

Intro a GraphQL

Marzo 2020

¿Qué es?

- LENGUAJE para hacer CONSULTAS y MANIPULACIONES en una API.
- Inventado por Facebook en 2012.
- Publicado en 2015.
<https://graphql.org/>



Graph API Explorer

≡

GET ▾ → / v4.0 ▾ / me?fields=id,name,accounts{id,name}

✕ ★

Submit

- Node: me
- ☒ id
 - ☒ name
 - ☒ accounts
 - ☒ id
 - ☒ name
 - + Search for a field
 - + Search for a field

```
{
  "id": "10152805105073230",
  "name": "Manuel Honorio De la Torre",
  "accounts": {
    "data": [
      {
        "id": "193754729575",
        "name": "gnomoz"
      },
      {
        "id": "470677586367095",
        "name": "VoxFeed"
      },
      {
        "id": "574347745999272",
        "name": "Drap"
      },
      {
        "id": "578676139318864",
        "name": "Real del Carmen Grand 174"
      }
    ]
  },
  "paging": {
    "cursors": {
      "before": "MTkzNzU0NzI5NTc1",
      "after": "NTc4Njc2MTM5MzE4ODY0"
    }
  }
}
```

Access Token

i

EAALJ1p0JJ9oBAJjNOKysFufErizHKkXFVJx3375FXd6NriETTOM

📄

Generate Access Token

Facebook App

VoxFeed - Test ▾

User or Page

User Token ▾

Permissions

i

↶ 🗑

- ✕ user_link
- ✕ read_insights
- ✕ manage_pages
- ✕ pages_show_list
- ✕ instagram_basic
- ✕ instagram_manage_insights
- public_profile

Add a Permission

7 options selected ▾

¿Para qué sirve?

- **QUERIES**
- **MUTACIONES**
- **SUBSCRIPCIONES**

Queries

- Para obtener datos. Ej:

```
query {  
  viewer {  
    id  
    email  
  }  
}
```

Mutations

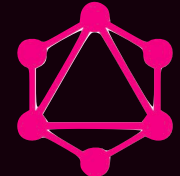
- Para modificar datos. Ej:

```
mutation {  
  updateUser (  
    id: "1"  
    email: "new@email.com"  
  ) {  
    email  
    updatedAt  
  }  
}
```

Subscriptions

- Para subscribirte a eventos. Ej:

```
subscription {  
  requisitionUpdate(id: "1") {  
    balance  
    status  
  }  
}
```



Ventajas

Sólo los datos necesarios

```
GET /users/1
{
  id: "1",
  name: "Fulanito",
  lastName: "Lala",
  buroId: "2",
  balance: 5000,
  ...
  rfc: "XXX123456XXX",
  createdAt: "1/1/19",
  updatedAt: "1/5/20"
}
```

```
user(id: "1") {
  balance

  {
    id: "1",
    balance: 5000
  }
}
```

Consistencia entre “endpoints”

- GET /users
- GET /users/:id
- POST /requisitions/:id/investors
- POST /requisitions/:id/borrower

Reducción de peticiones

```
query {  
  viewer {  
    id  
    email  
    requisitions {  
      amount  
      questions {  
        title  
        answers  
      }  
    }  
  }  
}
```

Reduccion de peticiones

```
query {  
  viewer {  
    id  
    email  
  }  
}  
mutation {  
  lend(requisitionId: "1", amount: 200) { balance }  
  lend(requisitionId: "2", amount: 500) { balance }  
}  
subscription {  
  requisitionCreated { amount }  
}
```

Data y errors

HTTP STATUS: 200 OK

```
{
  data: {
    viewer: { ... }
  },
  errors: [
    { ... },
    { ... }
  ]
}
```

