  },
});

export default mutation;
```

```
mutation {
  createMovie(
    name: "coco",
    year: 2017,
    score: 9.8,
    actors: [{ name: "Benjamin" }]
    comments: [ { user: "alevsk", commentary: "Love it" } ]
  ) {
    name
    year
    score
    actors {
      name
    }
    comments {
      user
      commentary
    }
  }
}
```

GraphQL Schema

```
import { GraphQLSchema } from 'graphql';
import queryQL from './queryQL';
import mutationQL from './mutationQL';

const schemaQL = new GraphQLSchema({
  query: queryQL,
  mutation: mutationQL,
});

export default schemaQL;
```

Poniendo todo junto
y jugando con graphiQL

+


```
import express from 'express';
import GraphHTTP from 'express-graphql';
// import auth from './auth'
import schemaQL from './graphql/schemaQL';

var app = express();

app.use('/graphiql', GraphHTTP({
    schema: schemaQL,
    pretty: true,
    graphiql: true
}));

app.use('/graphql', GraphHTTP({
    schema: schemaQL
}));

app.get('/', function (req, res, next) {
  const reponse = {
    message: 'hello world',
  };
  return res.json(reponse);
});

module.exports = app;
```

Dudas preguntas

Demo: https://github.com/jamayoral/graphql-demo

https://github.com/jamayoral