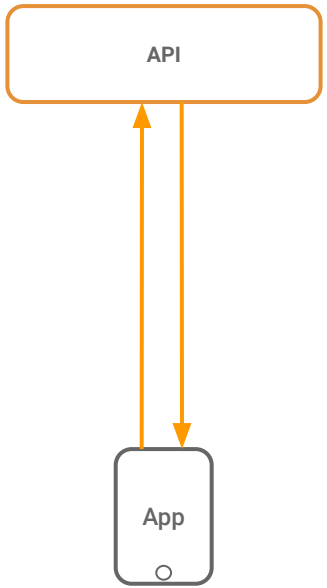
GraphQL is the API developers ❤️

@razvanprichici

#1
An API call

An API call (before)



HTTP request

```
GET /api/user?id=1
```

HTTP response

```
{
  "id": 1,
  "name": "Elmo"
}
```

```
GET /api/address?user_id=1
```

```
{
  "street": "Sesame street",
  "city": "New York City"
}
```

An API call (before)

Yuck! 2 API calls.

It can only get worse with more API calls.

An API call (before)

Let’s talk to our API developer to help us out.

With one API call.

HTTP request

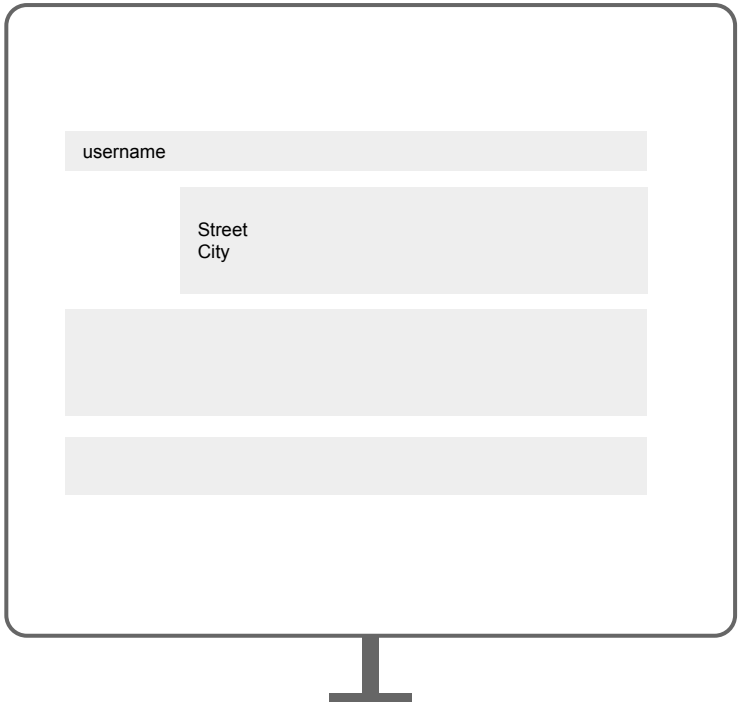
```
GET /api/userinfo?id=1
```

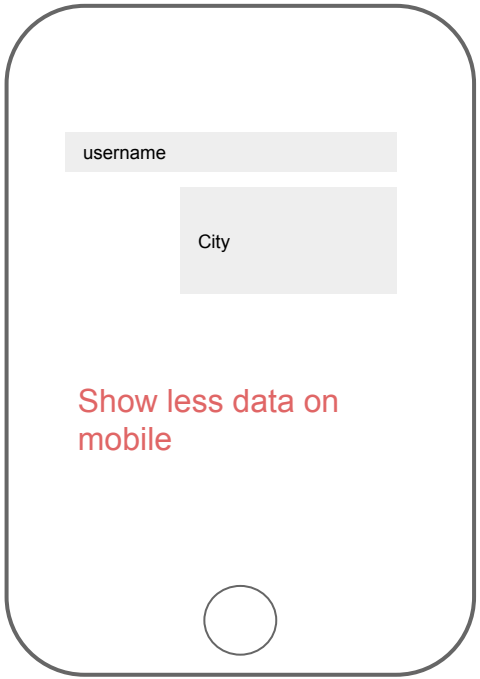
HTTP response

```
{
  "id": 1,
  "name": "Elmo"
  "address": {
     "street": "Sesame street",
     "city": "New York City"

  }
}
```

An API call (before)





An API call (before)

Let's talk to our API developer to help us out. *Again*.

