

# GraphQL

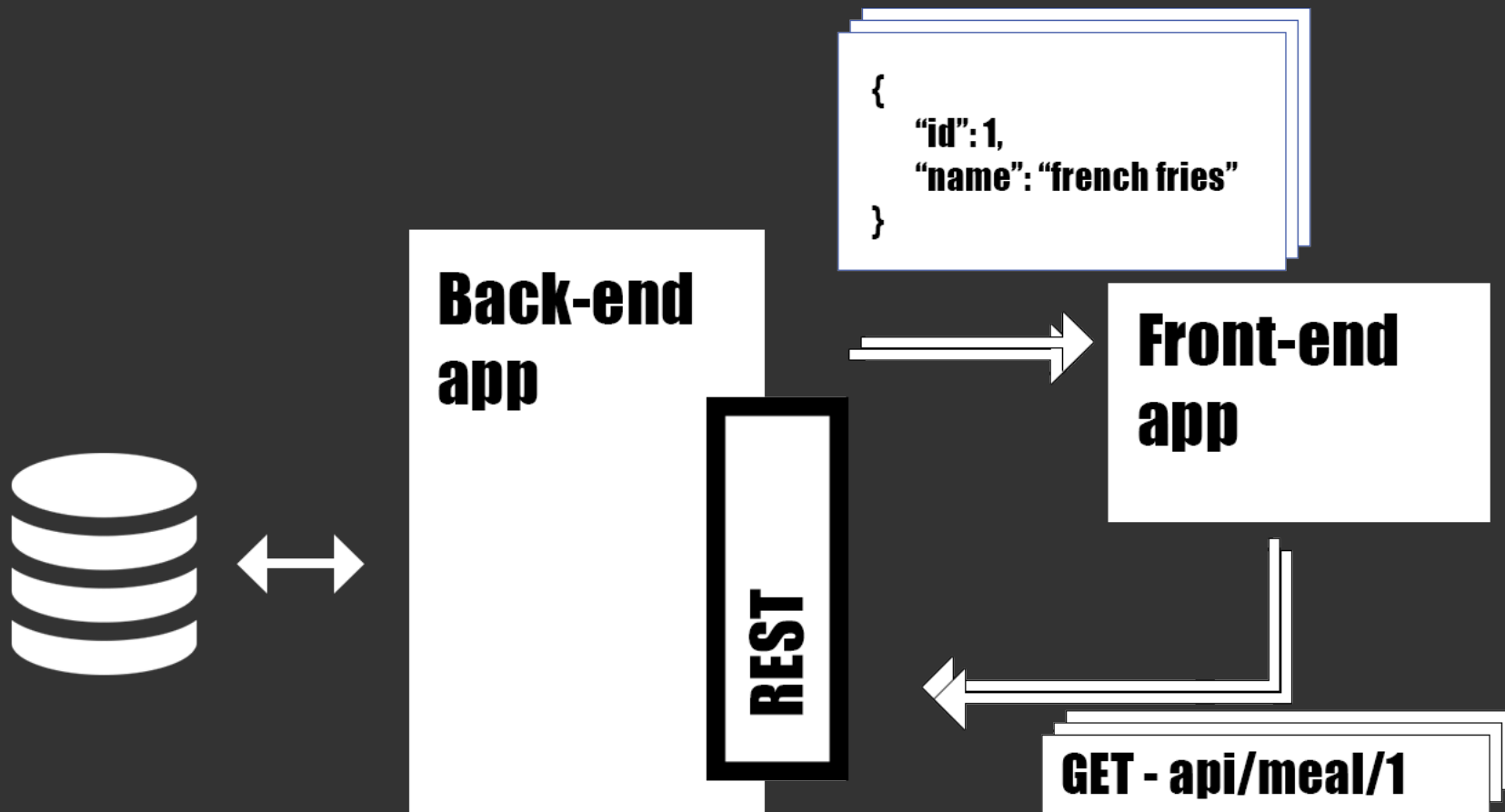
November 2018

# **GraphQL or back-end/front-end communication done right**

November 2018

Danielo **JEAN-LOUIS**  
Front-end developer

# Current workflow with an API



**It works well but...**

**...what's happen  
if the response is too complex ?**

**At Facebook,  
they have (I guess) User Stories like this one**



**As a user I want to know if one of my  $n+3$  friend likes french fries and only this**



**Me**



**Friend**



**Friend's  
friend's  
friend**

**Let's try to query a pseudo API**

GET /users/me

```
{  
  "user": {  
    [...]  
    friends: [1, 2, 3, 4...]  
  }  
}
```