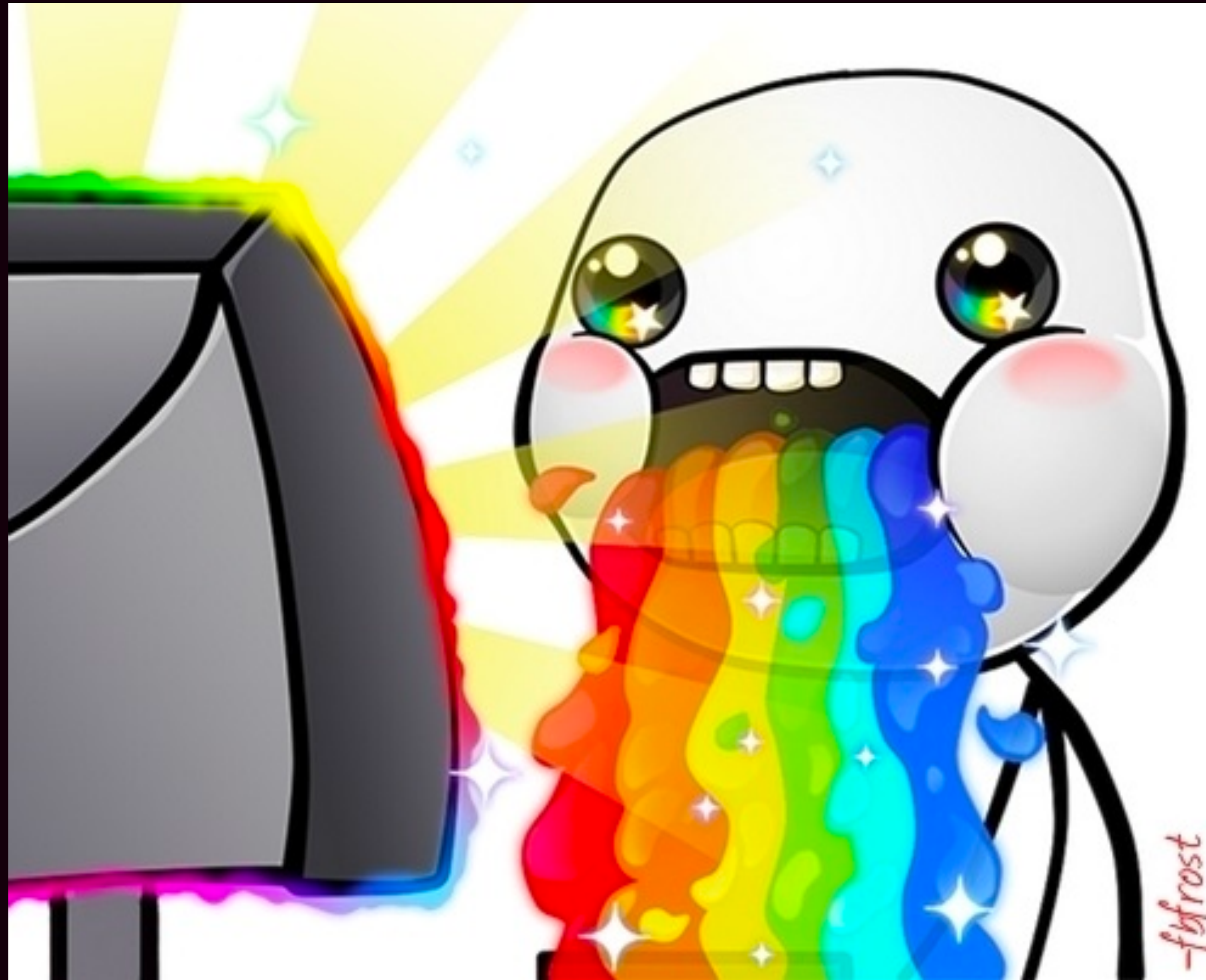
Uso de escalares

```
class Types
  Money < Types::BaseScalar
    description "An amount of money represented in MXN"

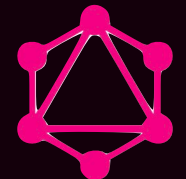
    def self.coerce_input(input_value, context)
      (input_value * 100).round
    end

    def self.coerce_result(ruby_value, context)
      (ruby_value / 100.0).round(2)
    end
  end
end
```


Auto-documentación



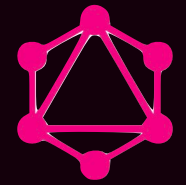
¿Cómo se implementa?