With one API call that *takes params*

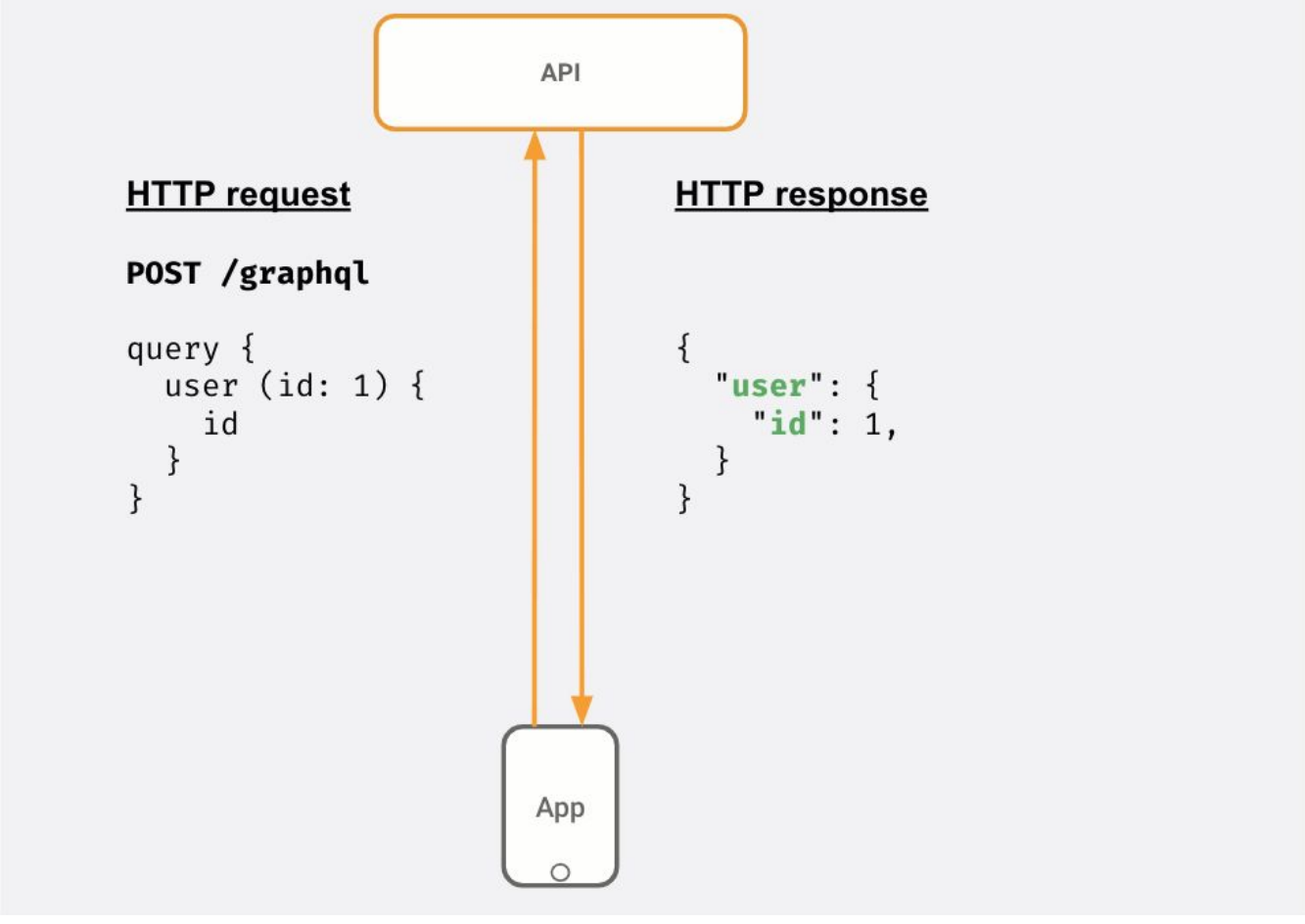
HTTP request

```
GET /api/userinfo?id=1&fields=id,name,address.city
```

