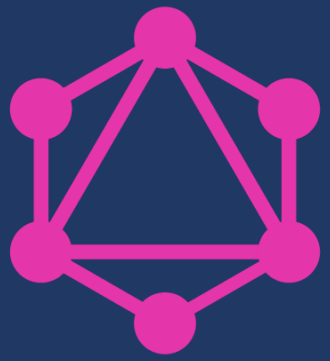


GraphQL – forget (the) REST?

Christian Schwendtner

API CONFERENCE

https://www.pexels.com/photo/man-person-people-emotions-1990/

https://www.pexels.com/photo/man-person-people-emotions-1990/

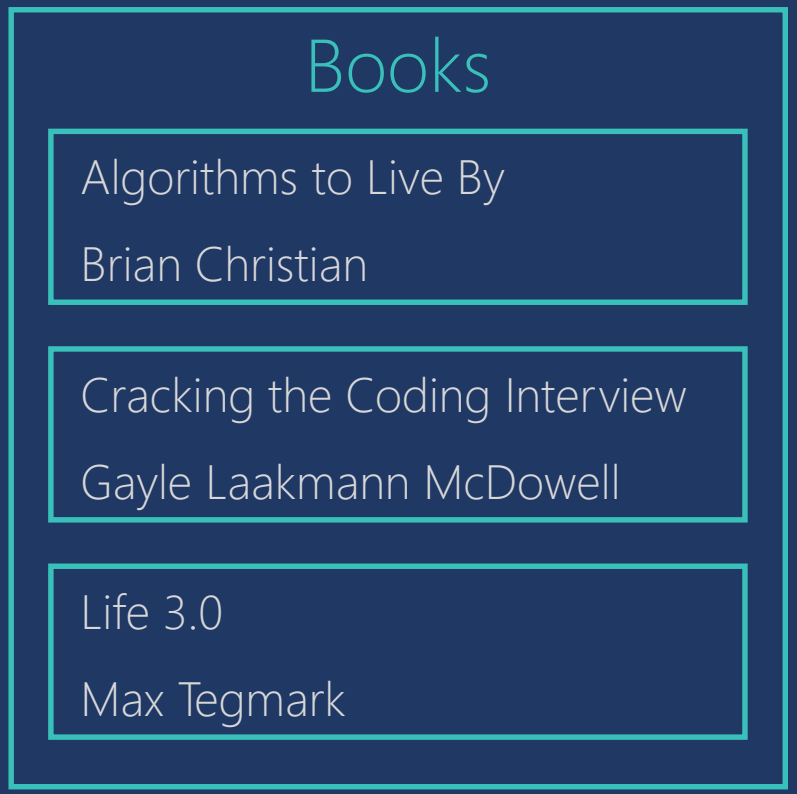
REST – Feelings

I'm lovin' it

Well, …

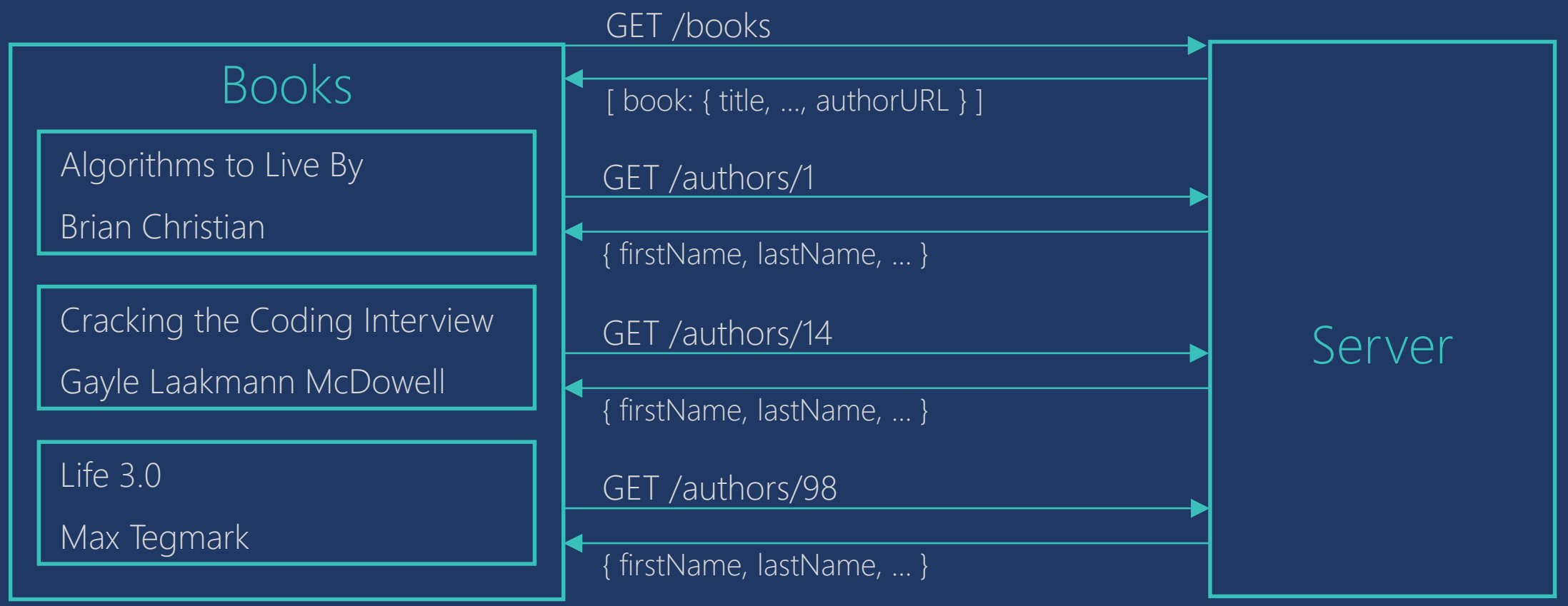
What else?

