THE DEVELOPER'S CONFERENCE

# Trilha – Ruby

**Sergio Lima**
Ruby Developer

THE DEVELOPER'S CONFERENCE

# Uma App Ruby On Rails Integrada com GraphQL

## A Linguagem do Facebook para APIs

por **SERGIO LIMA**
RUBY DEVELOPER

THE DEVELOPER'S CONFERENCE

**Autor da Palestra:**
**Sergio Lima**

**Systems Analyst / Developer**

/sergiosouzalima

**#rubydev.rb**

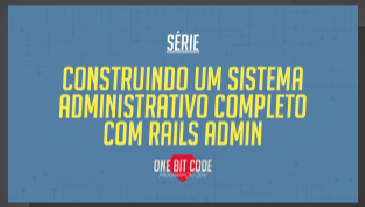
@sergiosouzalima

**Escritor no OneBitCode**

onebitcode.com/author/sergio-lima

# OneBitCode • Full Stack School

## APRENDENDO NA **PRÁTICA** COM **PROJETOS REAIS**









*e muito mais!*

ONEBITCODE FULL STACK SCHOOL | https://onebitcode.com

- Curso de Ruby On Rails.
- Imersão (6 meses).
- Serão criadas 10 apps.
- Websites,  APIs, chatbots.
- Tecnologias Mobile.
- Frameworks Javascript:
  - React, Angular, VueJS.
- Aulas práticas e conceitos em vídeos e textos.


**Bootcamp Super Full Stack**

**http://onebitcode.com**

# Agenda

THE DEVELOPER'S CONFERENCE

- Objetivo da Palestra.
- Minha experiência com GraphQL.
- Rever os termos REST e API.
- GraphQL, o que é isso?
- Como funciona?
- Que problemas o GraphQL resolve?
- Integração com Ruby On Rails.
- Perguntas?

THE DEVELOPER'S CONFERENCE

# Objetivo da Palestra.

# Objetivo da Palestra

THE DEVELOPER'S CONFERENCE

- Explicar o essencial e justificar a utilização do GraphQL.

# Objetivo da Palestra

THE DEVELOPER'S CONFERENCE

- Explicar o essencial e justificar a utilização do GraphQL.

- Mostrar, passo a passo, a integração do GraphQL numa aplicação Ruby On Rails.

THE
DEVELOPER'S
CONFERENCE

# Minha Experiência

# com GraphQL.

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Sistema de transações financeiras.
  - Feito em NodeJS
  - Arquitetura de dados informacional (data-mart)
  - Servidor de dados GraphQL

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Artigo para o blog da Onebitcode
  - How to “Rails x GraphQL”
    - https://onebitcode.com/graphql-introducao/
  - Projeto
    - https://github.com/OneBitCodeBlog/graphqlapp

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Palestra na Guide Investimentos
  - Slides
    - https://www.slideshare.net/sergiosouzalima/graphql-a-linguagem-do-facebook-para-apis-93202036
  - Projeto
    - https://github.com/sergiosouzalima/transactionapp

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Palestra na TDC 2018 São Paulo
  - Slides
    - https://pt.slideshare.net/sergiosouzalima/rails-api-com-graphql
  - Projeto
    - https://notabug.org/sergiosouzalima/graphqlapp
    - tdc-rails-graphql.herokuapp.com/gq
    - tdcapi.herokuapp.com/gq

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Palestra no GURU-SP 2018 São Paulo

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Palestra no GURU-SP 2018 São Paulo
- Talk On Line no OneBitTalk

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Palestra no GURU-SP 2018 São Paulo
- Talk On Line no OneBitTalk
- WorkWeb 2018 – Evento online ao vivo

# Minha Experiência

THE DEVELOPER'S CONFERENCE

- Palestra no GURU-SP 2018 São Paulo
- Talk On Line no OneBitTalk
- WorkWeb 2018 – Evento online ao vivo
- Palestra na TDC 2018 Porto Alegre

THE
DEVELOPER'S
CONFERENCE

# Rever os Termos

# REST e API.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- REST (Representational State Transfer)
  - 2000: surgimento do termo por Roy Fielding.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- REST (Representational State Transfer)
  - 2000: surgimento do termo por Roy Fielding.
  - Estilo de arquitetura de software
    - restrições (constraints) a serem usadas em web services.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- REST (Representational State Transfer)
  - 2000: surgimento do termo por Roy Fielding.
  - Estilo de arquitetura de software
    - restrições (constraints) a serem usadas em web services.
  - Conjunto de melhores práticas e semântica.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- REST (Representational State Transfer)
  - 2000: surgimento do termo por Roy Fielding.
  - Estilo de arquitetura de software
    - restrições (constraints) a serem usadas em web services.
  - Conjunto de melhores práticas e semântica.
  - Melhor entendimento usando verbos HTTP.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- REST (Representational State Transfer)
  - 2000: surgimento do termo por Roy Fielding.
  - Estilo de arquitetura de software
    - restrições (constraints) a serem usadas em web services.
  - Conjunto de melhores práticas e semântica.
  - Melhor entendimento usando verbos HTTP.
  - RESTful: implementa todas essas características.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- REST

| Operação | Verbo |
| --- | --- |
| Create | POST |
| Read (Retrieve) | GET |
| Update (Modify) | PUT / PATCH |
| Delete (Destroy) | DELETE |

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- API (Application Program Interface)

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- API (Application Program Interface)
- Métodos de comunicação entre computadores.

# Rever os Termos REST e API

THE DEVELOPER'S CONFERENCE

- API (Application Program Interface)
- Métodos de comunicação entre computadores.
- Expõe somente objetos e ações necessários.

# Rever os Termos REST e API


- API (Application Program Interface)
- Métodos de comunicação entre computadores.
- Expõe somente objetos e ações necessários.
- Não possui interface (telas).

# Rever os Termos REST e API

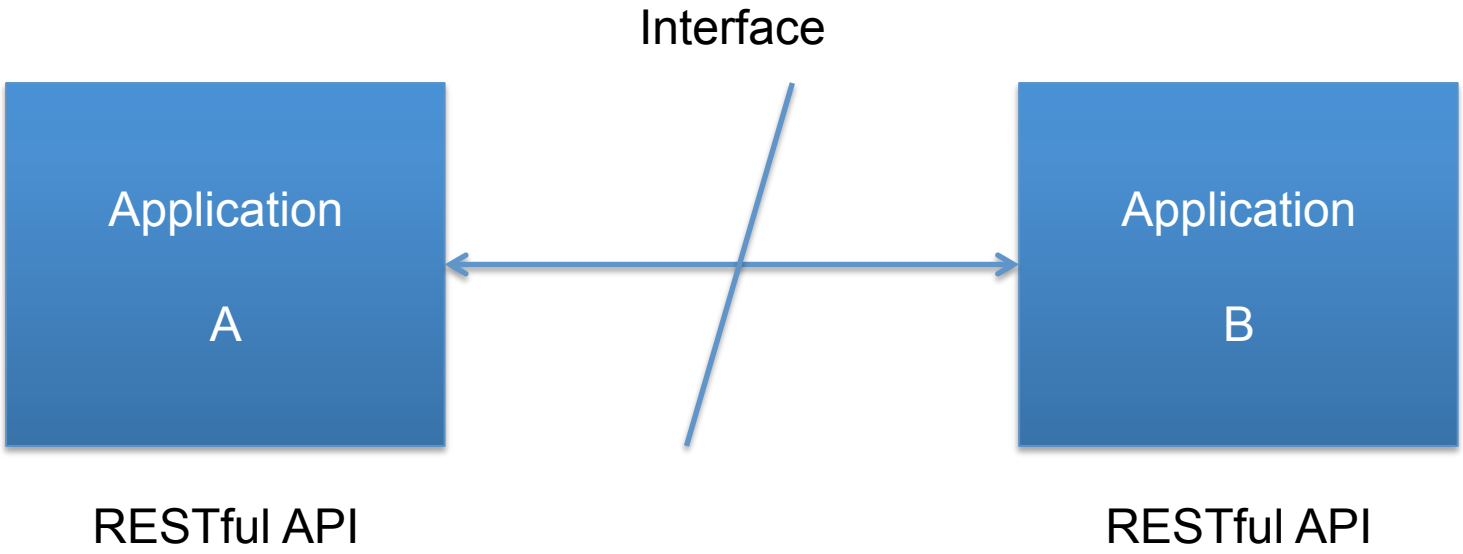
THE DEVELOPER'S CONFERENCE

- API (Application Program Interface)
- Métodos de comunicação entre computadores.
- Expõe somente objetos e ações necessários.
- Não possui interface (telas).
- RESTful APIs.

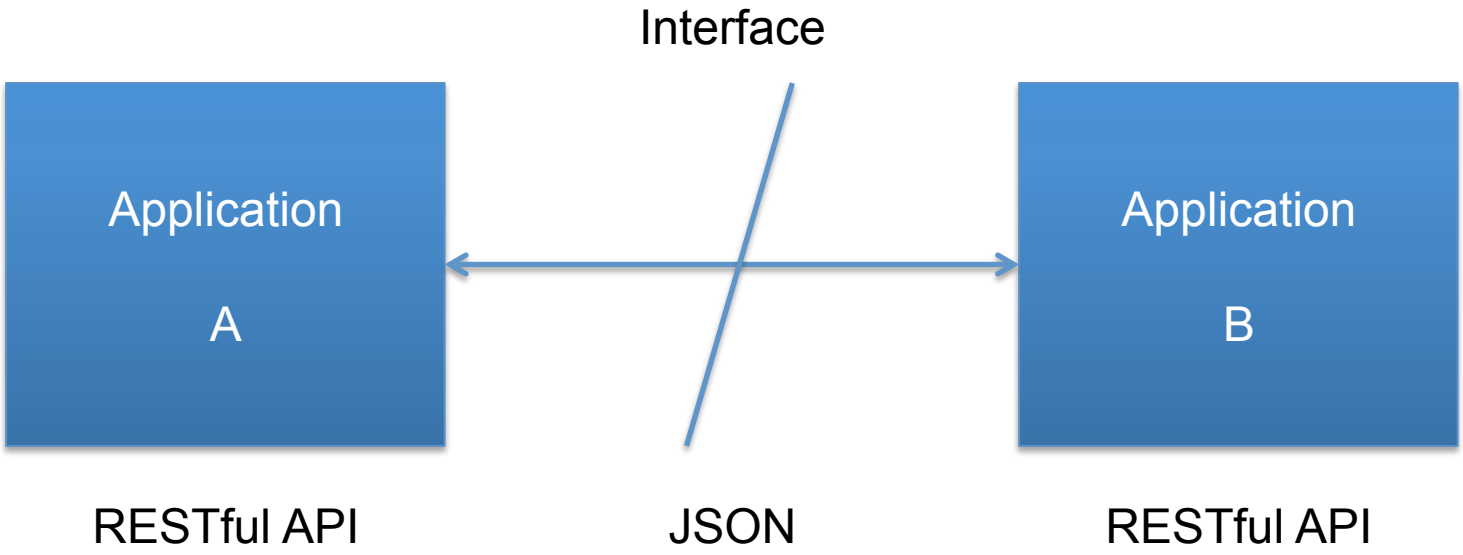
# Rever os Termos REST e API

# Rever os Termos REST e API

# Rever os Termos REST e API



# Rever os Termos REST e API



THE
DEVELOPER'S
CONFERENCE

# GraphQL,

# o que é isso?

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

- Alternativa à maneira tradicional (REST) de se construir APIs.

# GraphQL, o que é isso?


- Alternativa à maneira tradicional (REST) de se construir APIs.
- É algo que colocamos entre um aplicativo front-end e um serviço de dados de back-end.

# GraphQL, o que é isso?


- Alternativa à maneira tradicional (REST) de se construir APIs.
- É algo que colocamos entre um aplicativo front-end e um serviço de dados de back-end.
- Projetada pelos engenheiros do Facebook, 2012.

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

- Alternativa à maneira tradicional (REST) de se construir APIs.
- É algo que colocamos entre um aplicativo front-end e um serviço de dados de back-end.
- Projetada pelos engenheiros do Facebook, 2012.
- Site oficial
  - http://graphql.org
  - “A query language for your API”

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

- Mais que linguagem de consulta

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

- Mais que linguagem de consulta
- Graph*i*QL

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

- Mais que linguagem de consulta
- Graph*i*QL
  - Mecanismo de execução de consulta
  - Feito em React
  - Documentação da API no browser
  - Permite testes

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

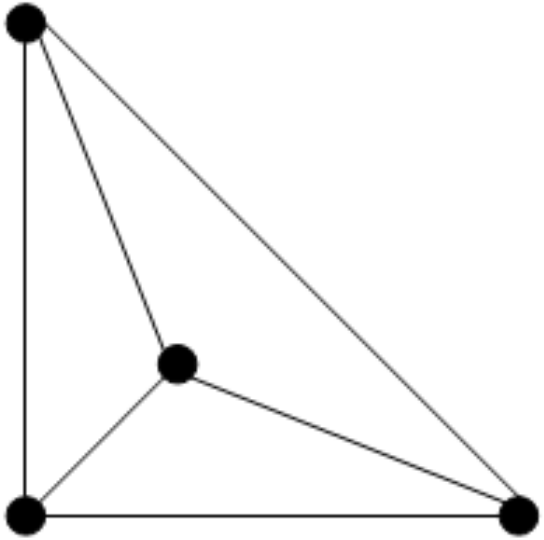
- Mais que linguagem de consulta
- Graph*i*QL
  - Mecanismo de execução de consulta
  - Feito em React
  - Documentação da API no browser
  - Permite testes
- Mas por que esse nome?