HTTP response

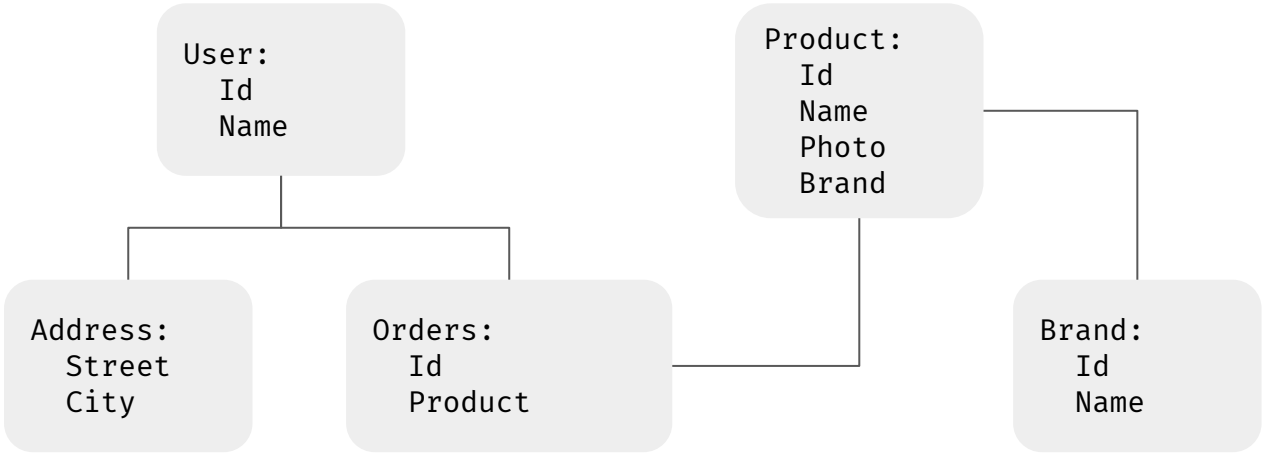
```
{
  "id": 1,
  "name": "Elmo"
  "address": {
     "city": "New York City"
  }
}
```

An API call (after)