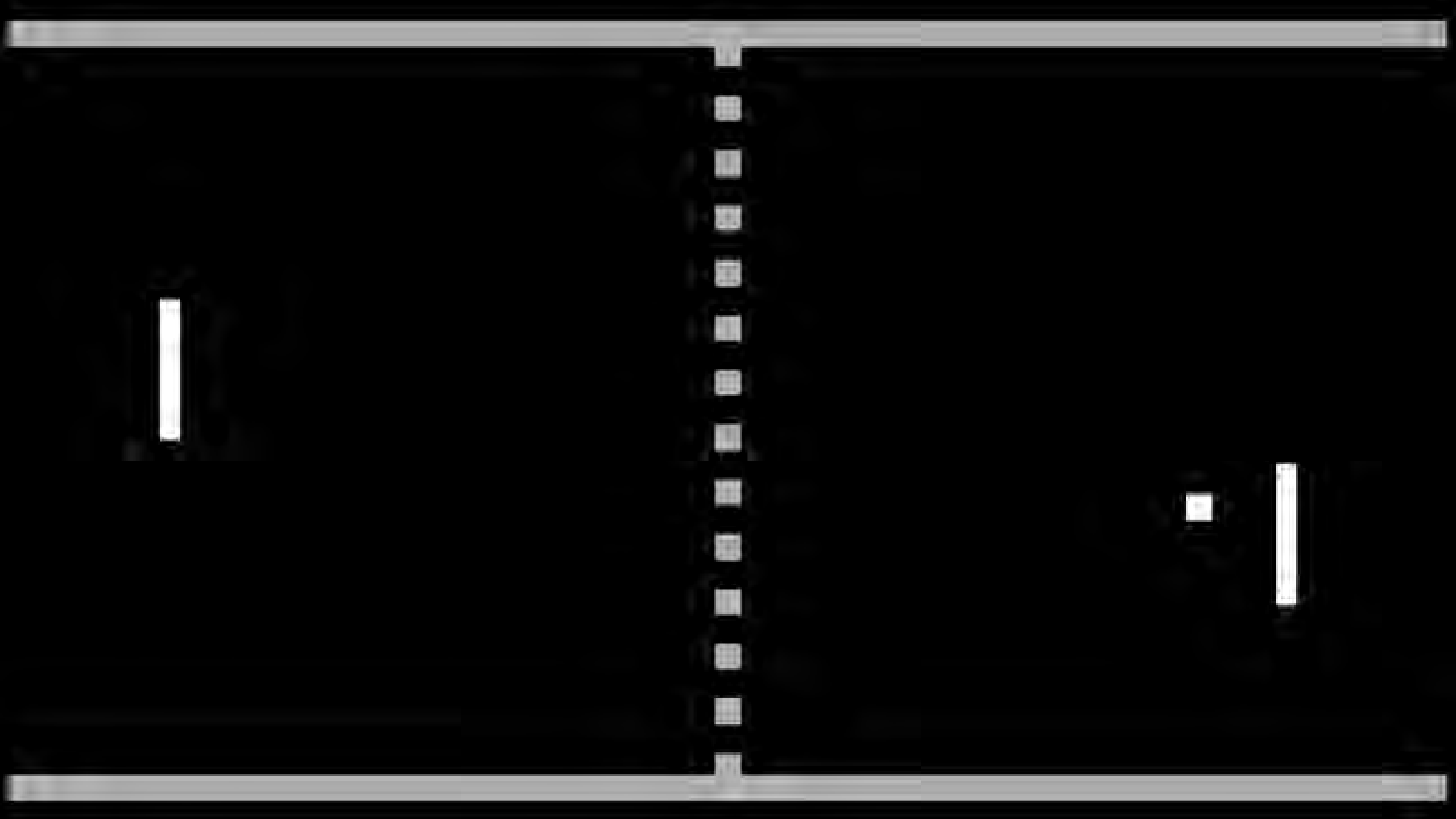
The BookStore aka “Amazon (very) light”



Challenges

Interaction





https://www.pexels.com/photo/adult-alcohol-blur-celebrate-290316/

https://www.pexels.com/photo/time-lapse-photography-of-water-bobbling-beside-lemon-fruit-161425/

“Custom endpoints”?