# GraphQL, o que é isso?

THE DEVELOPER'S CONFERENCE

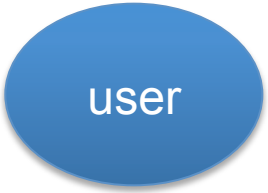
- Mais que linguagem de consulta
- Graph*i*QL
  - Mecanismo de execução de consulta
  - Feito em React
  - Documentação da API no browser
  - Permite testes
- Mas por que esse nome?
  - Especificação de API baseada em grafos.

# GraphQL, o que é isso?

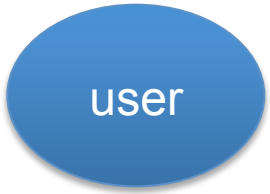


- Cada nó: um recurso do sistema.
- Recurso: usuário, cliente, fornecedor.
- Outro recurso:
  - país do usuário
  - peça fornecida pelo fornecedor
  - nota fiscal do cliente
  - …

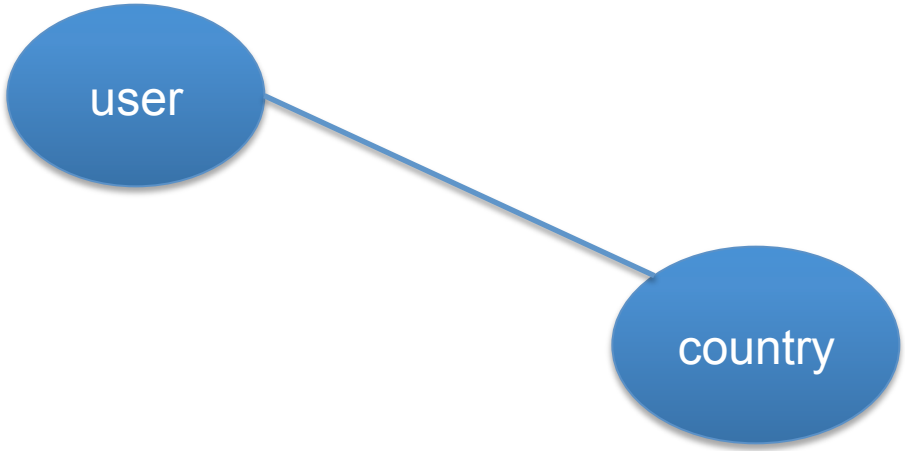
# GraphQL, o que é isso?



# GraphQL, o que é isso?



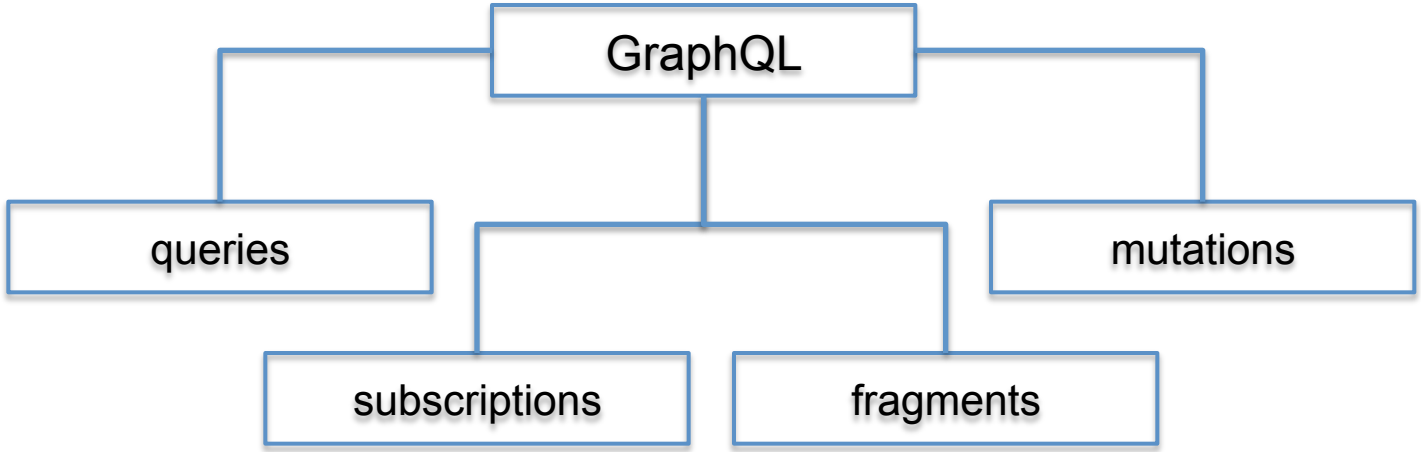

# GraphQL, o que é isso?



THE
DEVELOPER'S
CONFERENCE

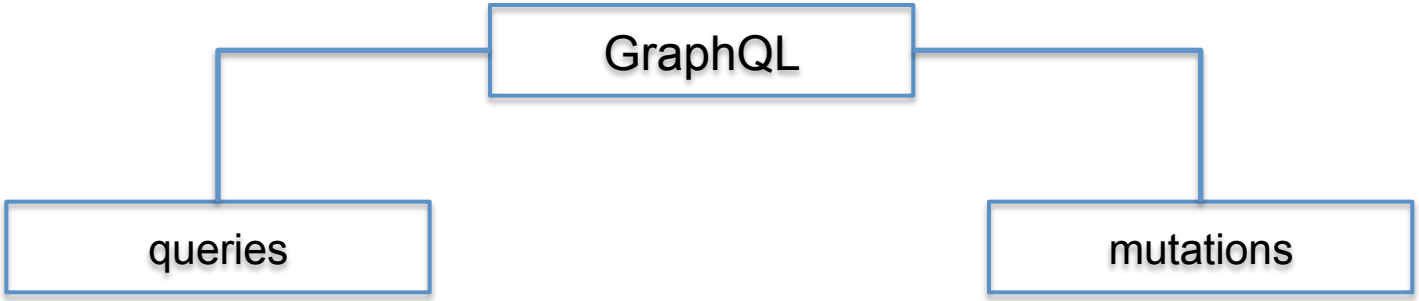
# Como funciona?

# Como funciona?



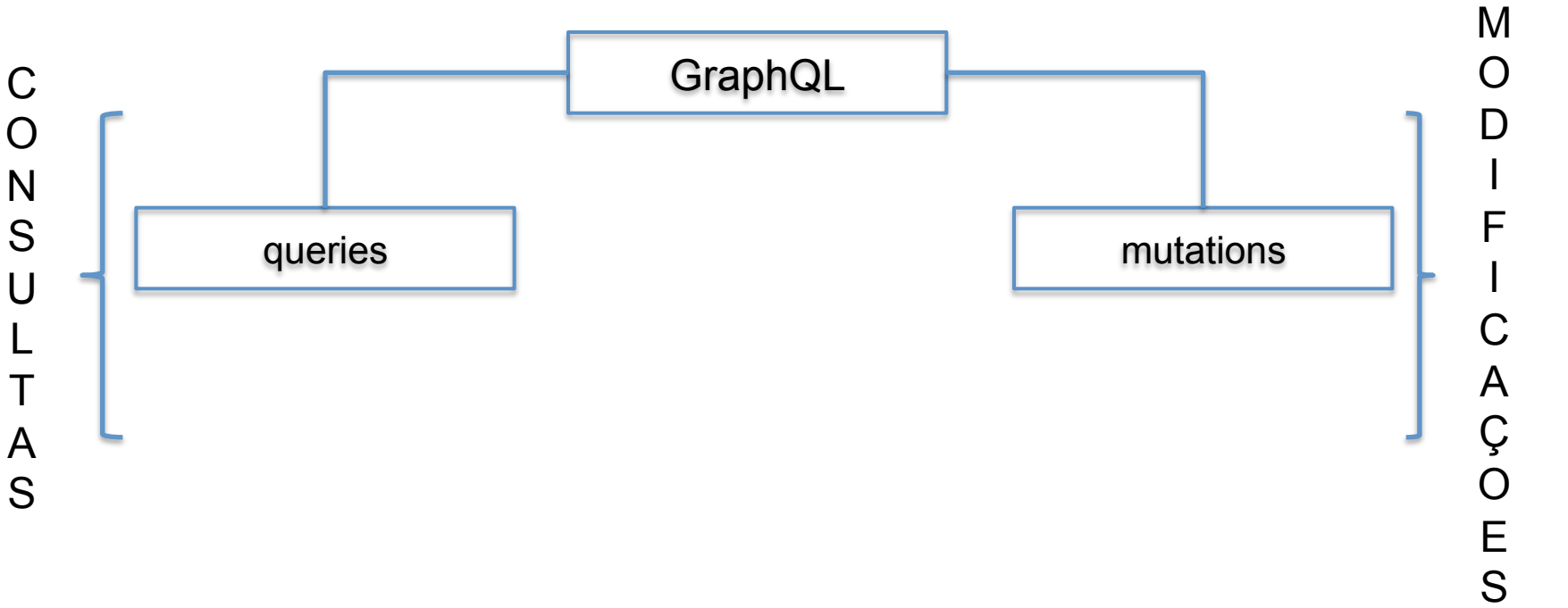
# Como funciona?

