jnesis

SENCHACOMMUNITYDAYS 19

GRAPHQL : REST IN PEACE?

Thomas SIMON

Sencha Community Days
2019
European Sencha ExtJS Conference

jnesis

SENCHACOMMUNITYDAYS 19

NEW ZEALAND

USA

AUSTRIA

THOMAS SIMON

AGE : 33

SOFTWARE ENGINEER SINCE 2015

HOBBIES : TRAVELLING, BASKET-BALL

JNESIS

MULHOUSE, FRANCE

SENCHACOMMUNITYDAYS 19

jnesis

Java & Javascript specialist

Agile team of experts consultants

Sencha partner for France since 2013

Consulting

Development

Training

jnesis

SENCHACOMMUNITYDAYS 19

AGENDA

- Small reminder
- GraphQL
- ExtJS & GraphQL
- Wrapping up

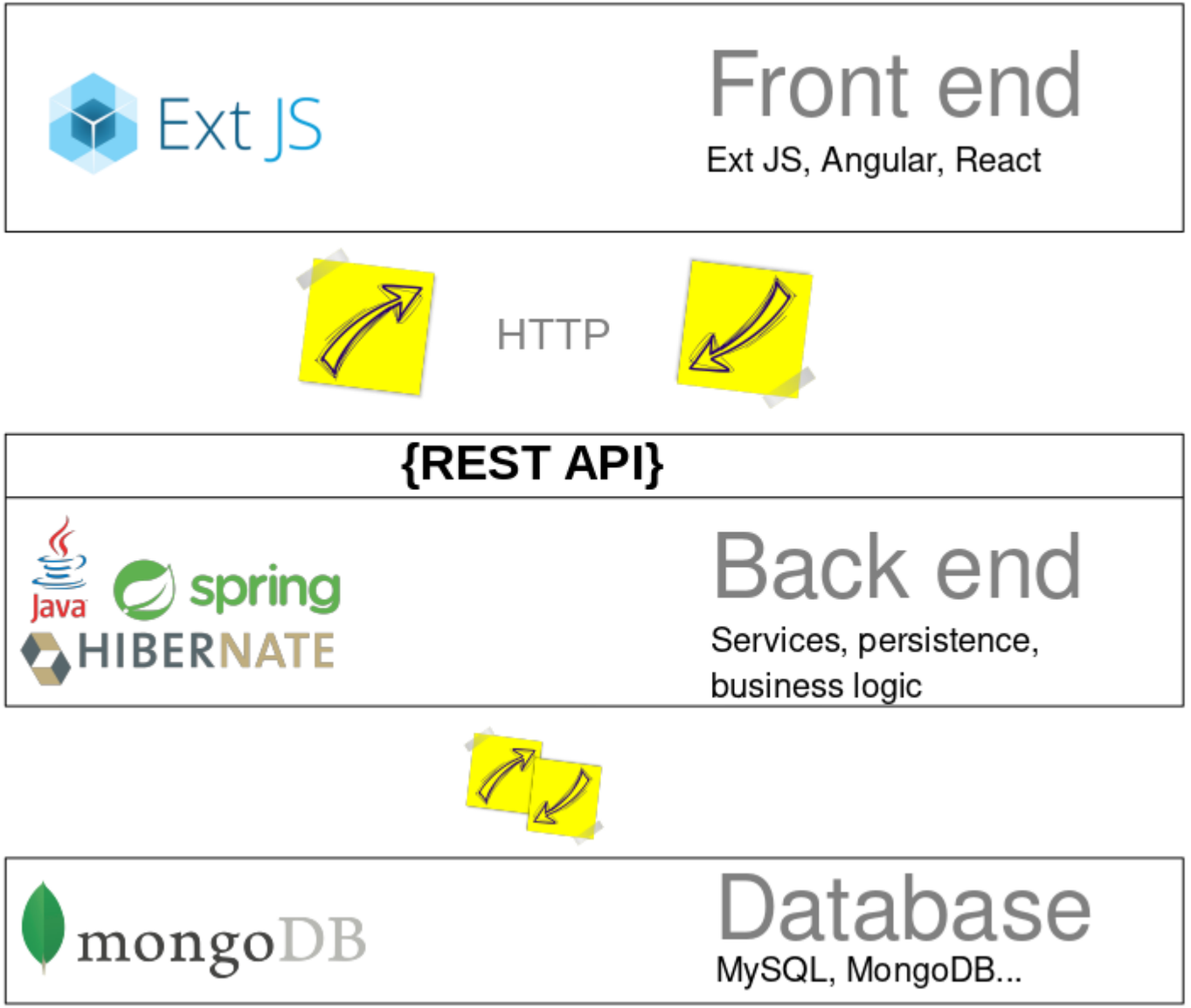
jnesis

SENCHACOM MUNITYDAYS 19

FULL STACK APPLICATION

jnesis

SENCHACOMMUNITYDAYS 19



jnesis

SENCHACOMMUNITYDAYS 19

REST

jnesis

SENCHACOMMUNITYDAYS 19

REST

Representational State Transfer

Architecture style described in 2000

Takes advantages of any protocol

Based on 6 principles

REST PRINCIPLES

- Client - server
- Stateless
- Cacheable
- Layered system
- Uniform interface
- Code on demand (optional)

Key abstraction of information : resources

- URL to identify resources
- HTTP method to identify operation on resources
- HTTP response to view resources


1
HTTP GET
{
"user":{ … }
}
/users/87
/users/87/posts
/users/87/followers
2
HTTP GET
{
"posts" : { … }
}
/users/87
/users/87/posts
/users/87/followers
3
HTTP GET
{
"followers":{ … }
}
/users/87
/users/87/posts
/users/87/followers

jnesis

SENCHACOMMUNITYDAYS 19

WHY CHANGE ?

Easy to use

One of the most popular API

Widely used

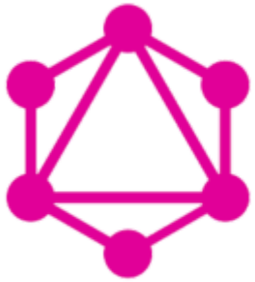
Proven architecture

“Oldies but goodies.”

jnesis

SENCHACOMMUNITYDAYS 19

GRAPHQL



GraphQL

- Created by Facebook in 2012
- Open sourced in 2015
- Facebook‘s native app
- Reduce transmitted informations
- Focused on front-end development