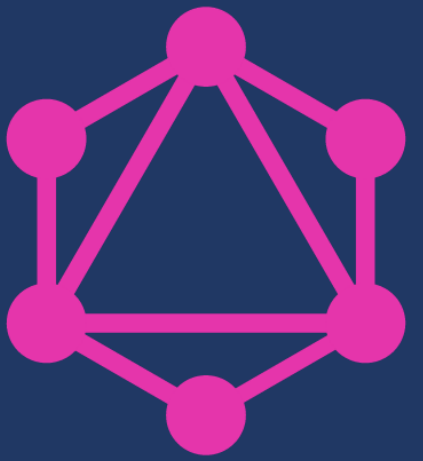
GET /books_overview
GET /books?dataset=books_with_authors
GET /books?dataset=books_with_authors_with_ratings

Multiple views?

Multiple client types?

Private / Public API

GraphQL



A query language for your API

Just for once, let me look on you with my own eyes.
Anakin Skywalker

https://pixabay.com/en/star-wars-warrior-space-cinema-2473201/

GraphiQL


GraphiQL
Prettify
History
query {
books {
title
author {
firstName
lastName
}
}
}
id
firstName
lastName
books
the id of the author
{
"data": {
"books": [
{
"title": "Algorithms to Live By: The Computer Science ...",
"author": {
"firstName": "Brian",
"lastName": "Christian"
}
},
{
"title": "Cracking the Coding Interview: 189 Programming ...",
"author": {
"firstName": "Gayle",
"lastName": "Laakmann McDowell"
}
},
{
"title": "Life 3.0: Being Human in the Age of Artificial ...",
"author": {
"firstName": "Max",
"lastName": "Tegmark"
}
},
{
Query
Author
Search Author...
an author
FIELDS
id: Int!
the id of the author
firstName: String
the firstname of the author
lastName: String
the lastname of the author
books: [Book]
the list of books this author has written

GraphQL is ...

Specification

Reference implementation

Schema (domain specific)

Independent of the data store

Independent of the transport protocol

GraphQL Features

GraphQL Operations

Query	Read
Mutation	Write
Subscription	Notification (“Event”)

Fields

```
query {
  books {
    title
    author {
      firstName
      lastName
    }
  }
}
```

Arguments

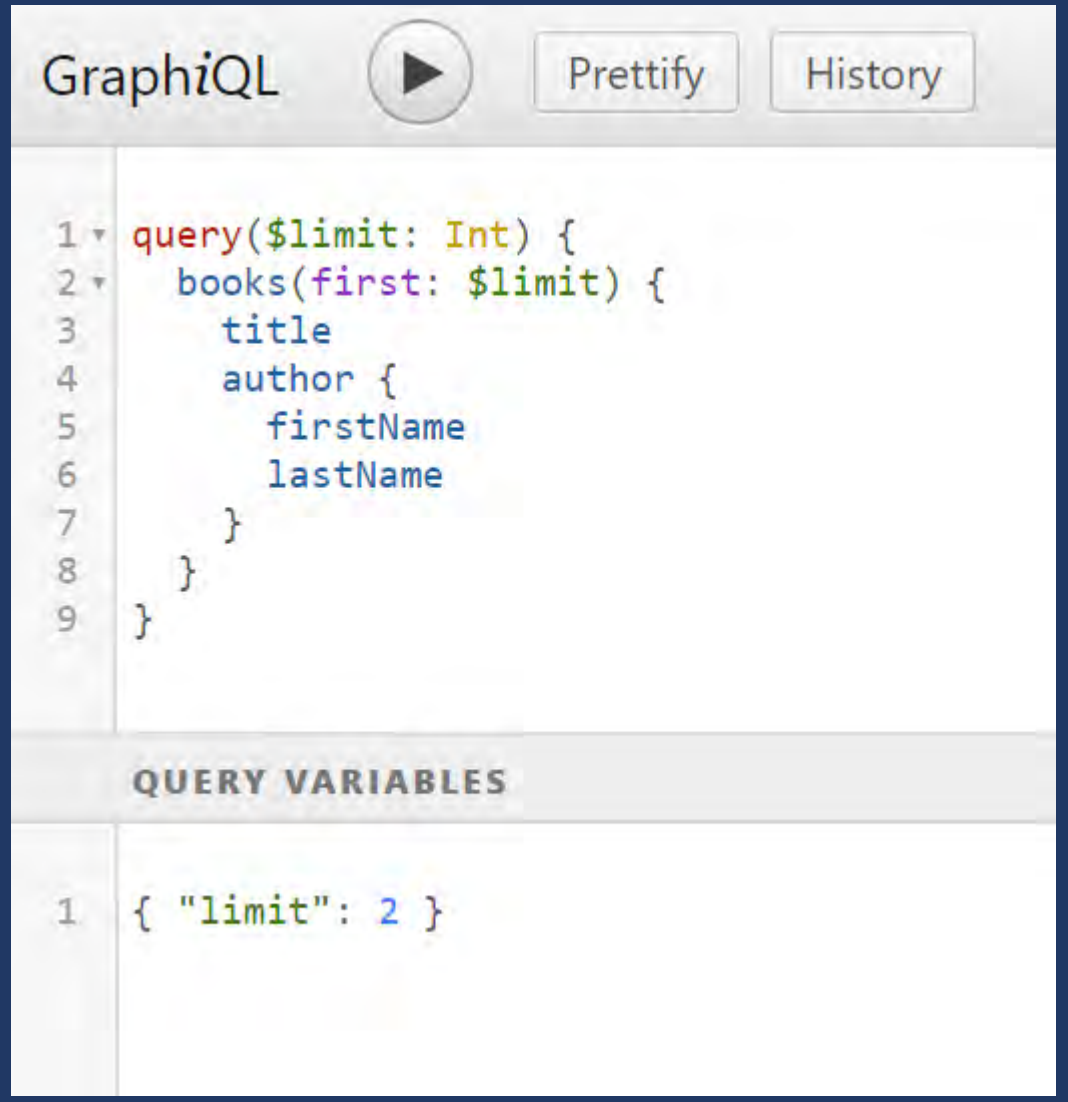
```
query {
  books(first: 2) {
    title
    author {
      firstName
      lastName
    }
  }
}
```