THE DEVELOPER'S CONFERENCE

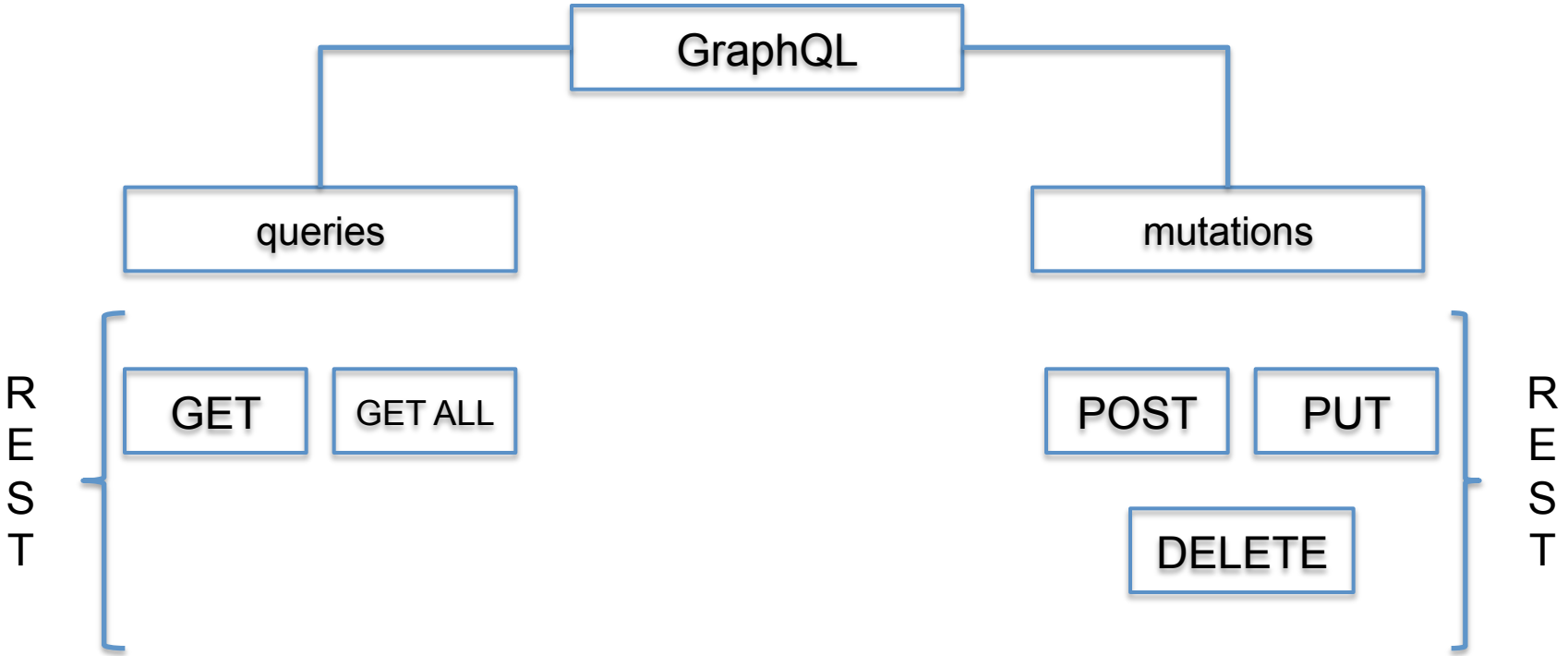


# Como funciona?

THE DEVELOPER'S CONFERENCE

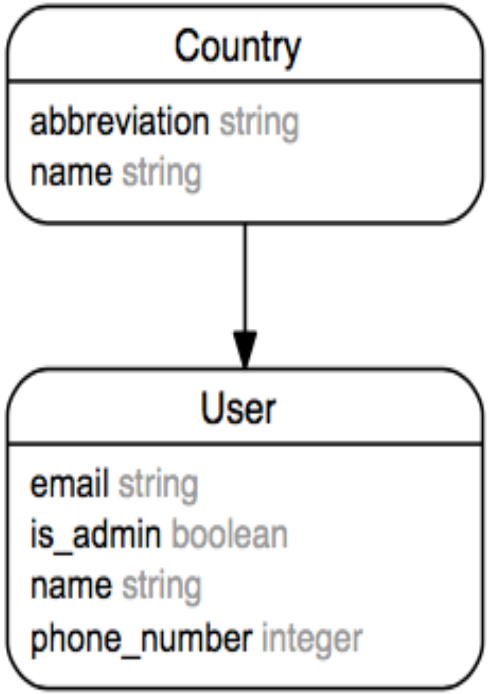


# Como funciona?



# Como funciona?

THE DEVELOPER'S CONFERENCE



# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL |
| --- | --- |
| **GET user?id=1** | `query { user(id: 1) { name email } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL | Resposta da query |
| --- | --- | --- |
| **GET user? id=1** | `query { user(id: 1) { name email } }` | `{ "data": { "user": { "name": "Isaac", "email":"isaac@gmail.com" } } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL |
| --- | --- |
| **GET users** | `query { allUsers { id } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL | Resposta da query |
| --- | --- | --- |
| **GET users** | `query { allUsers { id } }` | `{ "data": { "allUsers": [ { "id": "1" }, { "id": "2" } ] } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL |
| --- | --- |
| **POST users name=Steve&is_admin=true &email=jobs@apple.com&p hone_number=55555555** | `mutation {`<br>`  createUser(name: "Steve",`<br>`  is_admin: true,`<br>`  email:"jobs@apple.com",`<br>`  country_id: 1,`<br>`  phone_number: 55555555) {`<br>`  id`<br>`  }`<br>`}` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL | Resposta da mutation |
|---|---|---|
| POST | `mutation { createUser(name: "Steve", is_admin: true, email:"jobs@apple.com", country_id: 1, phone_number: 55555555) { id } }` | `{ "data": { "createUser": { "id": "9" } } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL |
| --- | --- |
| **PUT users id=9&name='Steve Jobs'** | `mutation { updateUser(id: 9, name: "Steve Jobs") { name } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL | Resposta da mutation |
| --- | --- | --- |
| PUT | `mutation { updateUser(id: 9, name:"Steve Jobs"){ name } }` | `{ "data": { "updateUser": { "name": "Steve Jobs" } } }` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL |
| --- | --- |
| **DELETE users id=9** | `mutation {`<br>`  deleteUser(id: 9)`<br>`}` |

# Como funciona?

THE DEVELOPER'S CONFERENCE

| REST | GraphQL | Resposta da mutation |
| --- | --- | --- |
| **DELETE** | `mutation {`<br>`  deleteUser(id: 9)`<br>`}` | `{`<br>`  "data": {`<br>`  "deleteUser": true`<br>`  }`<br>`}` |

THE DEVELOPER'S CONFERENCE

# Que problemas o GraphQL resolve?

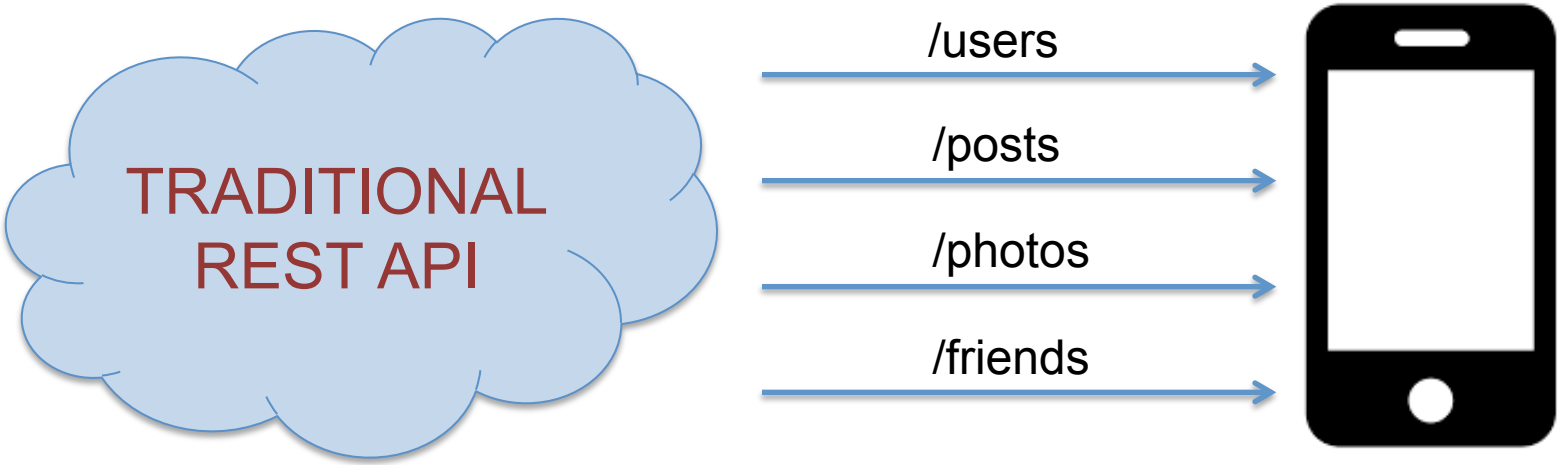
# Que problema resolve?

THE DEVELOPER'S CONFERENCE

- Motivação do Facebook
  - Acessos por conexões de rede lentas
  - A situação se complicava a medida que cresciam os acessos ao Facebook via dispositivos móveis

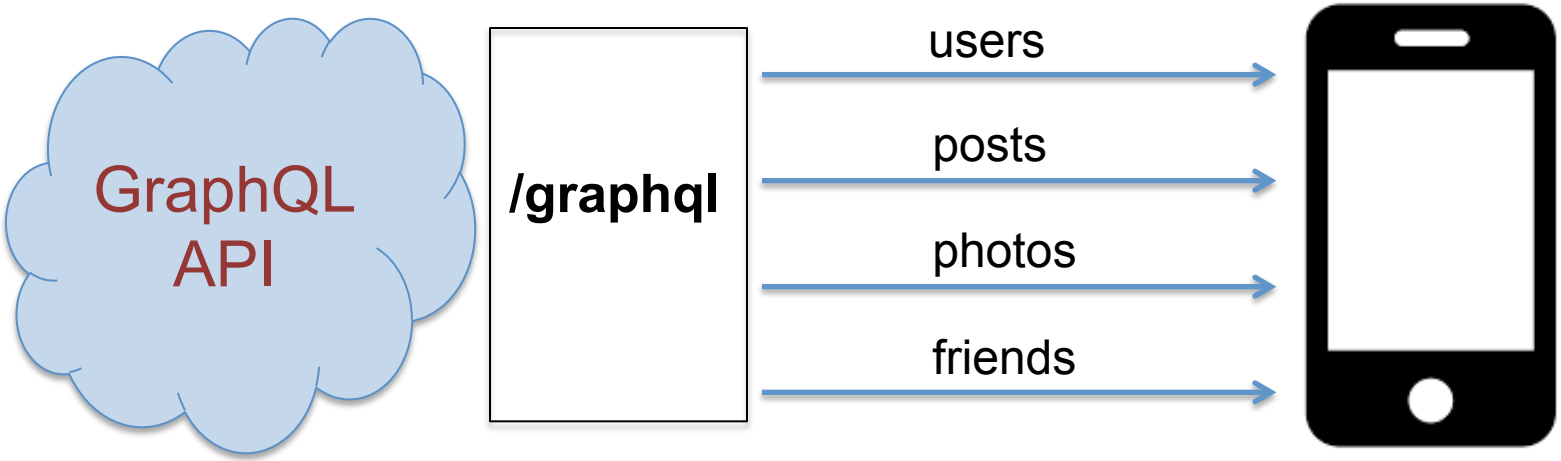
# Que problema resolve?

THE DEVELOPER'S CONFERENCE



# Que problema resolve?

THE DEVELOPER'S CONFERENCE



# Que problema resolve?

THE DEVELOPER'S CONFERENCE

| Problema | Solução |
| --- | --- |
| **Under-fetching**<br>Cliente consumidor de dados recebe menos dados que o necessário. | Apenas uma chamada para vários recursos, recebendo todos os dados necessários. |

# Que problema resolve?

THE DEVELOPER'S CONFERENCE

| Problema | Solução |
| --- | --- |
| **Over-fetching**<br>Cliente consumidor de dados recebe muito mais dados que o necessário. | Receber somente o necessário, assim evita-se demora na consulta e menor processamento no servidor e no cliente. |

# Que problema resolve?

THE DEVELOPER'S CONFERENCE

| Problema | Solução |
| --- | --- |
| **Versionamento da API**<br>APIs tradicionais, com o passar do tempo, acontecem modificação de campos. | Consulta é feita pelo cliente. Ele decide quais dados aparecem na query.<br><br>Sem necessidade de manter versões das APIs. |

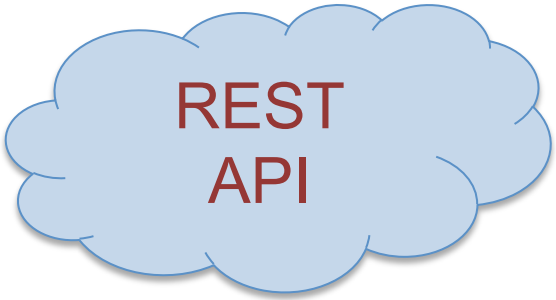
# Que problema resolve?

THE DEVELOPER'S CONFERENCE

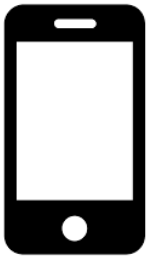
| Problema | Solução |
| --- | --- |
| **Muitas solicitações da equipe de front-end**<br><br>Tempo e esforço de comunicação entre equipes é comum hoje em dia | A equipe de front-end escolhe o que pedir da API.<br><br>A comunicação para este propósito diminuirá, liberando tempo para outras atividades. |

# Que problema resolve?

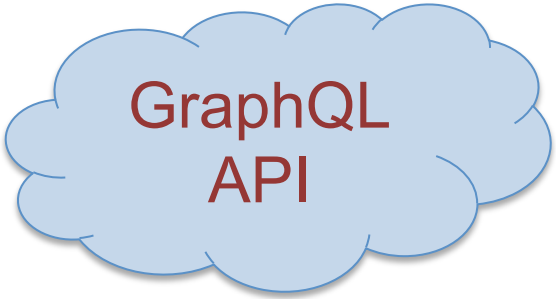
THE DEVELOPER'S CONFERENCE



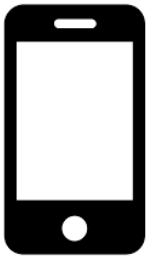

data
requirements
characteristics



client




data
requirements
characteristics



client

THE
DEVELOPER'S
CONFERENCE

# Integração com uma aplicação Ruby On Rails.

# Integração com Rails

THE DEVELOPER'S CONFERENCE

- How to GraphQL integrado com Rails
  - https://onebitcode.com/graphql-introducao/

# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
$ mkdir graphqlapp && cd graphqlapp

$ rvm use ruby-2.5.0@graphqlapp --ruby-version --create

$ gem install rails -v 5.2.0 --no-ri --no-rdoc

$ rails _5.2.0_ new . --api -T
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