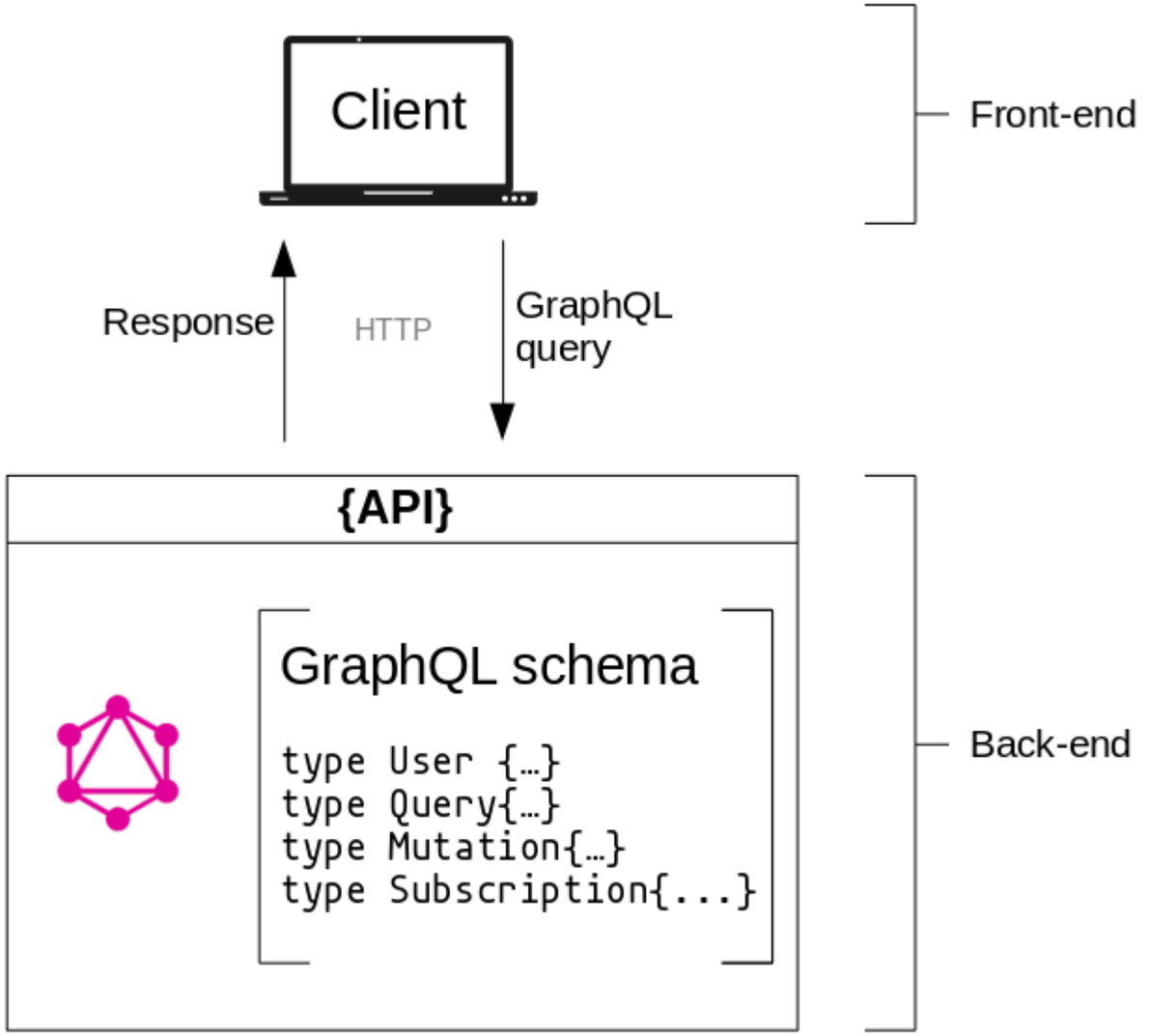
jnesis

SENCHACOMMUNITYDAYS 19

WHAT IS GRAPHQL?

- Query language
- Environment to execute queries
- Main concepts:
  - Hierarchical
  - Strongly typed
  - Defines a data shape

"Ask for what you need, get exactly that"


Client
Front-end
Response
HTTP
GraphQL
query
{API}
GraphQL schema
type User {…}
type Query{…}
type Mutation{…}
type Subscription{...}
Back-end

jnesis

SENCHACOMMUNITYDAYS 19

GRAPHQL SCHEMA

Core of GraphQL server

Data structure

What is possible

Accessible from a single endpoint

TYPE AND FIELD

Main and most basic components

Object you can fetch

```
type User {
        id: ID!