Fragments

```
fragment authorFragment on Author {
  firstName
  lastName
}

query {
  books {
    title
    author {
      ...authorFragment
    }
  }
}
```

Variables

```
query($limit: Int) {
  books(first: $limit) {
    title
    author {
      firstName
      lastName
    }
  }
}
```



Validation / Errors

Introspection

GraphiQL ▶ Prettify History

```
query {
  __schema {
    types {
      name
    }
  }
}
```

```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "Query"
        },
        {
          "name": "Author"
        },
        {
          "name": "Int"
        },
        {
          "name": "String"
        },
        {
          "name": "Book"
        },
```

GraphiQL ▶ Prettify History

```
query {
  __type(name: "Author") {
    name
    description
    fields {
      name
      description
      |
    }
  }
}
```

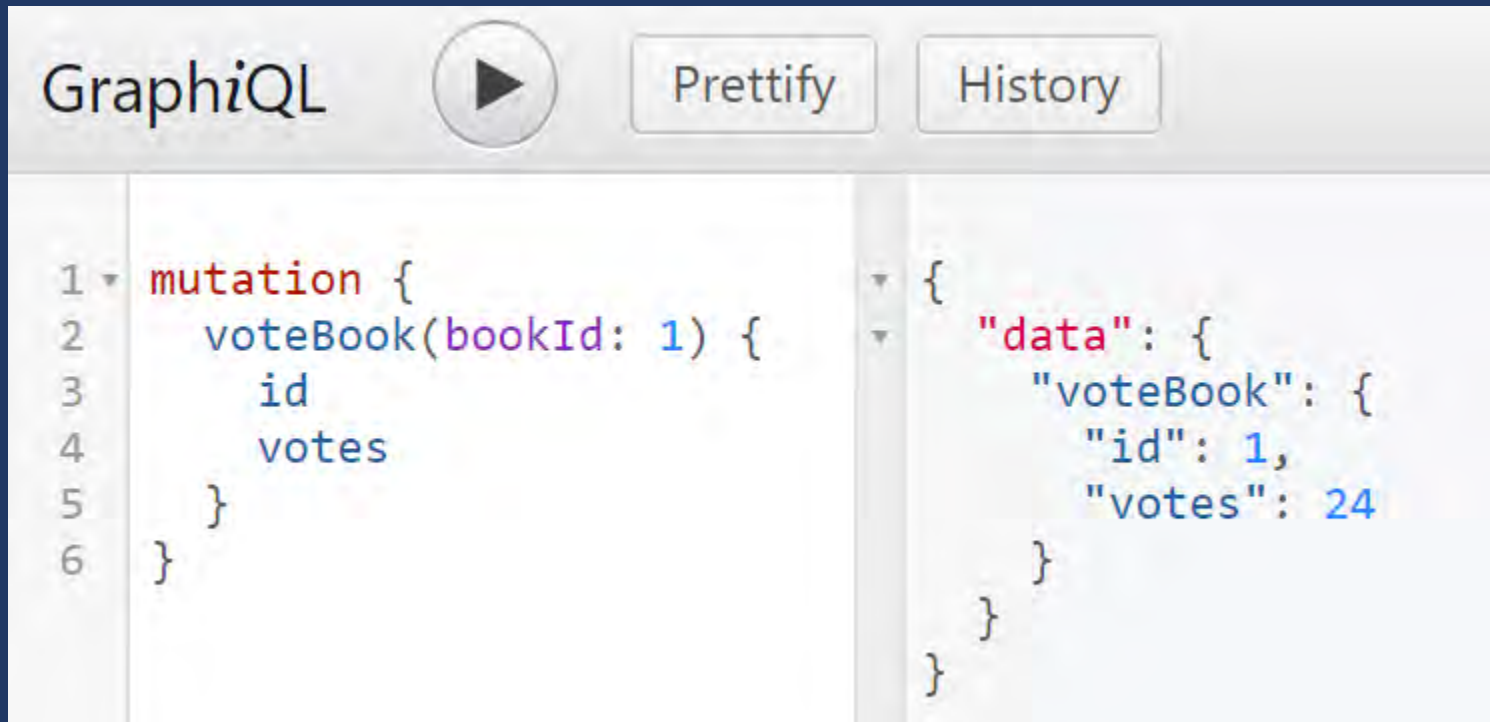
name
description
args
type
isDeprecated
deprecationReason