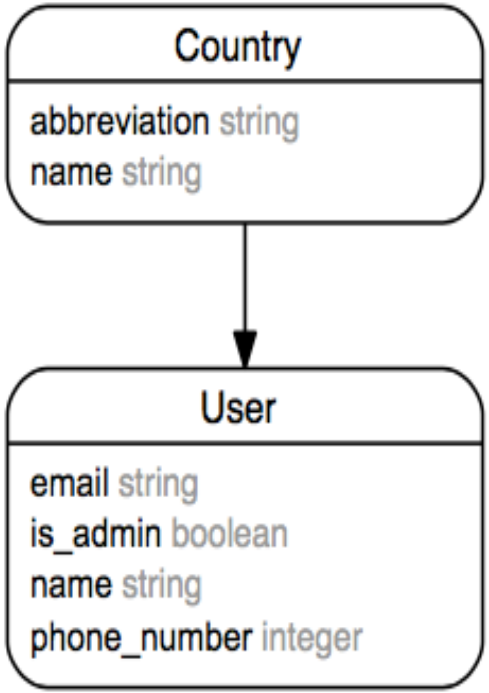
```
# Gemfile
gem 'graphql', '~> 1.7.13’

gem ‘graphiql-rails’, '~> 1.4.10', group: :development
## GraphiQL fornece uma interface
## de consulta via browser para GraphQL.


$ bundle install
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
$ rails generate model
  Country abbreviation name
```

```
$ rails generate model
  User name email phone_number:integer
  country:references is_admin:boolean
```



# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
$ rails db:migrate

$ rails console

 Country.create abbreviation: 'BRA', name: 'BRAZIL'

 Country.create abbreviation: 'USA', name: 'UNITED STATES OF AMERICA'

 User.create name: 'Isaac', email: 'isaac@gmail.com', phone_number: 5556767, country:
Country.first, is_admin: true

 User.create name: 'Stephen', email: 'stephen@yahoo.com', phone_number: 5558989, country:
Country.second, is_admin: false

 User.create name: 'Albert', email: 'albert@outlook.com', phone_number: 5554545, country:
Country.second, is_admin: false

exit
```

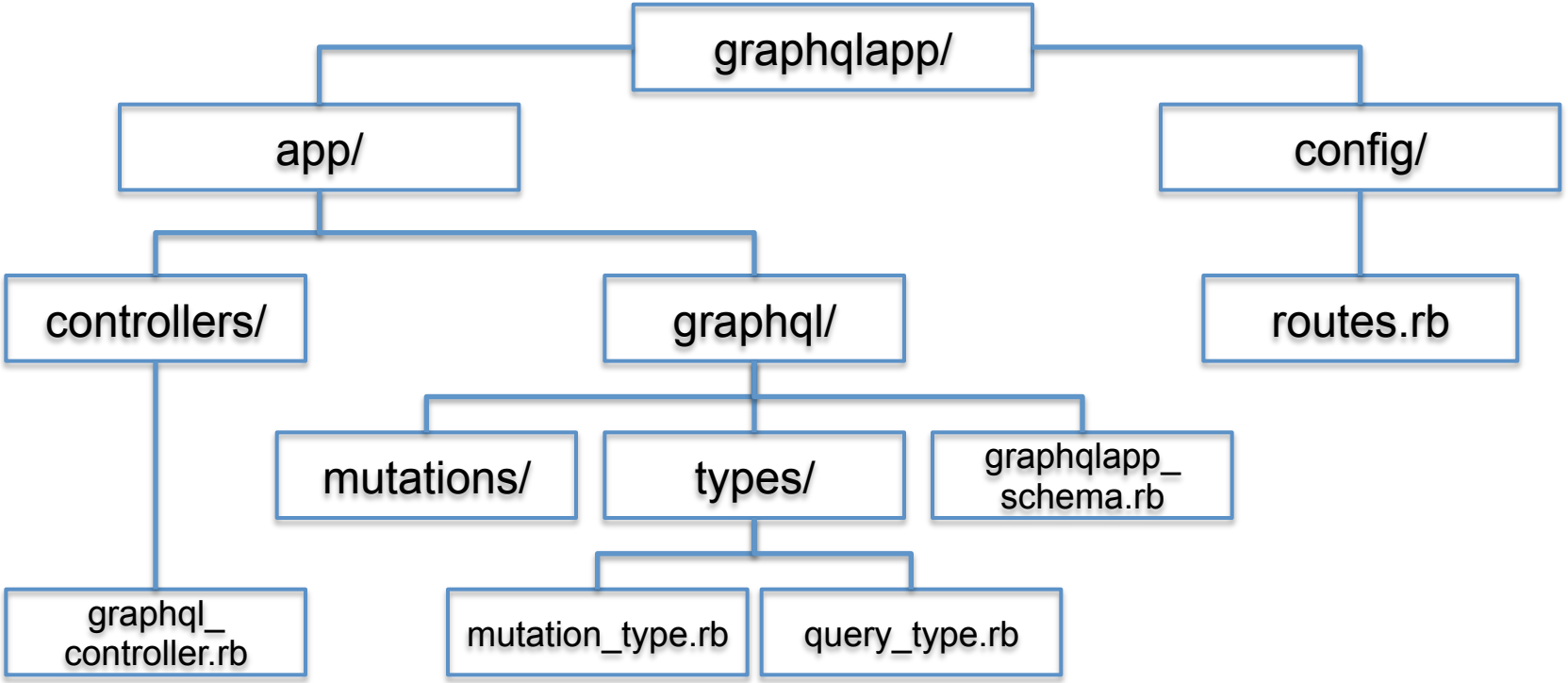
# Integração com Rails

THE DEVELOPER'S CONFERENCE

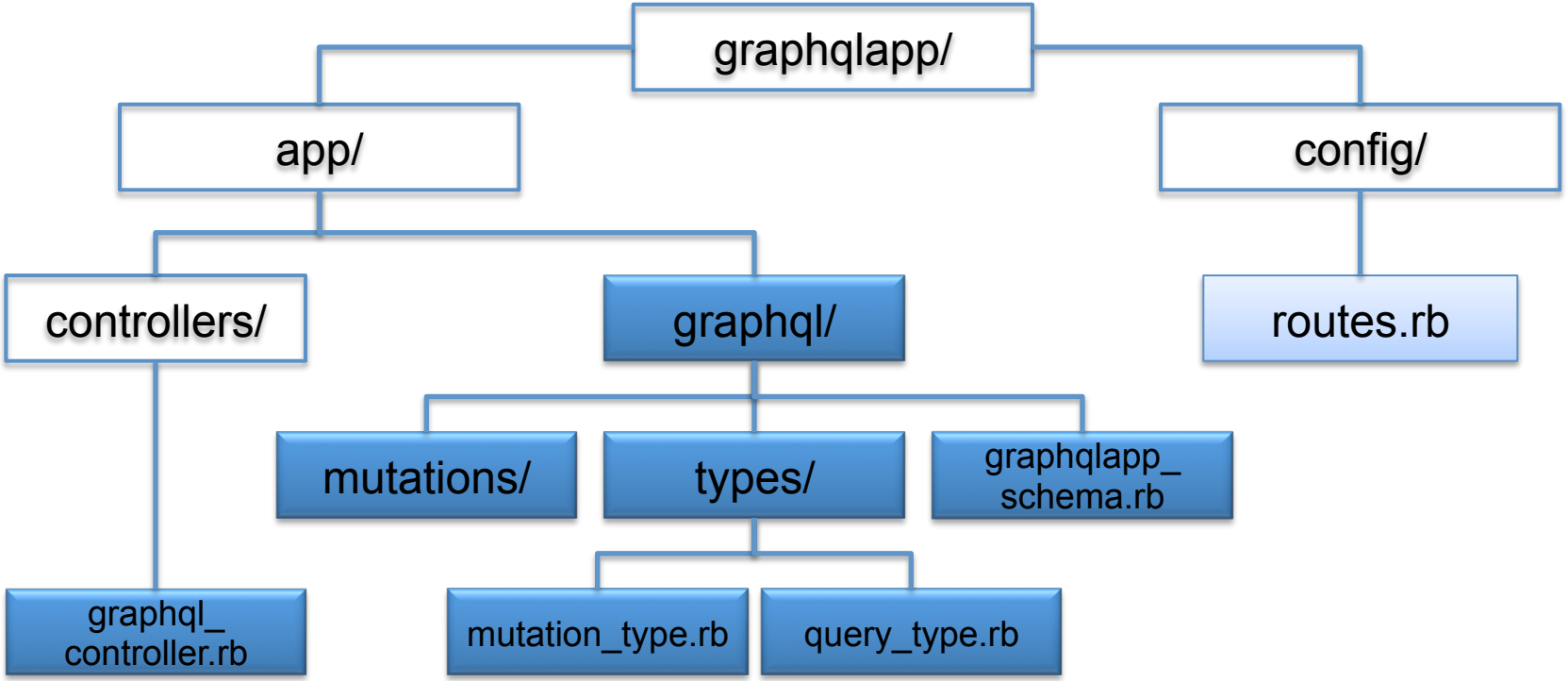
```
$ bundle exec rails generate graphql:install
```

```
# config/application.rb
# retirar o comentário da linha:
require "sprockets/railtie"
```

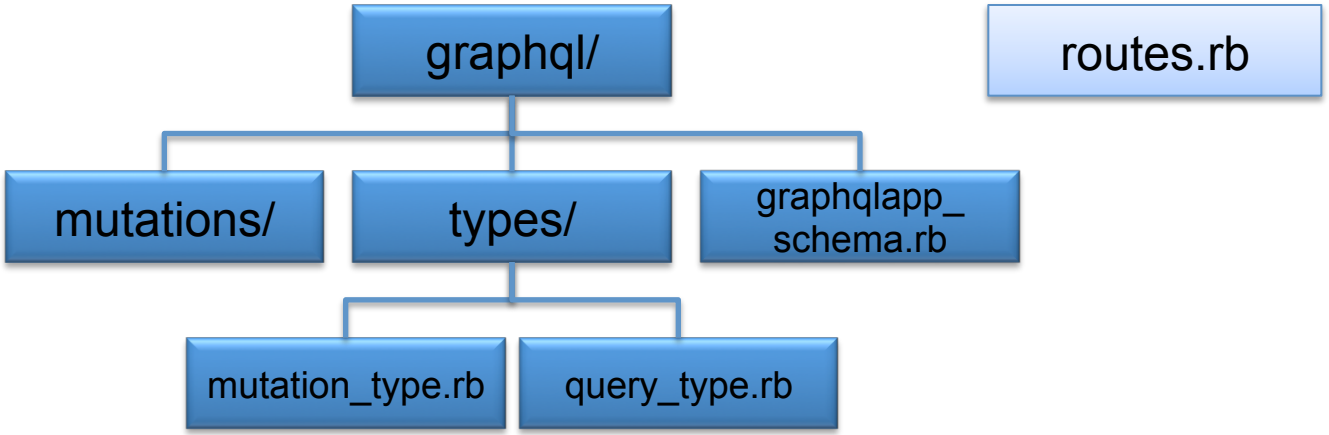
# Integração com Rails



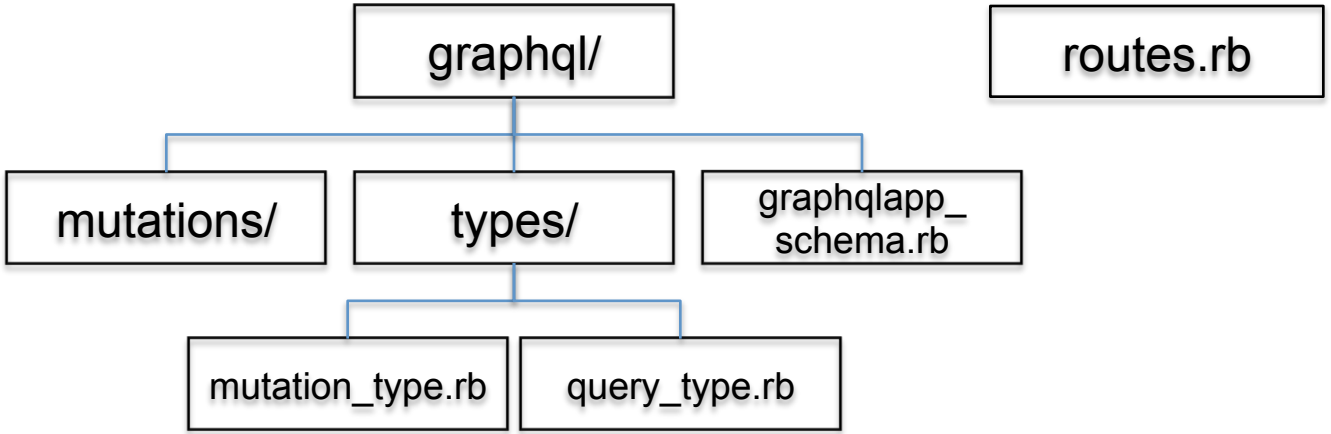
# Integração com Rails



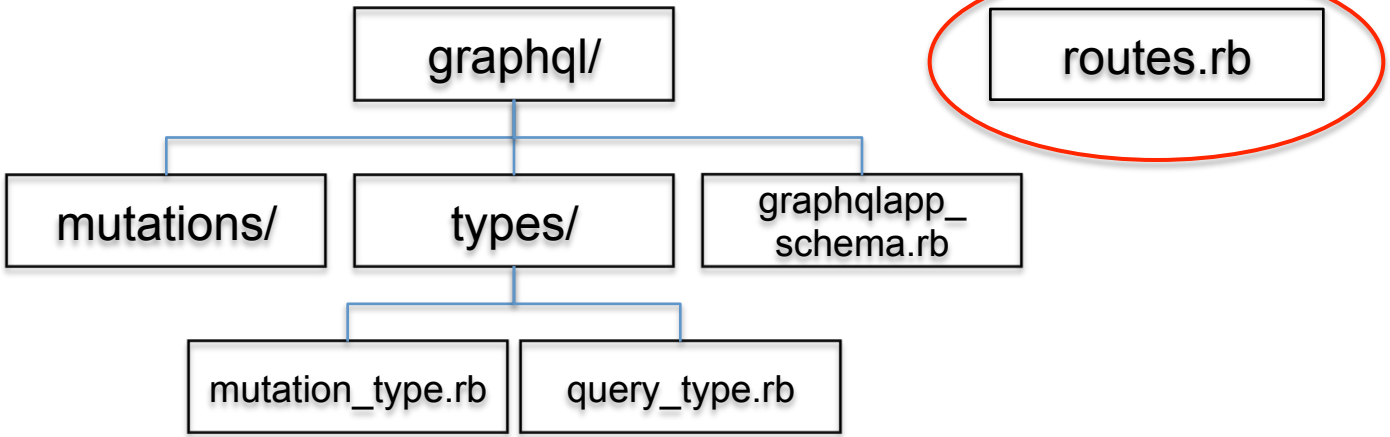
# Integração com Rails



# Integração com Rails



# Integração com Rails



# Integração com Rails

THE DEVELOPER'S CONFERENCE

config/routes.rb