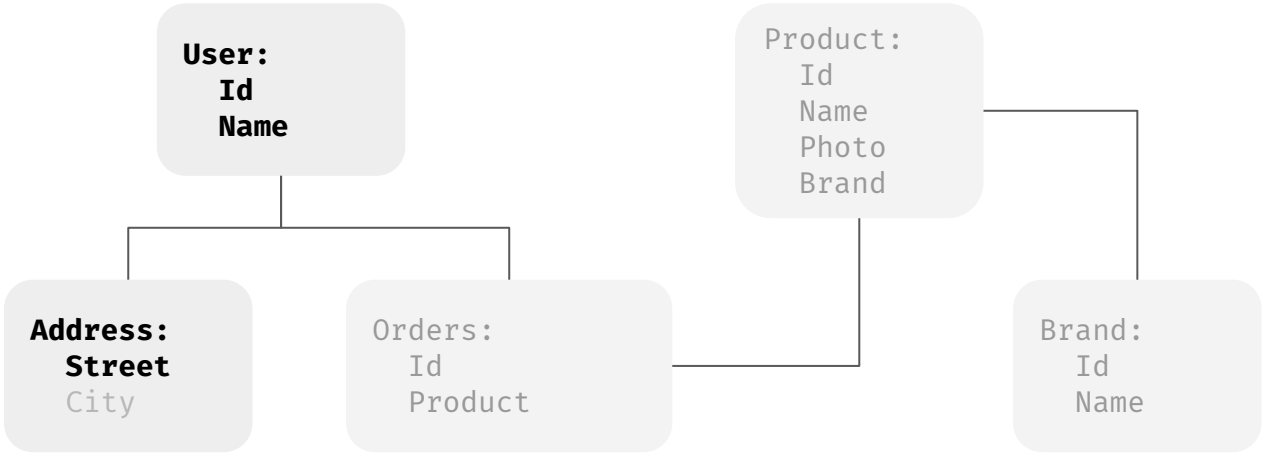
Key insights #1

Your API models are “graph” like.



Key insights #2

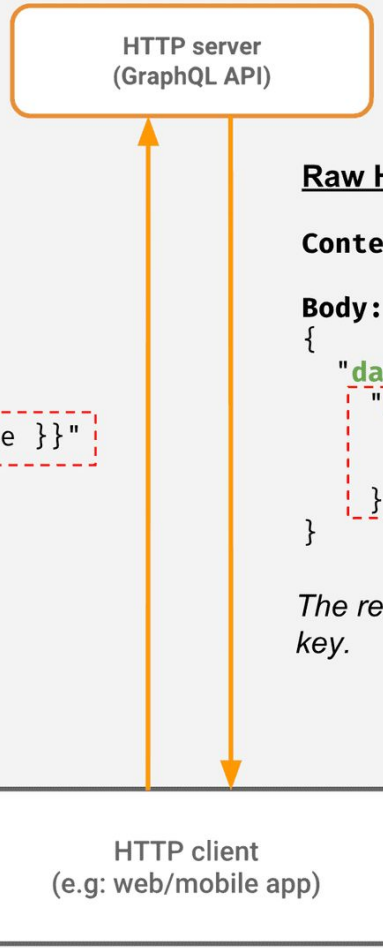
You want to control the data you get



A GraphQL query

```
query {
  user (id: 1) {
    id
    name
    address {
      street
    }
  }
}
```

GraphQL underneath



Raw HTTP request

```
Method:       POST
URL:          https://api.com/graphql
Content-Type: application/json

Body:
{
  "query": "query { user (id: 1) { id name }}"
}
```

The GraphQL query is sent as a string inside a JSON object.

Raw HTTP response

```
Content-Type: application/json

Body:
{
  "data": {
    "user": {
      "id": 1,
      "name": "Elmo"
    }
}
```

*The response object is inside the **`data`** key.*

What is GraphQL?

1) GraphQL is an **API query language**, not a database query language.

2) While GraphQL exposes your application data as a graph, it's **not (just) for graph databases**

#3
“Write” APIs

“Writing” to your API (before)

- POST
- PUT
- DELETE
- PATCH

HTTP request

```
POST /api/todo

{
  "todo": "Grok GraphQL"
}
```

HTTP response

```
200

{
  "id": 987
}
```

“Writing” to your API (after)

HTTP request

POST /graphql

```
mutation {
  addTodo(todo: $newTodo) {
    id
  }
}
```

```
{
  "newTodo": {
    "todo": "Grok GraphQL"
  }
}
```