Self descriptive.

```
{
  "data": {
    "__type": {
      "name": "Author",
      "description": "an author",
      "fields": [
        {
          "name": "id",
          "description": "the id of the author"
        },
        {
          "name": "firstName",
          "description": "the firstname of the author"
        },
        {
          "name": "lastName",
          "description": "the lastname of the author"
        },
```

Mutations

```
mutation {
    voteBook(bookId: 1) {
      id
      votes
    }
}
```

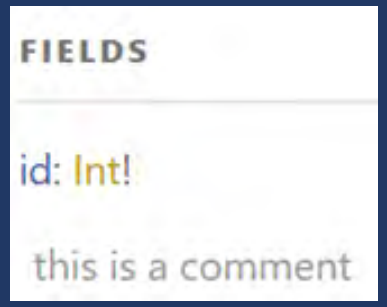


Schema

May the force be with you

https://pixabay.com/en/star-wars-lightsaber-laser-sword-2369317/

Schema

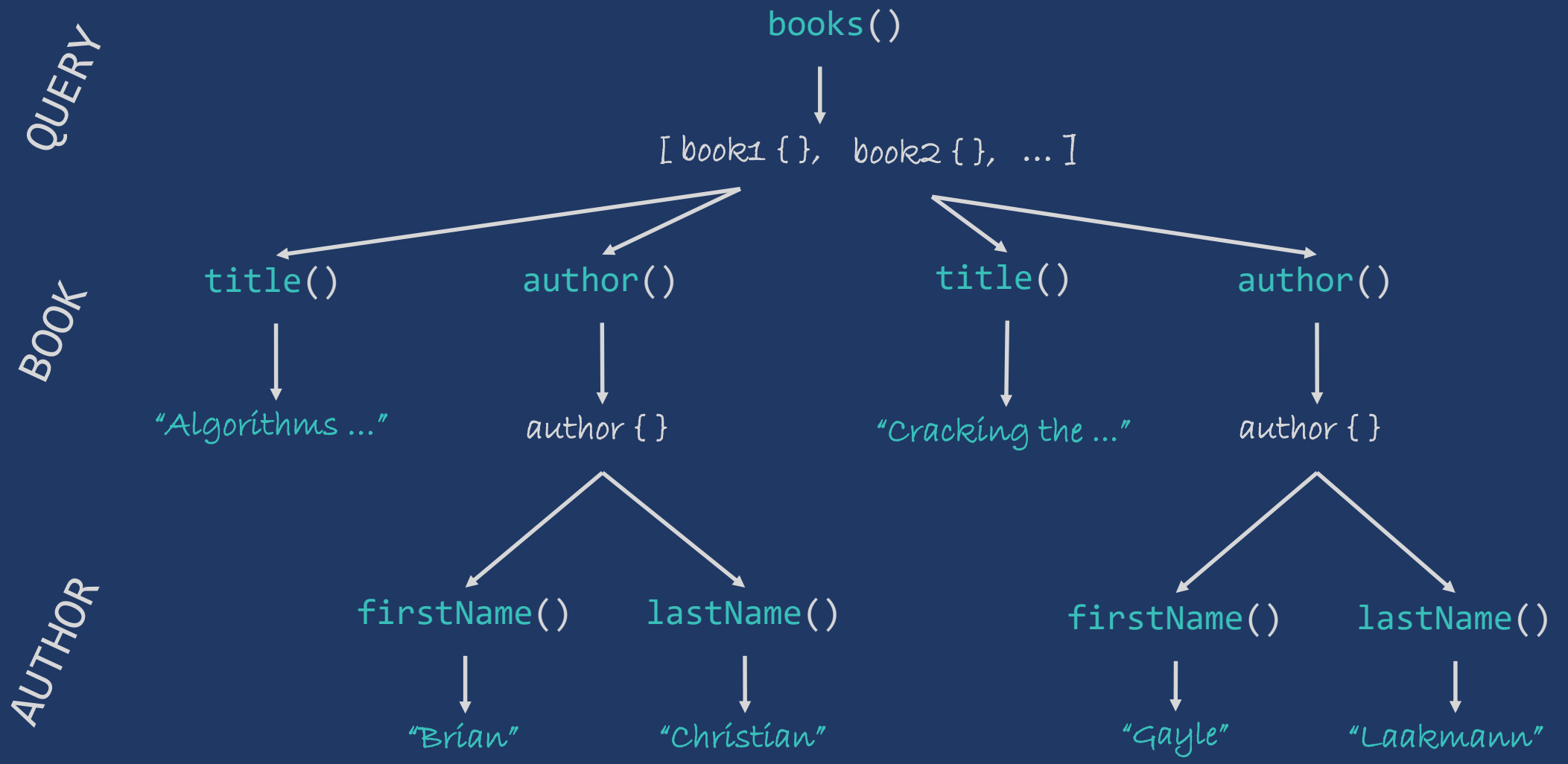



```
type Book {
  # this is a comment
  id: Int!
  title: String
  author: Author
}

type Author {
  id: Int!
  firstName: String
  lastName: String
  books: [Book]
}
```

```
type Query {
  books: [Book]
  authors: [Author]
}

type Mutation {
  voteBook(bookId: Int!): Book
}
```