```
01 Rails.application.routes.draw do
02
03   post "/graphql", to: "graphql#execute”
04
05 end
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

config/routes.rb

```
01 Rails.application.routes.draw do
02
03  post "/graphql", to: "graphql#execute”
04
05  mount GraphiQL::Rails::Engine, /
06  at: "/gq", graphql_path: "/graphql” If Rails.env.development?
07
08 end
```

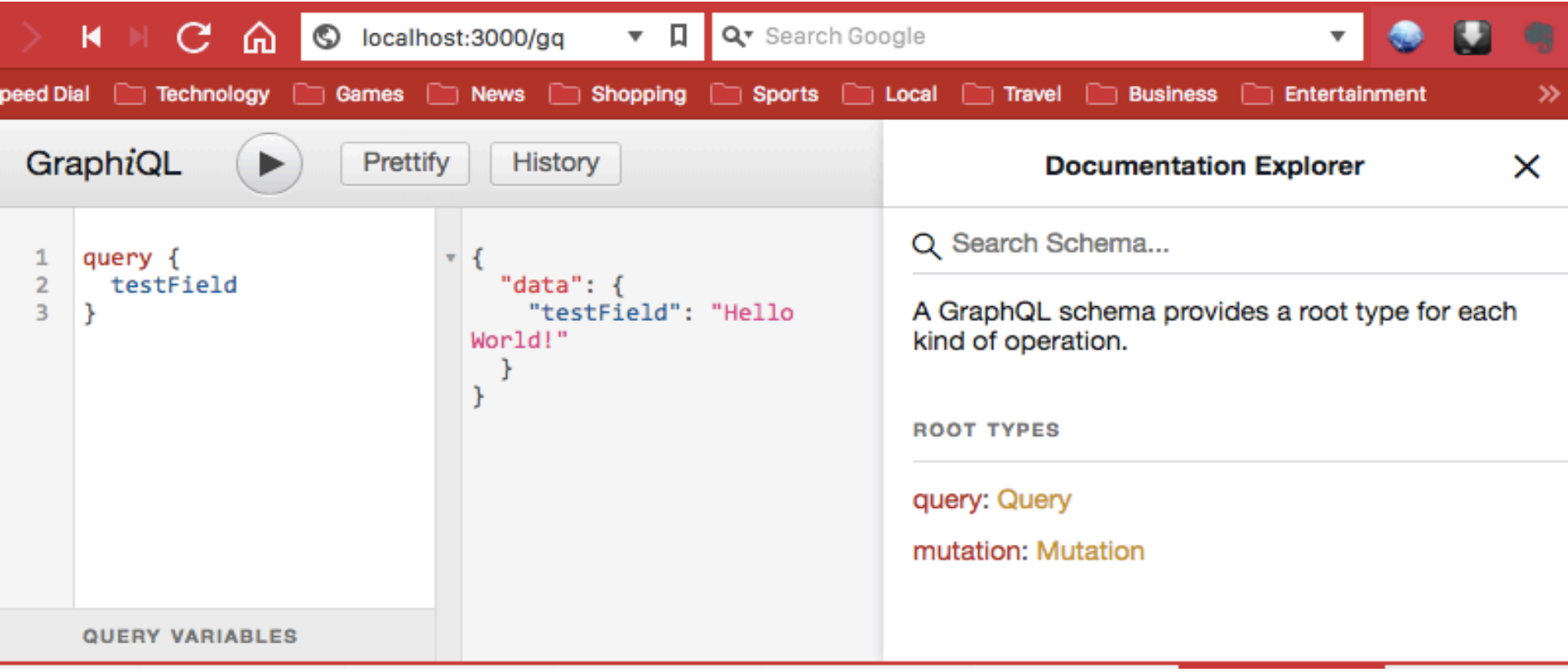
# Integração com Rails

THE DEVELOPER'S CONFERENCE

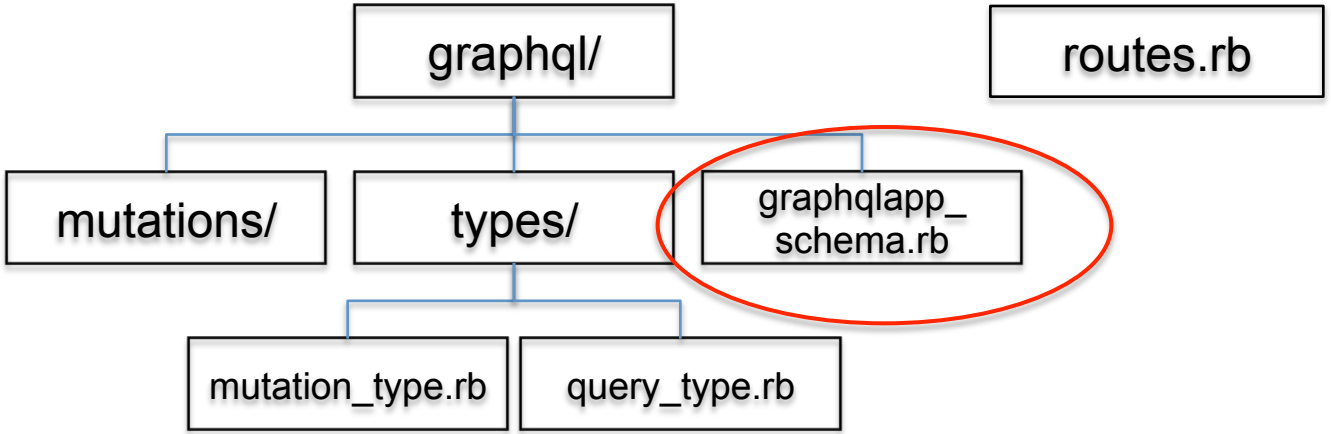
```
$ rails server
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE



# Integração com Rails



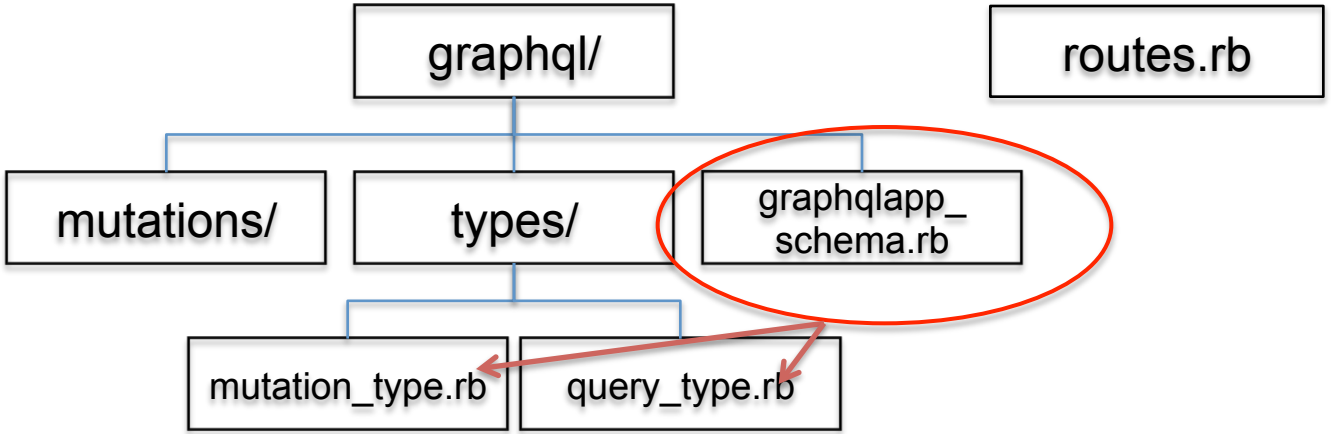
# Integração com Rails

THE DEVELOPER'S CONFERENCE

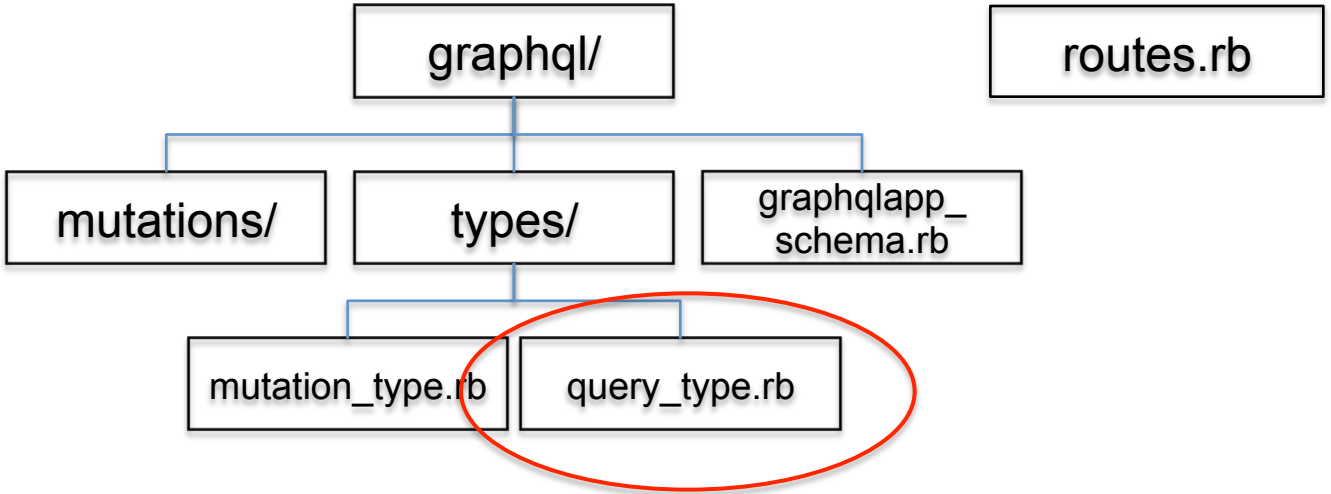
app/graphql/graphqlapp_schema.rb

