

# GraphQL

November 2018

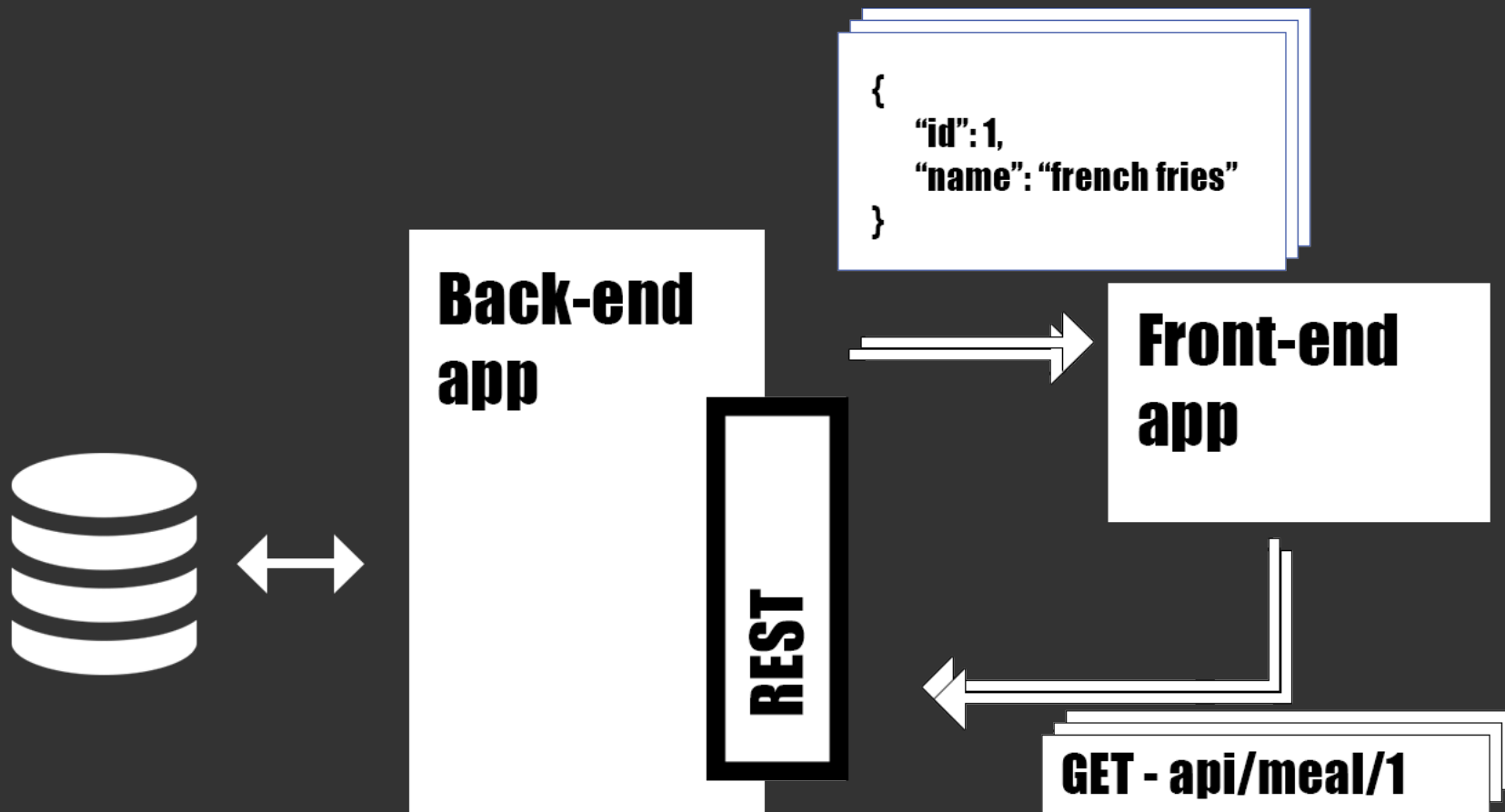
# **GraphQL or back-end/front-end communication done right**

November 2018



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

# Current workflow with an API



# Current workflow with an API

- Front-end asks for a resource
- Backend analyzes the request
- Backend queries the db
- DB returns the data
- Backend returns the resource

**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  $n+2$  friend likes french fries and only this**



**Me**



**Friend**



**Friend of  
friend of  
friend**

**{ }**

**{ { } }**

**{ { { } } }**

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

GET /users/me

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

GET /users/me

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

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

GET /users/me

GET /users/23

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

GET /users/me

GET /users/23

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

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



GET /users/me

GET /users/23

GET /users/42

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

GET /users/me

GET /users/23

GET /users/42

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

**Finally we have the data we want**

**For  $n+2$  level friend's info, I need to do, at least, **three** requests**

**And there's more problems!**

# Problems

- Bunch of useless keys returned\*
- Use bandwidth intensively
- 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 app?**