GraphQL on the Server

Resolvers




```
query {
  books {

    title
    author {


      firstName
      lastName
    }
  }
}
```

Execution

```
query {
  books {
    title
    author {
      firstName
      lastName
    }
  }
}
```

```
db.getAllBooks()
db.getAuthorForBook(1)
db.getAuthorForBook(2)
db.getAuthorForBook(3)
db.getAuthorForBook(4)
db.getAuthorForBook(5)
db.getAuthorForBook(6)
```

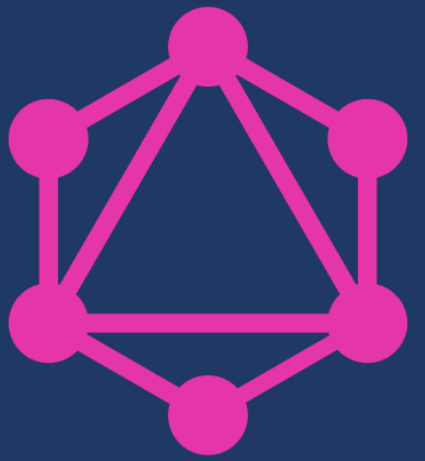
Caching

Batching

DataLoader

https://github.com/facebook/dataloader

GraphQL on the Client



APOLLO CLIENT

Bind GraphQL data to your UI with one function call.

Apollo Client is the ultra-flexible, community-driven GraphQL client for React, JavaScript, and native platforms.

GET STARTED CONTRIBUTE

Apollo Client works with every frontend platform:


React »


Vue.js »


Angular »

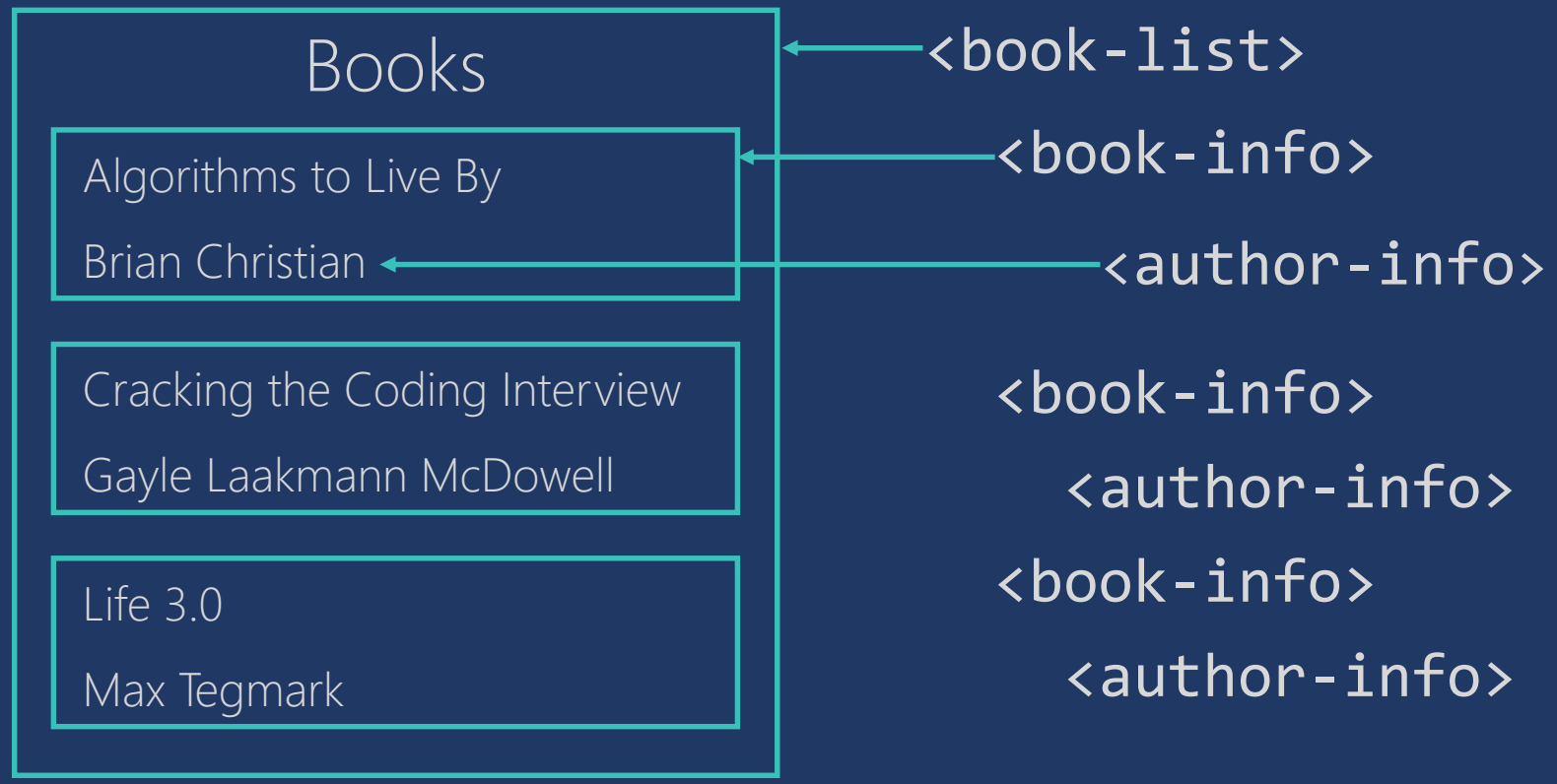

Android »


iOS »