        name: String!
}
```

jnesis

SENCHACOMMUNITYDAYS 19

3 ROOT TYPES

Operations provided by a GraphQL API

jnesis

SENCHACOMMUNITYDAYS 19

QUERY

Fetch data from server

GraphQL schema

```
type Query {
        users: [User!]!
        user(id: ID!): User
}
```

"users" is called a resolver

Query sent by client

```
query {
  users {
        name
  }
}
```

jnesis

SENCHACOMMUNITYDAYS 19

MUTATION

Update data

GraphQL schema

```
type Mutation {
 createUser(name: String!): User!
 updateUser(id: ID!, name: String): User!
}
```

Query sent by client

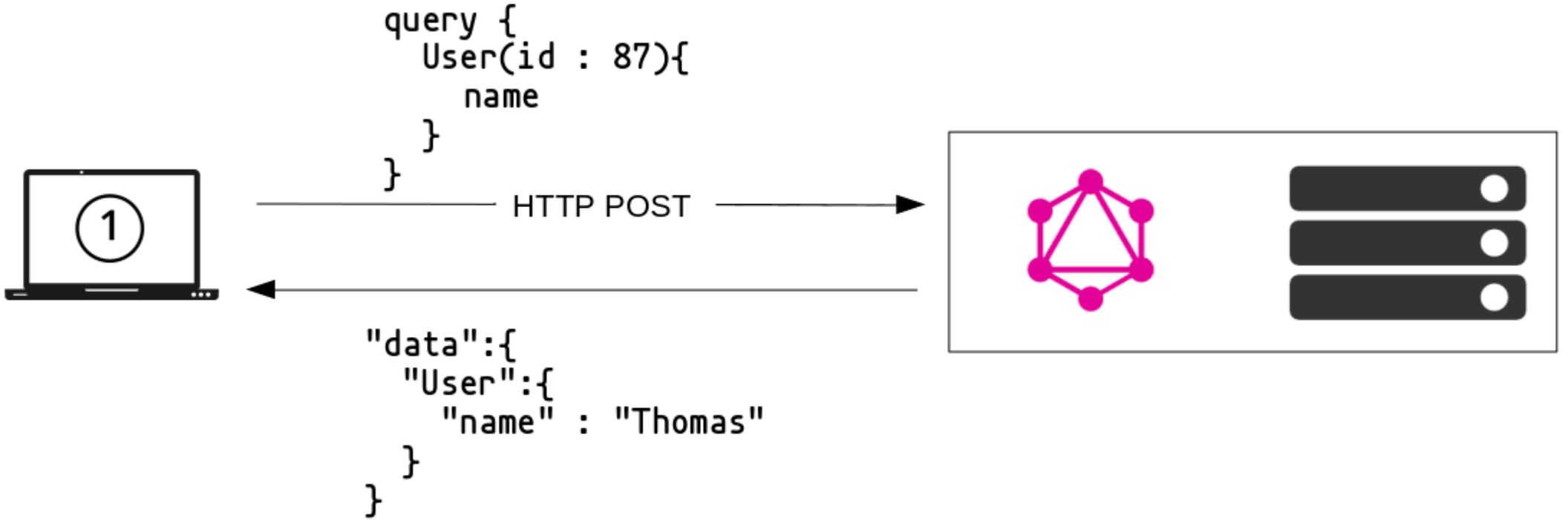
```
mutation {
  createUser(name: "Bob") {
        id
  }
}
```