```
01 GraphqlappSchema = GraphQL::Schema.define do
02
03  mutation(Types::MutationType)
04  query(Types::QueryType)
05
06 end
```

# Integração com Rails



# Integração com Rails



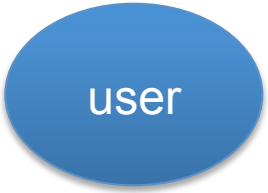
# Integração com Rails

THE DEVELOPER'S CONFERENCE

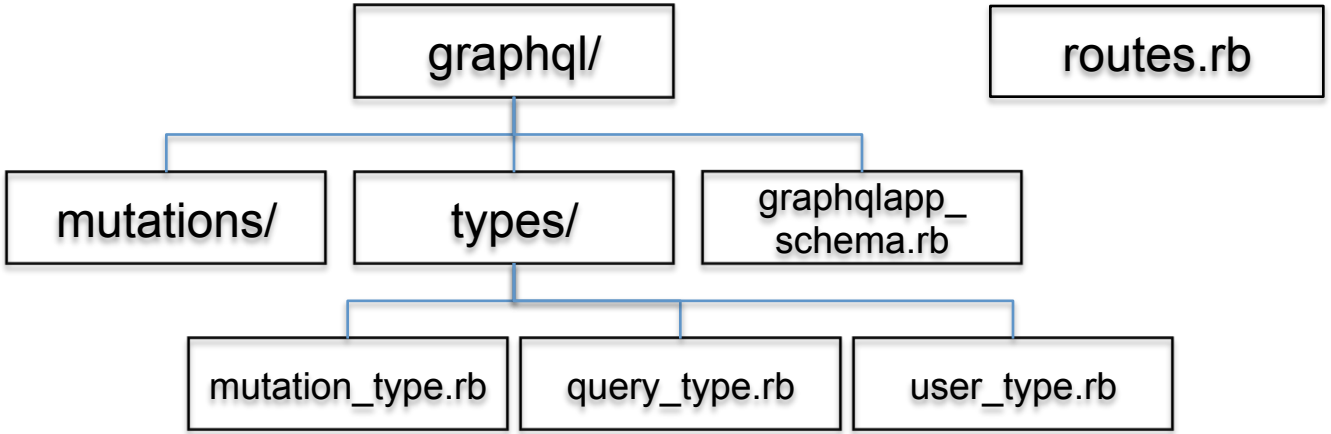
app/graphql/types/query_type.rb

```
01 Types::QueryType = GraphQL::ObjectType.define do
02   name "Query”
03
04   # chamamos o field de "user" e ele terá um namespace chamado "Types" que chamamos
05   # de "UserType" e ficará em app/graphql/types/user_type.rb
06
07   field :user, Types::UserType do
08     # passamos aqui o "id" do User como argumento de consulta
09     # esse valor de "id" vem do front-end para o back-end
10     argument :id, types.ID # o "id" tem um tipo especial chamado ID
11     description "Identificação do Usuário"
12
13     # aqui é o método chamado "resolve" que resgata os dados de User do banco de dados
14     resolve -> (obj, args, ctx) {
15       User.where(id: args[:id]).first
16     }
17   end
18 end
```

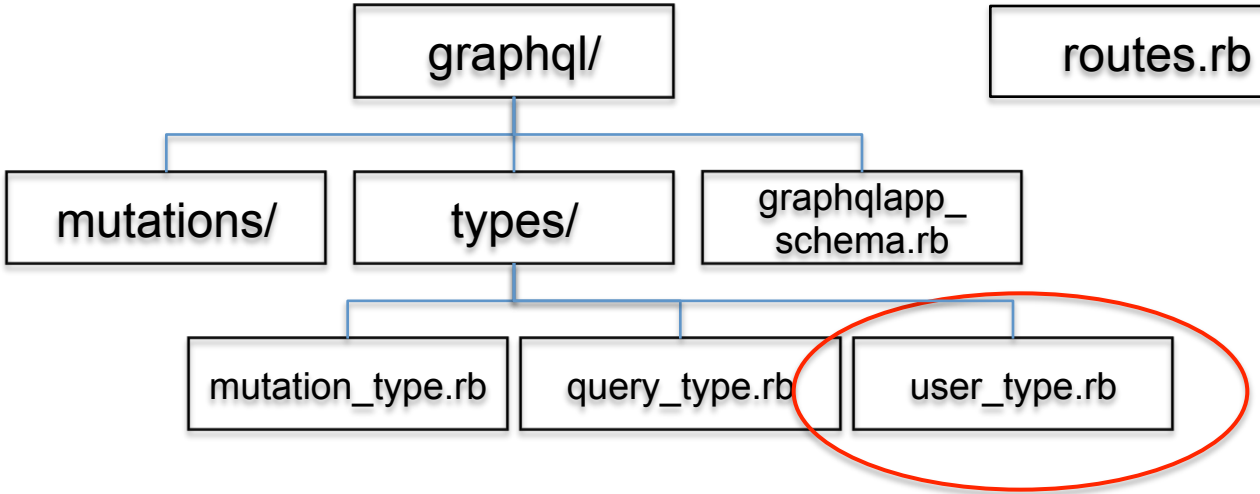
# Integração com Rails



# Integração com Rails



# Integração com Rails



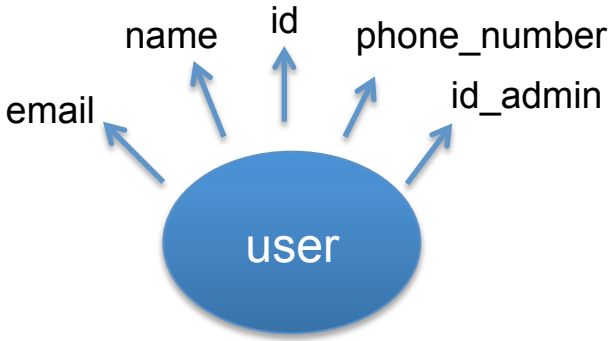
# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/types/user_type.rb

```
01 Types::UserType = GraphQL::ObjectType.define do
02  name "UserType" # É como o schema irá identificar essa type.
03
04  # Nesta parte informamos quais campos serão visíveis numa consulta
05  field :id, types.ID
06  field :name, types.String
07  field :email, types.String
08  field :phone_number, types.Int
09  field :is_admin, types.Boolean, "Usuario administrador?”
10
11
12
13
14
15
16
17
18
19 end
```

# Integração com Rails



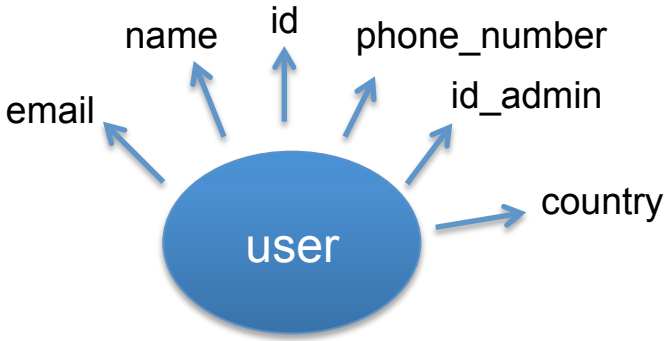
# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/types/user_type.rb

```
01 Types::UserType = GraphQL::ObjectType.define do
02  name "UserType" # É como o schema irá identificar essa type.
03
04  # Nesta parte informamos quais campos serão visíveis numa consulta
05  field :id, types.ID
06  field :name, types.String
07  field :email, types.String
08  field :phone_number, types.Int
09  field :is_admin, types.Boolean, "Usuario administrador?”
10
11
12  field :country, types.Int
13
14
15
16
17
18
19 end
```

# Integração com Rails




THE DEVELOPER'S CONFERENCE

# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/types/user_type.rb

```
01 Types::UserType = GraphQL::ObjectType.define do
02  name "UserType" # É como o schema irá identificar essa type.
03
04  # Nesta parte informamos quais campos serão visíveis numa consulta
05  field :id, types.ID
06  field :name, types.String
07  field :email, types.String
08  field :phone_number, types.Int
09  field :is_admin, types.Boolean, "Usuario administrador?”
10
11
12  field :country, types.Int
13
14
15
16
17
18
19 end
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/types/user_type.rb

```
01 Types::UserType = GraphQL::ObjectType.define do
02  name "UserType" # É como o schema irá identificar essa type.
03
04  # Nesta parte informamos quais campos serão visíveis numa consulta
05  field :id, types.ID
06  field :name, types.String
07  field :email, types.String
08  field :phone_number, types.Int
09  field :is_admin, types.Boolean, "Usuario administrador?”
10
11  # Vamos expor, todos dados do País que estão no relacionamento “user.country”
12  field :country do
13    type Types::CountryType
14    description "País associado a este usuário"
15    resolve ->(user, args, context){
16      user.country
17    }
18  end
19 end
```

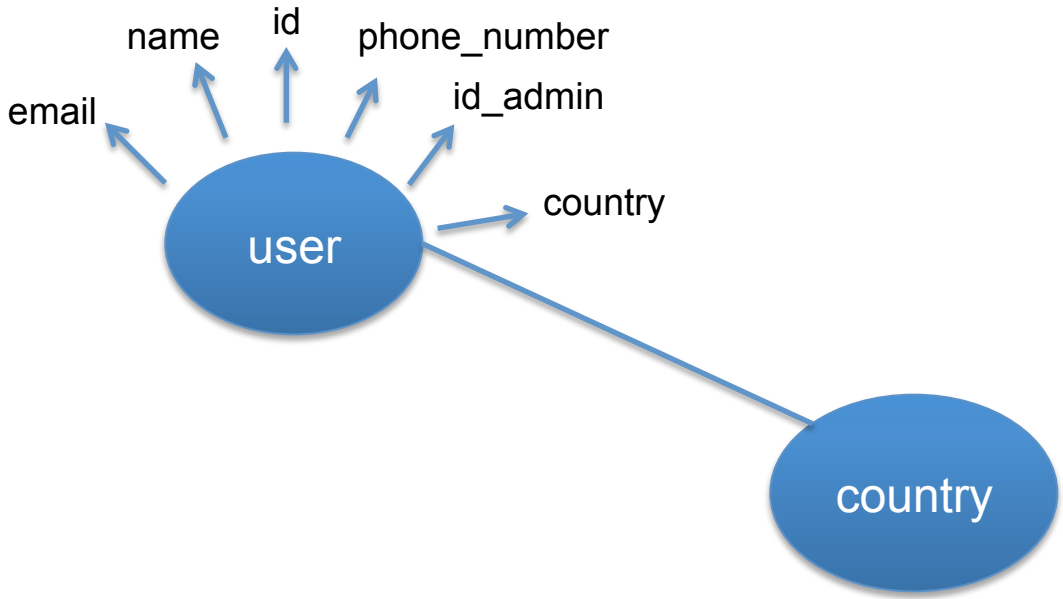
# Integração com Rails

THE DEVELOPER'S CONFERENCE

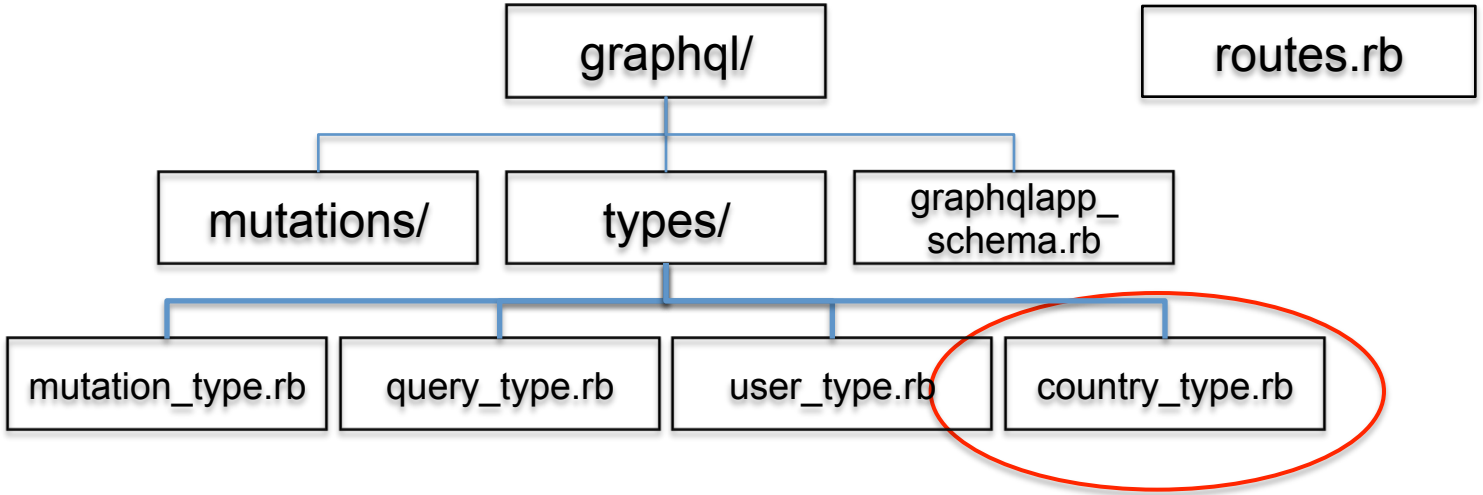
app/graphql/types/user_type.rb

```
01 Types::UserType = GraphQL::ObjectType.define do
02  name "UserType" # É como o schema irá identificar essa type.
03
04  # Nesta parte informamos quais campos serão visíveis numa consulta
05  field :id, types.ID
06  field :name, types.String
07  field :email, types.String
08  field :phone_number, types.Int
09  field :is_admin, types.Boolean, "Usuario administrador?
10
11  # Vamos expor, todos dados do País que estão no relacionamento “user.country”
12  field :country do
13    type Types::CountryType ## ←
14    description "País associado a este usuário"
15    resolve ->(user, args, context){
16      user.country
17    }
18  end
19 end
```