**Me**

GET /users/me

```
{  
  "user": {  
    [...]  
    friends: [1, 2, 3, 4...]  
  }  
}
```

**Oh wait, I need my friend's data**

GET /users/me

GET /users/23

```
{  
  "user": {  
    [...]  
    friends: [1, 2, 3, 4...]  
  }  
}
```



**Me**



**Friend**

GET /users/me

GET /users/23

```
{  
  "user": {  
    [...]  
    friends: [1, 2, 3, 4...]  
  }  
}
```

**Oh wait, I need my friend's friends data**

GET /users/me

GET /users/23

GET /users/42

```
{  
  "user": {  
    [...]  
    friends: [18, 122, 32, 41...]  
  }  
}
```



**Me**



**Friend**





GET /users/me

GET /users/23

GET /users/42

```
{  
  "user": {  
    [...]  
    friends: [18, 122, 32, 41...]  
  }  
}
```

**Oh wait, I need my friend's friend's friends data**

```
GET /users/me
GET /users/23
GET /users/42
GET /users/7777
```

```
{
  "user": {
    [...]
    like_french_fries: true
  }
}
```



**Me**



**Friend**



**Friend's  
friend's  
friend**

GET /users/me

GET /users/23

GET /users/42

GET /users/7777

```
{  
  "user": {  
    [...]  
    like_french_fries: true  
  }  
}
```

**Finally we have the data we want**

**For  $n+3$  level friend's info, I need to do, at least, four requests**

**And there's more problems!**

# Problems

- Bunch of useless keys returned\*
- Intensive bandwidth usage
- Multiple queries
- Can lead to complex callbacks
- Multiple endpoints (One for each CRUD part)