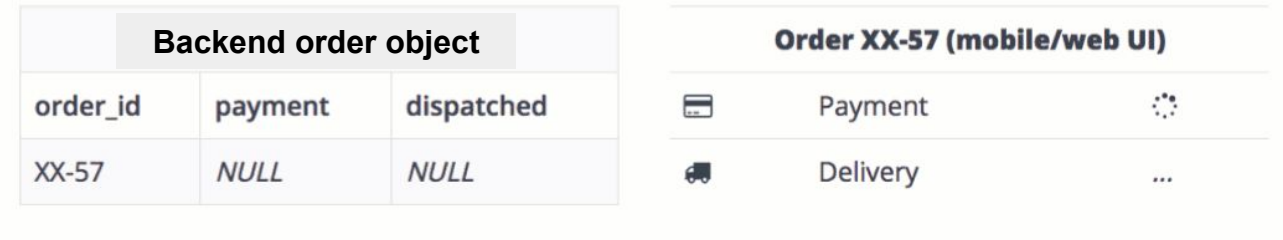
Query variable

HTTP response

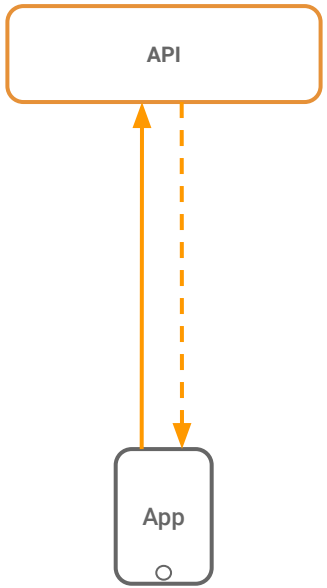
200

```
{
  "id": 987
}
```

#4
“Realtime” APIs



“Realtime” APIs (before)



Option 1: Polling

Client makes repeated requests every X seconds to refetch data.

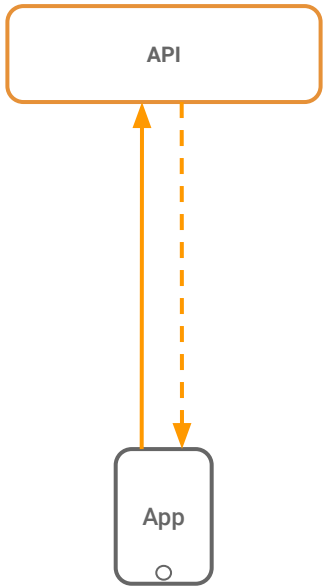
#yuck

Option 2: Websockets

Server pushes data to the client over websockets.

#nightmare

“Realtime” APIs (after)



HTTP request

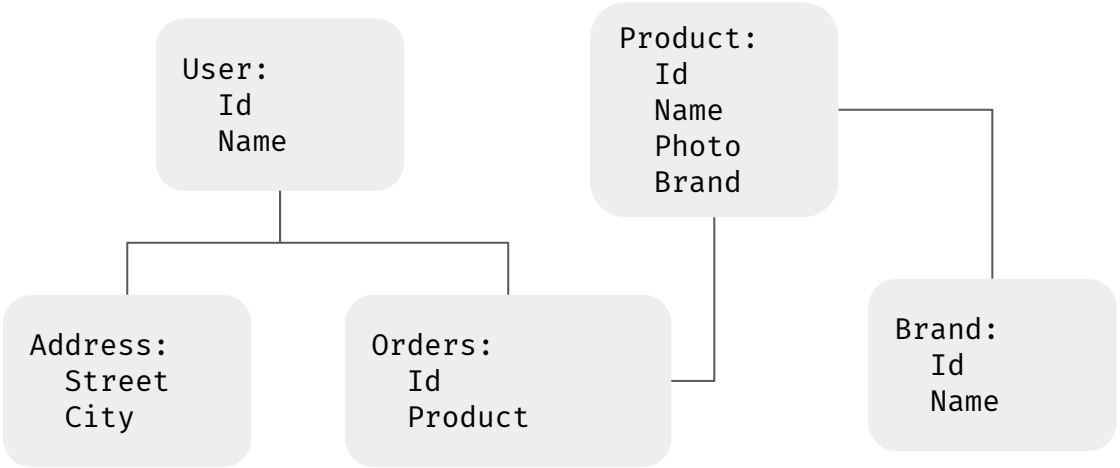
```
ws://myapi.com/graphql

subscription {
  order(id: "XX-57") {
    paid
    dispatched
  }
}
```

HTTP response

```
{
  "paid": true,
  "dispatched": false,
}
```

GraphQL schema: The type-system of your API




```
type User {
  id: Int
  name: String
  address: Address
}
```

```
type Address {
  id: Int
  street: String
  city: String
}
```