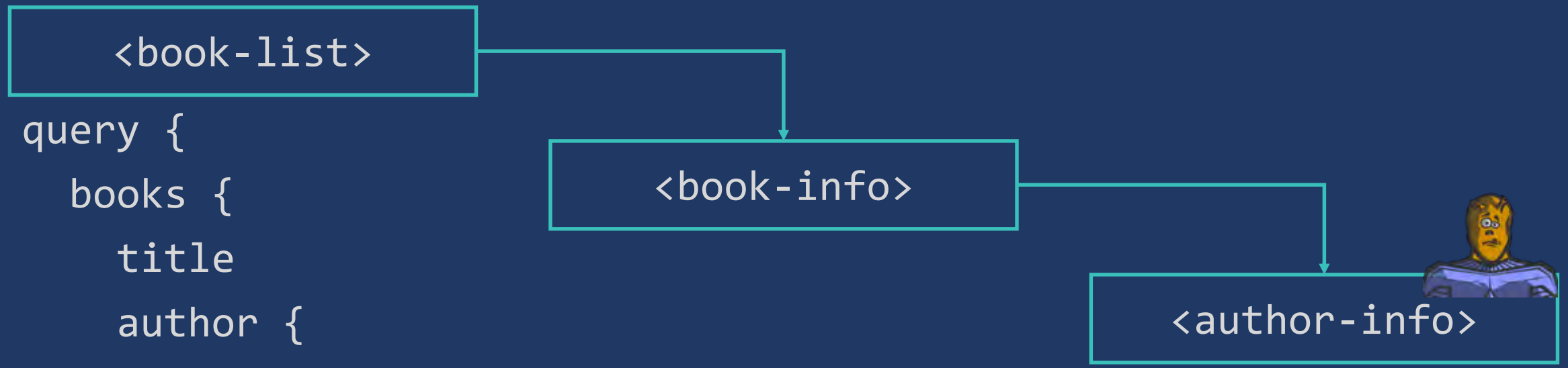
Ember »


Meteor »

UI - Structure



Version 1




```
query {
  books {
    title
    author {
      firstName
      lastName
      profilePic
    }
  }
}
```

Version 2

<book-list>

```
query bookList {
  books {
    ...book
  }
}
```

<book-info>

```
fragment book
  on Book {
    title
    author {
      ...author
    }
}
```

<author-info>

```
fragment author
  on Author {
    firstName
    lastName
    profilePic
}
```

<author-info>

```
fragment author
on Author {
  firstName
  lastName
  profilePic
}
```

TypeScript

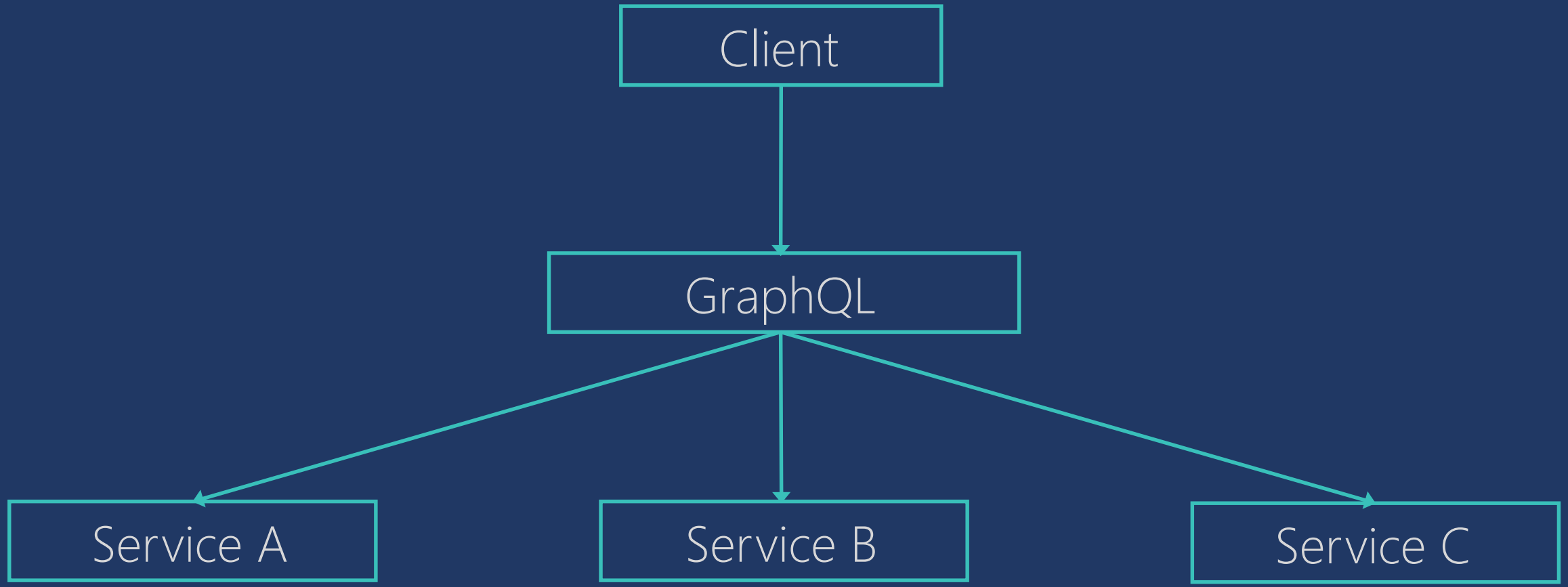
```
export interface authorFragment = {
  firstName: string | null,
  lastName: string | null,
  profilePic: string | null,
};
```