jnesis

SENCHACOMMUNITYDAYS 19

SUBSCRIPTION

A way to push data from server to the clients


query {
User(id : 87){
name
}
}
1
HTTP POST
"data":{
"User":{
"name" : "Thomas"
}
}

```
query {
  User(id : 87){
    name
    posts {...}
    followers {...}
  }
}
```


```
"data":{
  "User":{
    "name" : "Thomas",
    "posts" : [
         {"title" :"GraphQL is great"}
    ],
    "followers" : [
         {"name" :"John"}
    ]
  }
}
```

jnesis

SENCHACOMMUNITYDAYS 19

EXT JS & GRAPHQL

jnesis

SENCHACOMMUNITYDAYS 19

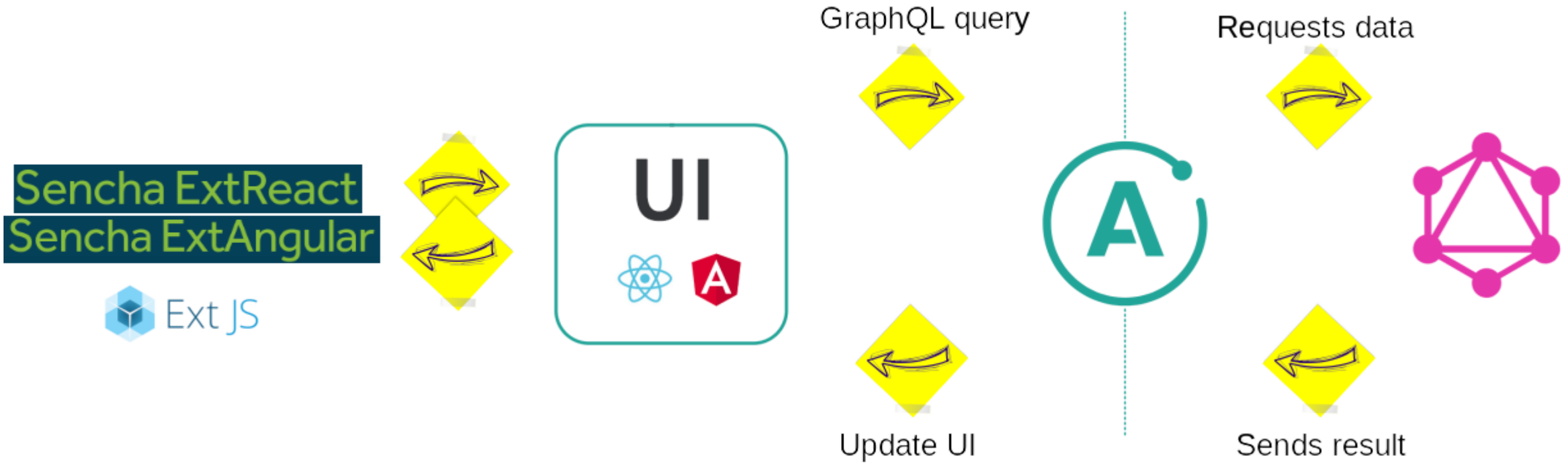
HOW TO USE GRAHQL WITH EXT JS ?