\* Yeah, we can pass as parameters which ones I need, but... no

**Change request !**

**The product owner doesn't want  
this feature anymore**



**Easy ticket, right ?**  
**But...**

**What about the old version of the front-end app?**

**I can't remove the key  
“like\_french\_fries”.\***

\* Yeah, we can create a new version of the API, but... no

**What about new developers and  
API's documentation ?**

**I forgot to create a swagger file.**

**Facebook devs had to handle this  
often, too often**

# So they created GraphQL

## Sources

<https://code.fb.com/core-data/graphql-a-data-query-language/>

# GraphQL

Developed by Facebook

Used internally since 2012

Open sourced in July 2015

Exists for almost all backend languages

## Sources

<https://graphql.org>



# GraphQL

No new server type

# GraphQL

No new server type

No new programming language

**Just a data query language for your  
APIs**

You get only what you **want**, what  
you **need**

# Workflow with GraphQL



**Back-end  
app**

**GraphQL**

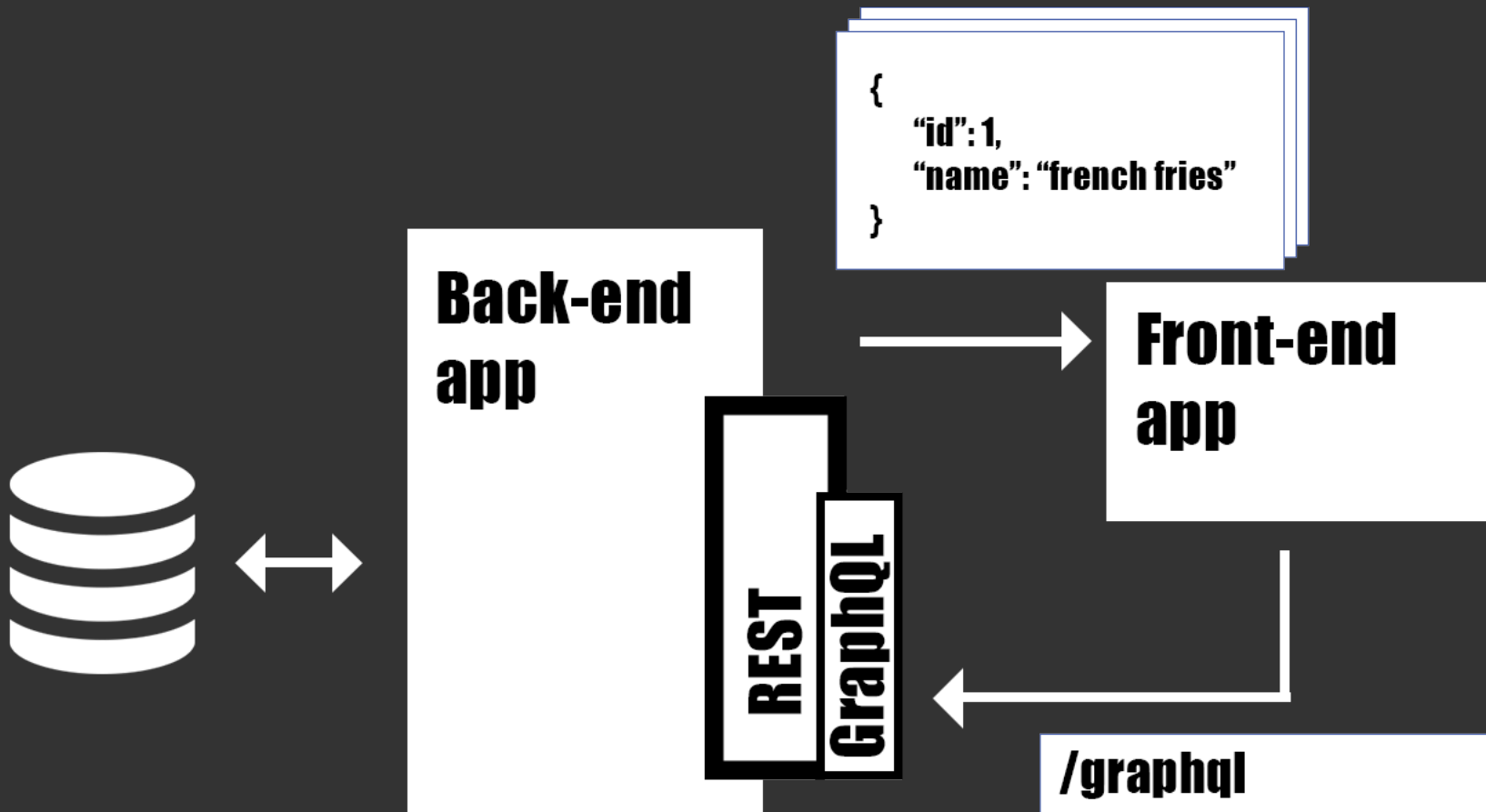
```
{  
  "id": 1,  
  "name": "french fries"  
}
```

**Front-end  
app**

**/graphql**



**GraphQL is a proxy of your API / DB / ...**





**The front-end is now connected to GraphQL**



**Back-end  
app**

**REST**



```
{  
  "id": 1,  
  "name": "french fries"  
}
```

**Front-end  
app**



**GET - api/meal/1**

**Let's summarize our main problems  
with the old way**

# Main problems with the old way

- Over-fetching (useless keys)
- Documentation
- Multiple queries

# Let's fix our problems with Reddit's GraphQL

## Sources

<https://www.graphqlhub.com/>

# Over-fetching

With GraphQL you describe to the server which data you need **and only this**.

# Over-fetching

```
1 {  
2   graphqlHub  
3   reddit {  
4     subreddit(name: "graphql") {  
5       title  
6     }  
7   }  
8 }  
9
```

```
{  
  "data": {  
    "graphqlHub": "Use GraphQLHub to explore popular APIs with GraphQL! Created by Clay Allsopp @  
    "reddit": {  
      "subreddit": {  
        "title": "GraphQL"  
      }  
    }  
  }  
}
```

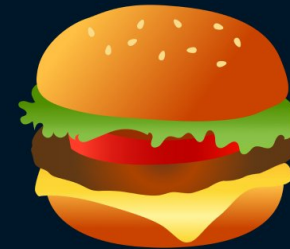
## Sources

<https://www.graphqlhub.com>

# Over-fetching

## GraphQL & Rest: A burger comparison

`https://your-api.com/burger/`



```
query getBurger {  
  burger {  
    bun  
    patty  
    bun  
    lettuce  
  }  
}
```



### Sources

<https://twitter.com/NikkitaFTW/status/1011928066816462848>



**Over-fetching**

# ~~Over-fetching~~\*

\* It also fixes the issue with the API versioning  
<https://graphql.org/learn/best-practices/#versioning>

# Documentation

GraphQL needs a data schema in order to work and to create the documentation **automatically**

# Documentation

< RedditSubredditRedditLink×

A link posted to a subreddit

**FIELDS**

title: String!

fullnameld: String!

score: Int!

numComments: Int!

url: String!

author: RedditUser!

comments(depth: Int, limit: Int): [RedditComment]!