https://github.com/apollographql/apollo-codegen

REST vs GraphQL

https://www.pexels.com/photo/man-couple-people-woman-343/

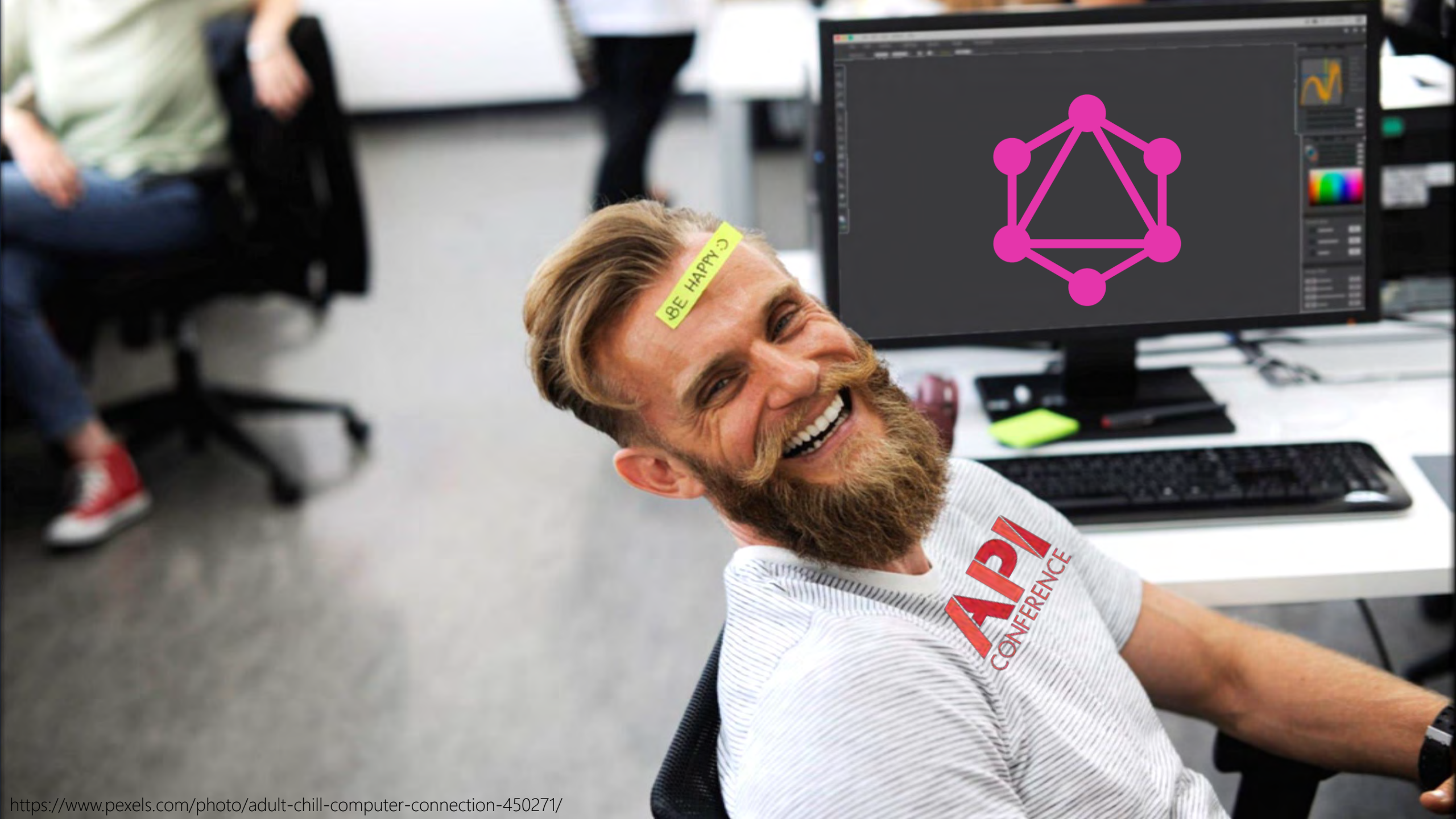
One last note: GraphQL as Gateway



https://www.pexels.com/photo/photo-of-a-hippie-woman-736603/

Christian Schwendtner
PROGRAMMIERFABRIK GmbH Linz, Austria

@CSchwendtner



https://www.pexels.com/photo/adult-chill-computer-connection-450271/