- ExtReact/ExtAngular & Apollo
- Ext JS & existing class
- Ext JS & custom class

jnesis

SENCHACOMMUNITYDAYS 19

EXTREACT/EXTANGULAR



EXTREACT/EXTANGULAR

- Easier to use GraphQL with Apollo

- Complex architecture
- Only ExtJS component

jnesis

SENCHACOMMUNITYDAYS 19

EXT JS & EXISTING CLASS

- Ext.data.proxy.Rest
- Ext.data.proxy.Ajax
- Ext.data.request.Ajax

Use query parameter and write the query

jnesis

SENCHACOMMUNITYDAYS 19

EXAMPLE

```
Ext.Ajax.request({
        url: 'http://localhost:4000/graphql',
        jsonData:{
                query: 'query{links{url,description}}'
        },
        /* ... */
});
```

```
this.getStore('links').load({
        params:{
                query: 'query{links{id,description}}'
        }
});
```

PARAMETRIZE QUERY

```
jsonData:{
        query: 'mutation($url: String!,$desc: String!){post(url:$url,description:$desc){id}}',
        variables: {
                url: 'www.url.com',
                desc: 'Fake url'
        }
}
```

EXT JS & EXISTING CLASS

- Very easy to use

- Difficult to parametrize query
- Very tedious

jnesis

SENCHACOMMUNITYDAYS 19

EXT JS & CUSTOM CLASS

- Extend Ext.data.proxy.Rest
- Extend Ext.data.writer.Writer
- Extend Ext.data.reader.Reader

jnesis

SENCHACOMMUNITYDAYS 19

EXAMPLE

Clone of Hacker News

- List links
- Add links

jnesis

SENCHACOMMUNITYDAYS 19

AUTOMATED PROCESS

```
Ext.define('HackerNews.model.Link', {
        extend: 'HackerNews.model.Base',
        /* Fields : id, url and description */
        proxy: {
                type: 'graphql',
                url: 'http://localhost:4000/graphql',
                //Reader is type 'graphql' by default
                reader: {
                        type: 'graphql',
                        rootProperty: 'data.links'
                },
                //Writer is type 'graphql' by default
                writer: {
                        type: 'graphql'
                }
```

jnesis

SENCHACOMMUNITYDAYS 19

AUTOMATED PROCESS

New class builds queries with default configuration

```
query{links{id,url,description}}
```

```
mutation{
        createLink(url: "Test url",description: "New description"){
                id
                url
                description
        }
}
```

jnesis

SENCHACOMMUNITYDAYS 19

USE CONFIGURATION TO BUILT QUERY

```
proxy: {
        type: 'graphql',
        url: 'http://localhost:4000/graphql',
        reader: {
                type: 'graphql',
                rootProperty: 'data.links',
                readProperties: ['id','url']
        },
        writer: {
                type: 'graphql',
                createResolverName: 'post',
                updateProperties:['url'],
                readProperties: ['id']
        }
}
```

jnesis

SENCHACOMMUNITYDAYS 19

USE CONFIGURATION TO BUILT QUERY

```
query{links{id,url}}
```

```
mutation{
        post(url: "Test url") {
                id
        }
}
```

jnesis

SENCHACOMMUNITYDAYS 19

AUTOMATED PROCESS

Use Ext.data.Model methods

```
let link = new HackerNews.model.Link({
        url: 'www.sencha.com',
        description: 'Nice website'
});
```

```
link.save();
```

```
link.erase();
```

EXT JS & CUSTOM CLASS

- More adaptable
- Possible to automate process
- Parametrize query
- Keep descriptive configuration

- Need time and knowledge to develop class

jnesis

SENCHACOMMUNITYDAYS 19

WRAPPING UP

jnesis

SENCHACOMMUNITYDAYS 19

SHALL WE GIVE UP REST ?

GraphQL is really different than REST

GraphQL is not the new REST

GraphQL is not better than REST

jnesis

SENCHACOMMUNITYDAYS 19

THE RIGHT TOOL FOR THE RIGHT JOB

REST

- CRUD app
- Few fields & always same request type
- Easy caching (server)
- Complex data format
  (no schema constraints)

GRAPHQL

- Reduce amounts of data
- Front-end development in autonomy
- Well-defined rules
- Automatic documentation
- Rich ecosystem/community

jnesis

SENCHACOMMUNITYDAYS 19

THANK YOU FOR YOUR ATTENTION

jnesis

SENCHACOMMUNITYDAYS 19

QUESTION ?!

tsimon@jnesis.com

jnesis

SENCHACOMMUNITYDAYS 19

jnesis jnesis.com/blog

@jnesis_fr

---

fr.linkedin.com/company/jnesis