Challenges with adding GraphQL

GraphQL Challenges

- Some well understood practices from REST don't apply
  - HTTP status codes
  - Errors
  - Caching
- Exposing arbitrary complexity to client
  - Performance considerations
- n+1 query problem
- Query costing / rate limiting

N+1 queries to your database

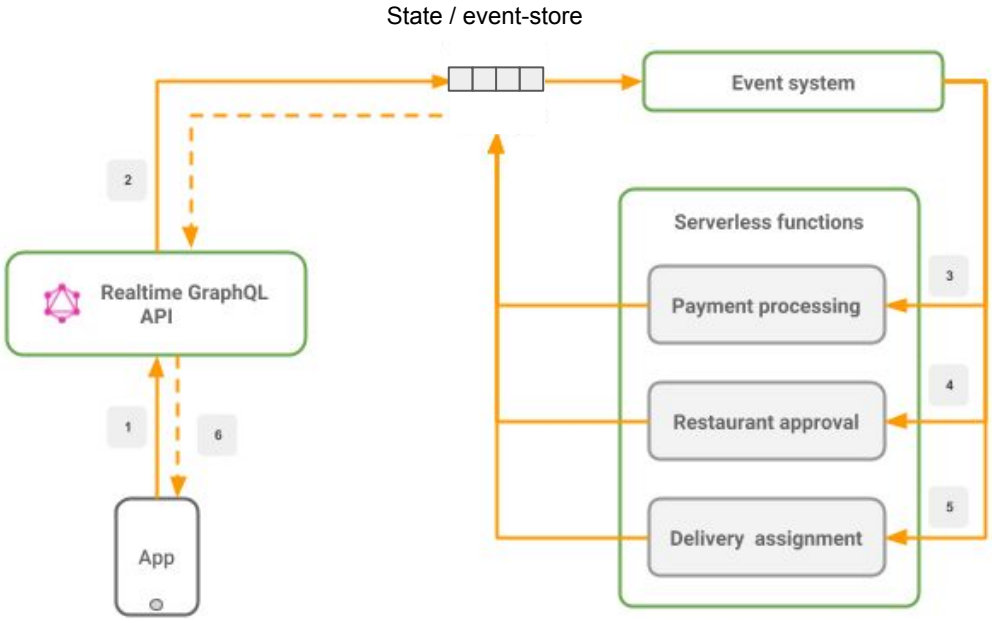
Terrible for your database; bad for for DBAs to debug

```
query {
  authors {
    name
    posts {
      title
      content
    }
  }
}
```

```
# Make a request for authors
authors = db.get('author') # 1 query

# Start iterating over each author
for author in authors:
  # Fetch all the posts for each author
  posts = author.post() # N queries
```

“Flux” style one way data-flow



Business benefits to a GraphQL API

API onboarding

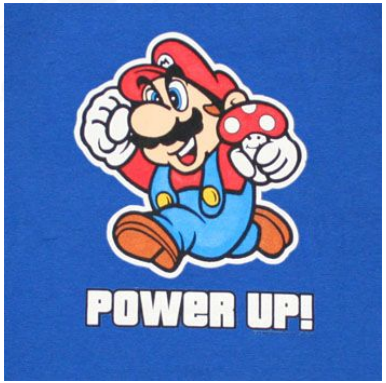
- Exploring APIs is easy
- Documentation is auto-generated
- No versioning required for non-destructive changes
- Auto-generated SDKs for all languages, community maintained

💯 Agility for API consumers

- Auto-generated SDKs for all languages, community maintained
- Auto-complete for API calls in their IDEs

GraphQL Advantages

- **Overfetching**
  - Less data over the wire
- **Underfetching**
  - Single round trip
- GraphQL **Specification**
- **“Graphs All The Way Down”**
  - **Relationships** vs Resources
  - Unify disparate systems (microservices)
- **Simplify data fetching**
  - Component based data interactions



Summary

Summary

GraphQL is a modern data API

GraphQL makes life easy for API consumers

GraphQL has serious team/org benefits

GraphQL is challenging, but solutions and patterns are emerging