**I can't remove the key  
“love\_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**

# GraphQL

Developed by Facebook

Used internally since 2012

Open sourced in July 2015

Exists for almost all backend languages

## Sources

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

# GraphQL

No new server type



# GraphQL

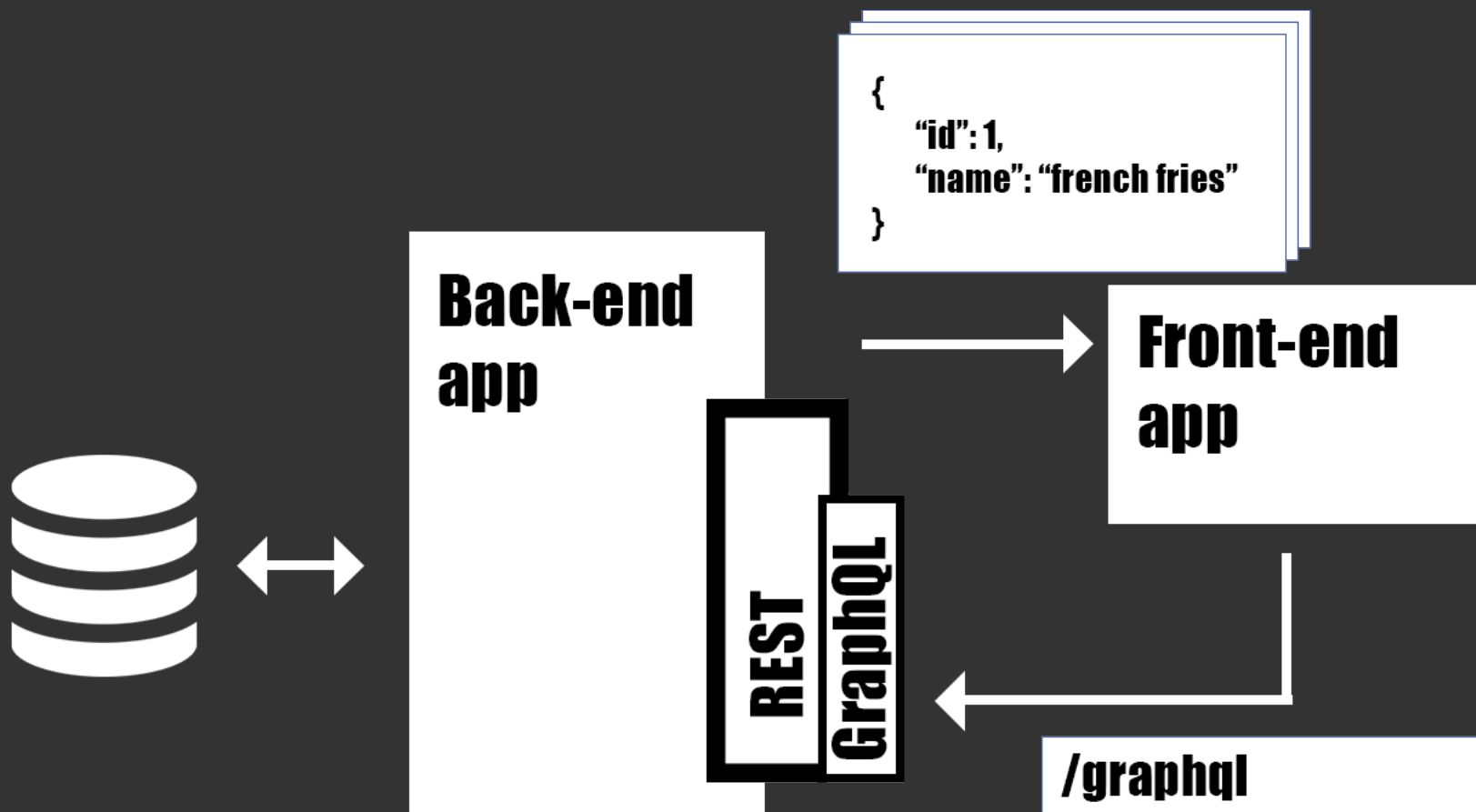
No new server type

No new programming language

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

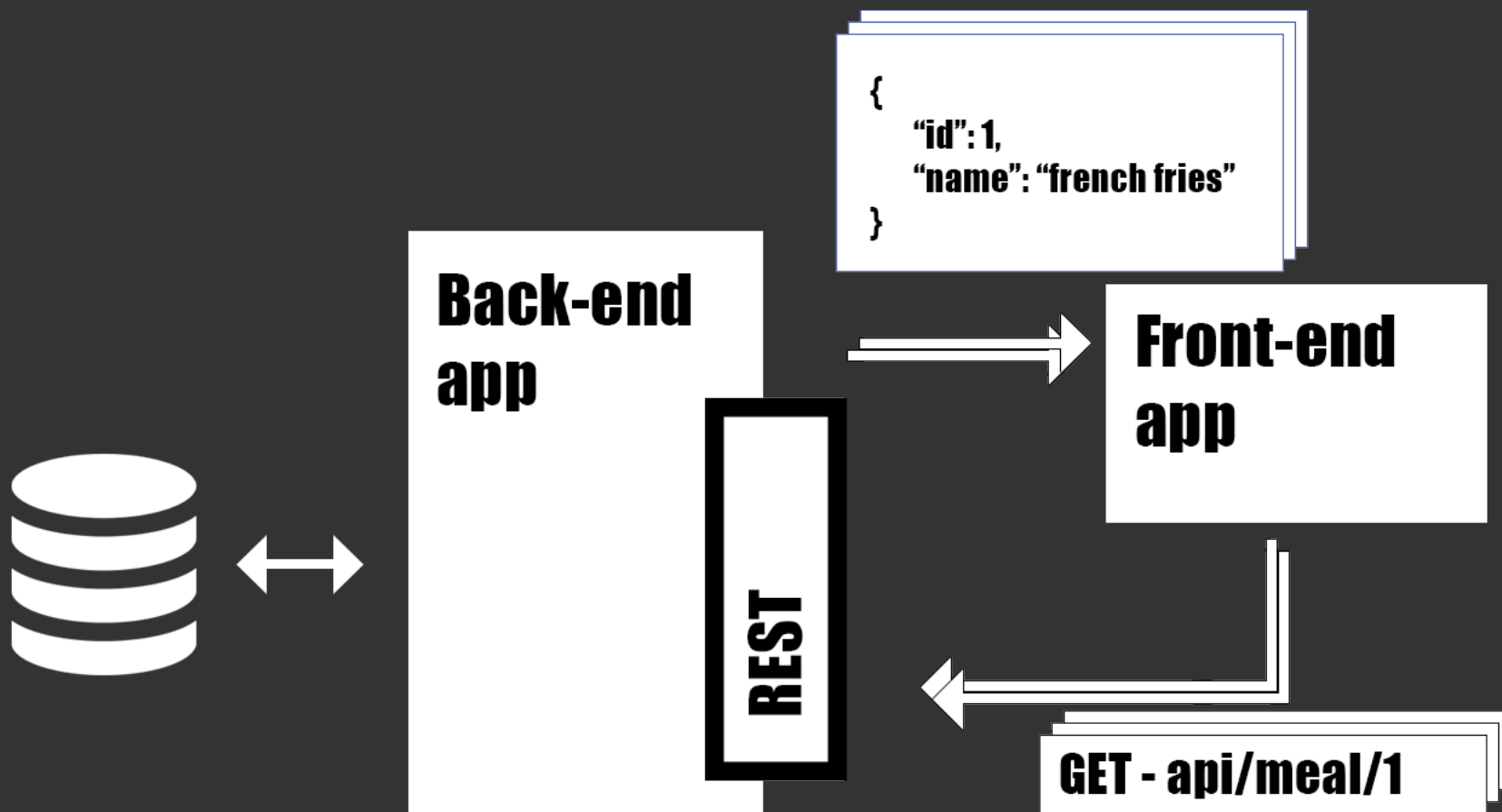
You get only what you **want**, what  
you **need**

# Workflow with GraphQL



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

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





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

# Main problems with the old way

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

# 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"  
      }  
    }  
  }  
}
```

**Over-fetching**

# ~~Over-fetching~~\*

\* It also fixes the issue with the API versioning

# Documentation

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

# Documentation

< RedditSubreddit **RedditLink** ×

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 @clayallsopp",
    "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 RUD of CRUD)
- query aliases
- directives
- GraphiQL – GUI for GraphQL
- Subscription (rfc currently)
- Types and custom Types
- Only one endpoint for everything
- and more

## Sources

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

# Examples with Reddit's GraphQL

## Sources

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

# Demo

## Sources

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

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

# Resolvers & Schema at glance

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

# Resolvers & Schema at glance

Resolvers are functions connected to your backend / api. 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”



**Questions ?**



# 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>