# Integração com Rails



# Integração com Rails



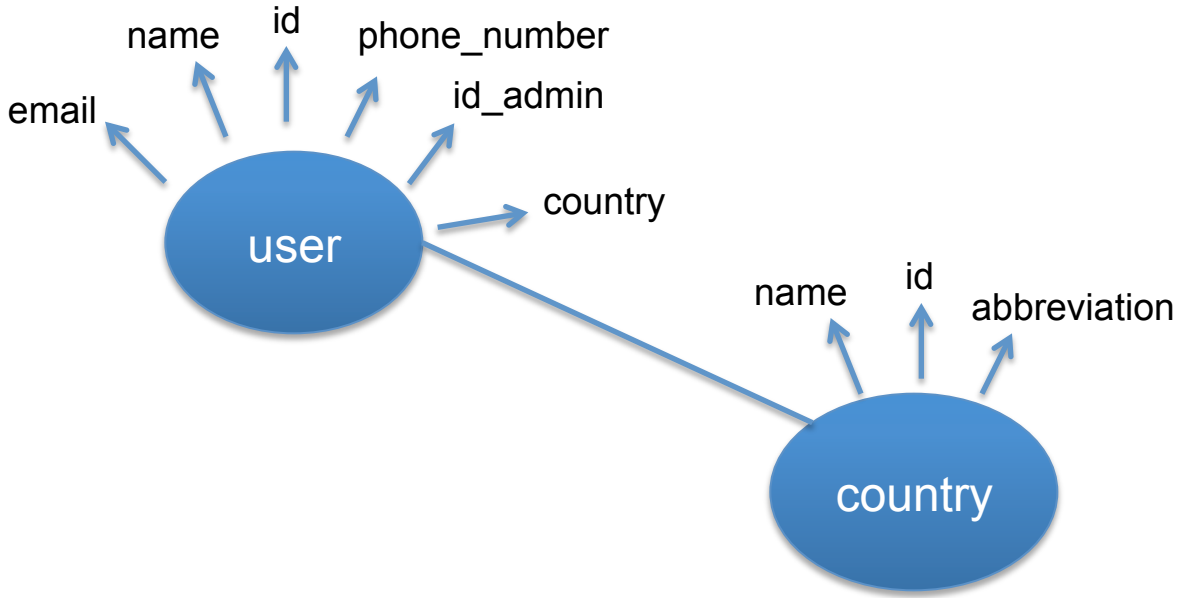
# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/types/country_type.rb

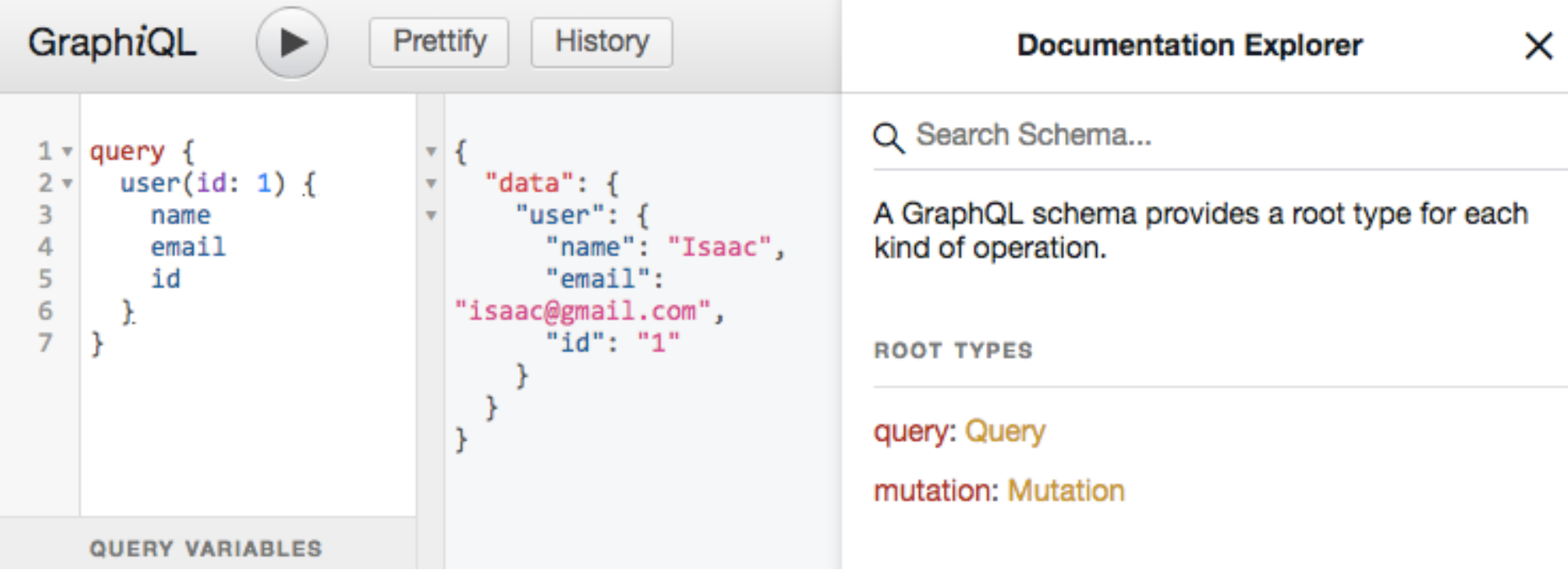
```
01 Types::CountryType = GraphQL::ObjectType.define do
02   name "CountryType" # É como o schema irá identificar essa type.
03
04   # Campos que serão visíveis numa consulta
05   field :id, types.ID
06   field :abbreviation, types.String
07   field :name, types.String
08 end
```

# Integração com Rails



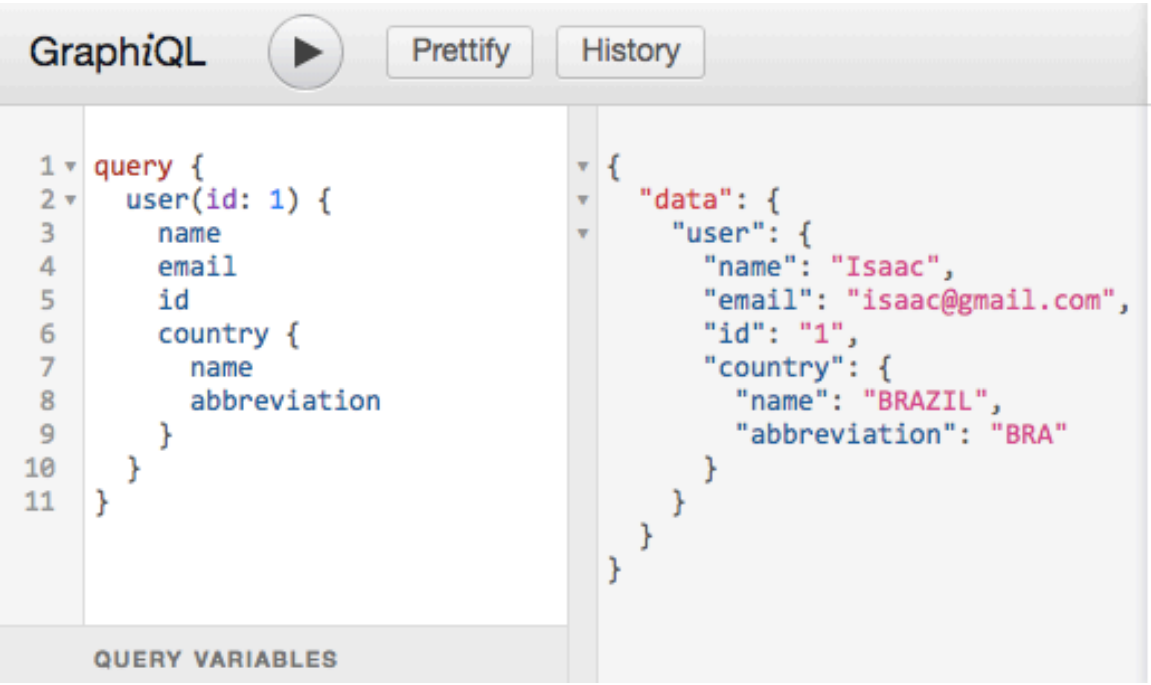
# Integração com Rails

THE DEVELOPER'S CONFERENCE



# Integração com Rails

THE DEVELOPER'S CONFERENCE

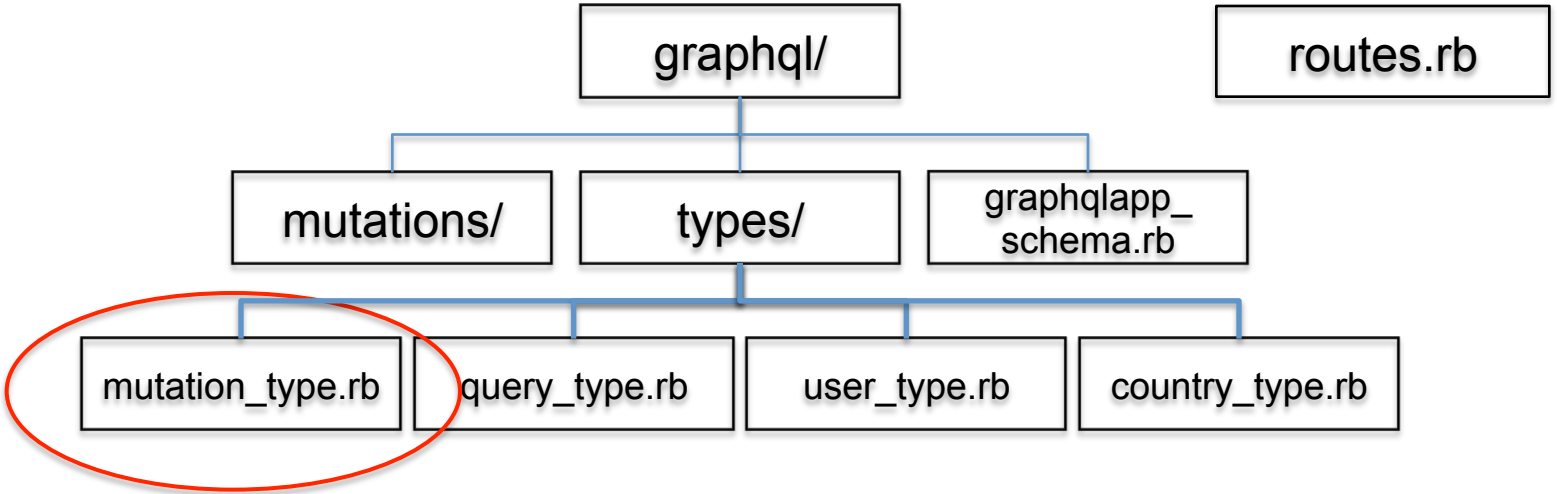


# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
$ curl -XPOST -d 'query={ user(id: 1) {id name}}'
http://localhost:3000/graphql

{"data":{"user":{"id":"1","name":Isaac"}}}
```

# Integração com Rails



# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/types/mutation_type.rb