# Documentation

**~~Documentation~~**

# Multiple queries

- We provide a data schema to GraphQL

# Multiple queries

- We provide a data schema to GraphQL
- The data schema contains entities relationships



# Multiple queries

- We provide a data schema to GraphQL
- The data schema contains entities relationships
- And we get in **one query** everything we need

# Multiple queries

```
1 {  
2   graphqlHub  
3   reddit {  
4     subreddit(name: "graphql") {  
5       title  
6       hotListings(limit: 3) {  
7         title  
8         author {  
9           fullNameId  
10          username  
11        }  
12      }  
13    }  
14  }  
15 }  
16
```

```
{  
  "data": {  
    "graphqlHub": "Use GraphQLHub to explore popular APIs with GraphQL! Created by Clay Allsopp @c",  
    "reddit": {  
      "subreddit": {  
        "title": "GraphQL",  
        "hotListings": [  
          {  
            "title": "graphql-kotlin: Generate a GraphQL schema from Kotlin code",  
            "author": {  
              "fullNameId": "t2_9m25m",  
              "username": "smyrick"  
            }  
          },  
          {  
            "title": "This belongs here:",  
            "author": {  
              "fullNameId": "t2_e9vd8",  
              "username": "ISkiAtAlta"  
            }  
          },  
          {  
            "title": "altair-express-middleware- express middleware for mounting an instance of al",  
            "author": {  
              "fullNameId": "t2_qabe1",  
              "username": "imolorhe"  
            }  
          }  
        ]  
      }  
    }  
  }  
}
```

# Multiple queries

~~Multiple queries~~

**All of our problems are fixed!**

# GraphQL other nice features

- simple syntax
- fragment (“keys” aliases)
- mutations (for delete/update/create)
- query aliases
- directives
- GraphiQL / ChromeiQL – GUI for GraphQL
- Subscription (rfc currently)
- Types and custom Types
- Only one entriypoint for everything
- and more

## Sources

<https://graphql.org/learn/queries/>

# Demo

## Sources

<https://github.com/DanYellow/presentations/tree/master/graphql/examples/node>

<https://github.com/DanYellow/presentations/tree/master/graphql/examples/php>

## dbo.editor

id	int	IDENTITY	PK
<hr/>			
name	string		AK
creation_date	datetime		
photo	string		

## author\_editor

author_id	int	PK	FK
editor_id	int	PK	FK
<hr/>			

## dbo.author

id	int	PK
<hr/>		
firstName	string	
lastName	string	
photo	string	

## dbo.book

id		PK
<hr/>		
title	string	
release_date	datetime	
cover_image	string	
summary	string	
author_id	int	FK
editor_id	int	FK



# Resolvers & Schema at glance

**Resolvers** are functions connected to your backend / api or other. They describe how and where the data will be fetched.

# Resolvers & Schema at glance

**Resolvers** are functions connected to your backend / api or other. They describe how and where the data will be fetched.

**Schema** is the model describing which data are *fetchable* in the GraphQL server. They list which queries are available.

# Who's using GraphQL?

- Facebook
- GitHub
- Pinterest
- Allocine
- Shopify

...

## Sources

<https://graphql.org/users/>

# Who's using GraphQL?

- Facebook
- GitHub
- Pinterest
- Allocine
- Shopify
- ...
- You?

## Sources

<https://graphql.org/users/>

# Who's using GraphQL?

- Facebook
- GitHub
- Pinterest
- Allocine
- Shopify
- ...
- You?
- Your backend developers?

## Sources

<https://graphql.org/users/>

## Summary / conclusion

- GraphQL is not a new programming language
- It allows to “get only what you want” from backend
- Fixes frequent problems with “classic API”

# More resources

- Presentation + examples:

<https://github.com/DanYellow/presentations/tree/master/graphql>

- GraphQL playgrounds:

<http://apis.guru/graphql-apis/>

- Official documentation:

<https://graphql.org/>

- Articles:

<https://blog.apollographql.com/graphql-explained-5844742f195e>

<https://blog.apollographql.com/how-do-i-graphql-2fcabfc94a01>





**Questions ?**