GraphQL



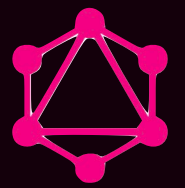
GraphQL Ruby



Queries

api > app > graphql > types >  query_type.rb

```
1  module Types
2    class QueryType < Types::BaseObject
3      field :viewer, UserType,
4        description: 'The currently authenticated User',
5        null: false
6
7      field :states, [StateType],
8        description: 'All valid States for the platform',
9        null: false
10
11     def viewer
12       context[:viewer]
13     end
14
15     def states
16       State.order(:name).all
17     end
18   end
19 end
20
```




Tipos

api > app > graphql > types > user_type.rb

```
1  module Types
2    class UserType < Types::BaseObject
3      describe 'types.user'
4
5      described_field :id, ID, 'user.id', null: false
6      described_field :email, String, 'user.email', null: false
7      described_field :name1, String, 'user.name1', null: true
8      described_field :name2, String, 'user.name2', null: true
9      described_field :names, String, 'user.names', null: true
10     described_field :last_name1, String, 'user.last_name1', null: true
11     described_field :last_name2, String, 'user.last_name2', null: true
12     described_field :birthdate, GraphQL::Types::ISO8601Date, 'user.birthdate', null: true
13     described_field :monthly_income, Integer, 'user.monthly_income', null: true
14     described_field :rfc, String, 'user.rfc', null: true
15     described_field :address, AddressType, 'user.address', null: true
16     described_field :created_at, GraphQL::Types::ISO8601DateTime, 'user.created_at', null: false
17     described_field :updated_at, GraphQL::Types::ISO8601DateTime, 'user.updated_at', null: false
18   end
19 end
20
```

Mutaciones

```
api > app > graphql > types >  mutation_type.rb
```

```
1  module Types
2    ..class MutationType < Types::BaseObject
3    |   ..field :update_email, mutation: Mutations::UpdateEmailMutation
4    |   ..field :update_password, mutation: Mutations::UpdatePasswordMutation
5    |   ..field :update_user, mutation: Mutations::UpdateUserMutation
6    | ..end
7  end
8
```


Mutación

```
api > app > graphql > mutations > update_password_mutation.rb
1  module Mutations
2    class UpdatePasswordMutation < Mutations::BaseMutation
3      describe 'mutations.update_password'
4      null false
5
6      described_argument :current_password, String, 'arguments.current_password', required: true
7      described_argument :new_password, String, 'arguments.new_password', required: true
8
9      described_field :user, Types::UserType, 'types.user', null: false
10
11     def resolve(current_password:, new_password:)
12       result = Users::UpdatePassword.call(
13         user: context[:viewer],
14         current_password: current_password,
15         new_password: new_password
16       )
17
18       resolve_interactor(result, %i[user])
19     end
20   end
21 end
22
```

Endpoints y Autenticación

api > config > routes.rb

```
1  Rails.application.routes.draw do
2    post '/auth/login', to: 'auth#login'
3    post '/auth/logout', to: 'auth#logout'
4    post '/auth/signup', to: 'auth#signup'
5    get  '/auth/email-verification', to: 'auth#request_email_verification'
6    post '/auth/email-verification', to: 'auth#confirm_email_verification'
7    get  '/auth/password-recovery', to: 'auth#request_password_recovery'
8    post '/auth/password-recovery', to: 'auth#confirm_password_recovery'
9    post '/auth/reset-password', to: 'auth#reset_password'
10   post "/graphql", to: "graphql#execute"
11 end
12
```

Uso

Launchpad

GET http://localhost:3000/graphql

GET https://buroserv...

GET http://internal-a...

POST http://localhos...

+

...

No Environment

👁

⚙

POST

http://localhost:3000/graphql

Send

Save

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

Code

☐ none

☒ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	query	{ viewer { id email } }			
	Key		Description		

Body

Cookies

Headers (12)

Test Results

status: 200 OK Time: 33ms Size: 567 B Save Response

Pretty

Raw

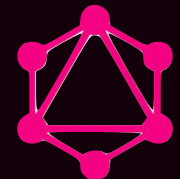
Preview

Visualize

JSON

🔄

```
1 {  
2   "data": {  
3     "viewer": {  
4       "id": "ab59fe7c-fa94-4ba2-959e-062ede35c254",  
5       "email": "user1@zenfi.mx"  
6     }  
7   }  
8 }
```



Downsides

Repetición de consultas

```
query {  
  books(limit: 100) {  
    author  
  }  
}
```

¿Y si los 100
libros son del
mismo autor?

DoS Attack

```
query {  
  books {  
    author {  
      books {  
        author {  
          books {  
            ...  
          }  
        }  
      }  
    }  
  }  
}
```

Thanks !



<https://github.com/manuelmhtr/talks>