```
01 Types::MutationType = GraphQL::ObjectType.define do
02   name "Mutation"
03
04
05 end
```

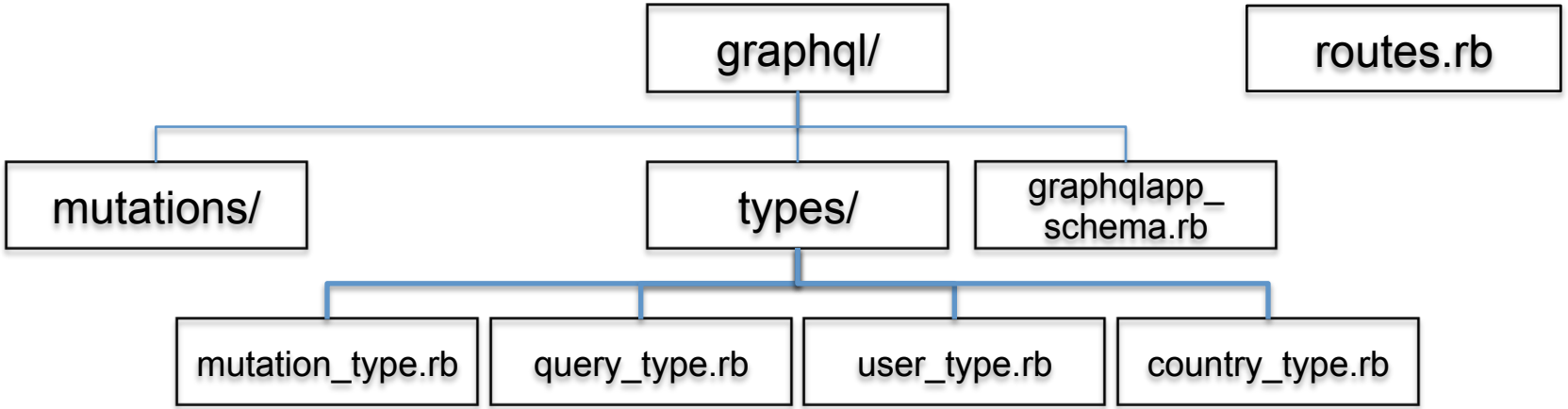
# Integração com Rails

THE DEVELOPER'S CONFERENCE

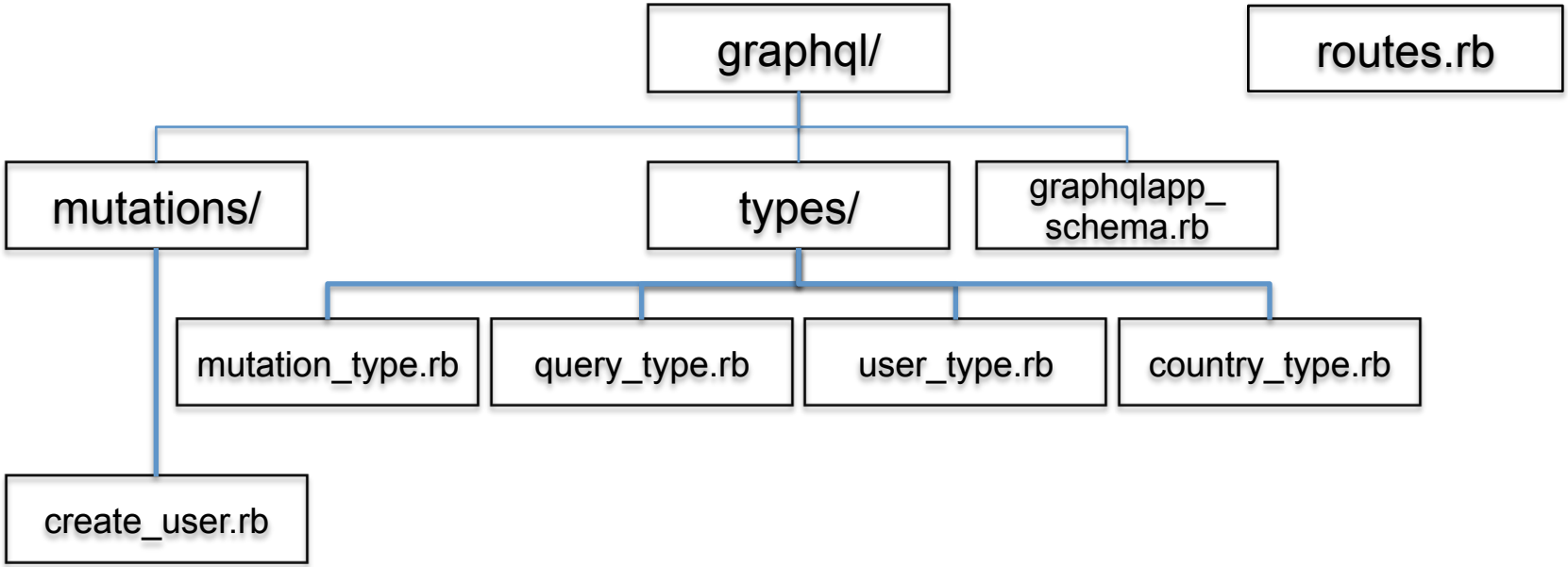
app/graphql/types/mutation_type.rb

```
01 Types::MutationType = GraphQL::ObjectType.define do
02   name "Mutation"
03
04   field :createUser, function: Mutations::CreateUser.new
05 end
```

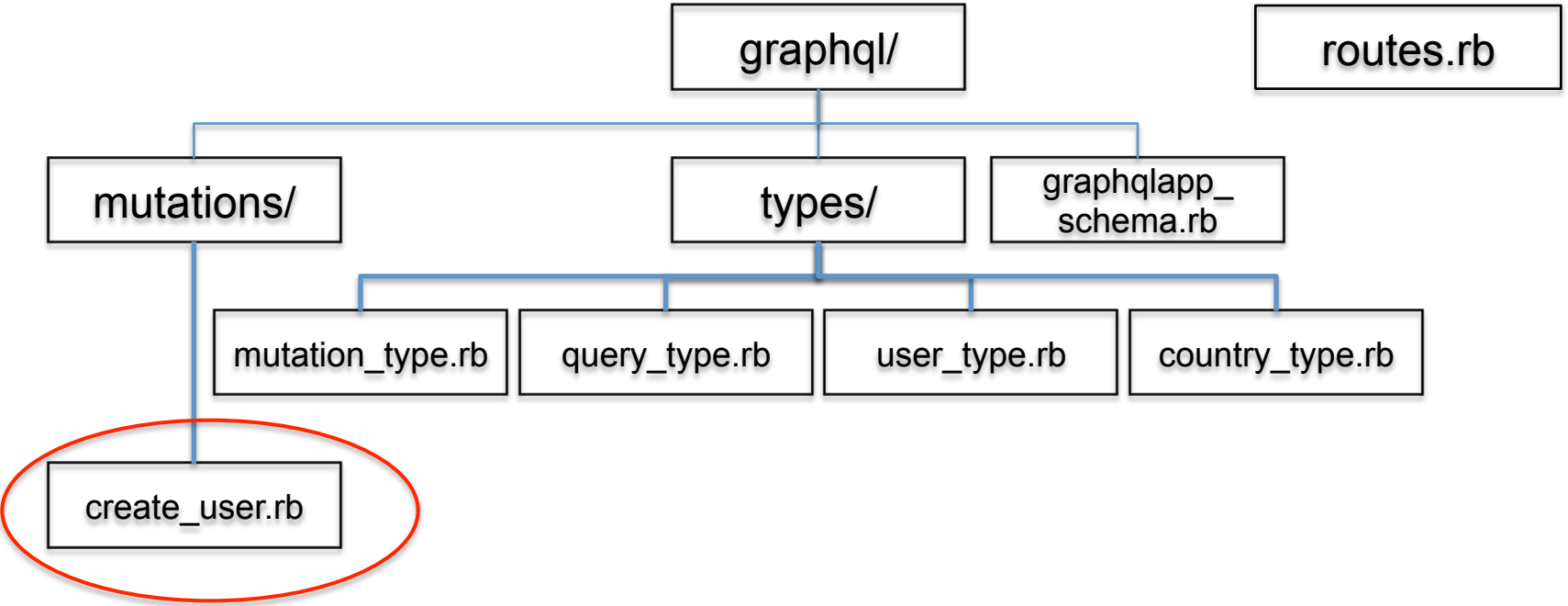
# Integração com Rails



# Integração com Rails



# Integração com Rails



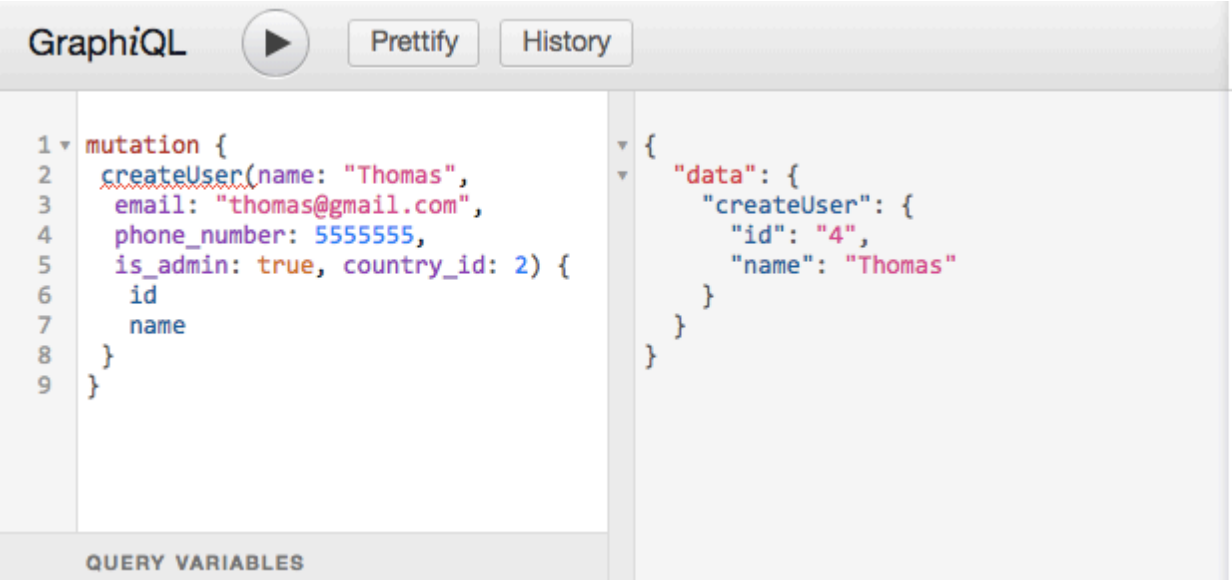
# Integração com Rails

THE DEVELOPER'S CONFERENCE

app/graphql/mutations/create_user.rb

```
01 class Mutations::CreateUser < GraphQL::Function
02  argument :id, types.ID
03  argument :name, types.String
04  argument :email, types.String
05  argument :phone_number, types.Int
06  argument :is_admin, types.Boolean
07  argument :country_id, types.Int
08
09  type Types::UserType # especificação do tipo de retorno
10
11  def call(obj, args, context)
12    User.create(args.to_h)
13  end
14 end
```

# Integração com Rails



# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
query {
  user(id: 1) {
    name
    email
  }
}
```

```
{
  "data": {
    "user": {
      "name": "Isaac",
      "email”:"isaac@gmail.com"
    }
  }
}
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
query {
    allUsers {
      id
    }
}
```

```
{
  "data": {
    "allUsers": [
      {
        "id": "1"
      },
      {
        "id": "2"
      }
    ]
  }
}
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
mutation {
  createUser(name: "Steve",
   is_admin: true,
   email:"jobs@apple.com",
   country_id: 1,
   phone_number: 55555555)
 {
    id
  }
}
```

```
{
  "data": {
    "createUser": {
      "id": "9"
    }
  }
}
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
mutation {
  updateUser(id: 9,
  name:"Steve Jobs"){
    name
  }
}
```

```
{
  "data": {
    "updateUser": {
      "name": "Steve Jobs"
    }
  }
}
```

# Integração com Rails

THE DEVELOPER'S CONFERENCE

```
mutation {
  deleteUser(id: 9)
}
```

```
{
  "data": {
    "deleteUser": true
  }
}
```

THE DEVELOPER'S CONFERENCE

# Finalizando com

# Perguntas e Respostas

# Finalizando...

THE DEVELOPER'S CONFERENCE

- O GraphQL irá dominar o mercado?
- O padrão REST vai morrer?

# Finalizando...

THE DEVELOPER'S CONFERENCE

- Empresas que usam GraphQL
  - Github
  - Twitter
  - Beek.io: a social network for book lovers
  - Nubank

# Finalizando...

THE DEVELOPER'S CONFERENCE

- Empresas que usam GraphQL
  - Atlassian
  - Coursera
  - KLM
  - Facebook

# Finalizando...

THE DEVELOPER'S CONFERENCE

- Retornar erro no caso das validações ou exceções

# Finalizando...

THE DEVELOPER'S CONFERENCE

app/graphql/mutations/create_user.rb

```
01 class Mutations::CreateUser < GraphQL::Function
02   ...
03   ..
04   .
11   def call(obj, args, context)
12     User.create(args.to_h)
13     User.save!
14     User
15   rescue ActiveRecord::RecordInvalid
16     GraphQL::ExecutionError.new("#{user.errors.full_messages.join(', ')}")
17   end
18 end
```

# Finalizando...

THE DEVELOPER'S CONFERENCE

```
mutation {
  createUser(name: "",
   country_id: 999)
 {
    id
  }
}
```

```
{
  "data": {
    "createUser": null
  },
  "errors": [
    {
      "message": "Country must exist, Name
can't be blank",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "createUser"
      ]
    }
  ]
}
```

# Finalizando...

THE DEVELOPER'S CONFERENCE

- Autenticação e Autorização

# Finalizando...

THE DEVELOPER'S CONFERENCE

```
mutation {
 SignIn(email: john@email.com”,
        password: “123456”) {
   authentication_token
 }
}
```

```
{
  "data": {
    ”SignIn": {
      "authentication_token":
”Hjkl982wz8"
    }
  }
}
```

# Finalizando...

THE DEVELOPER'S CONFERENCE

app/controllers/graphql_controller.rb

```
01 class GraphqlController < ApplicationController
02  def execute
    ...
    ..
03    session = Session.where(key: request.headers['Authorization']).first
04
05     context = {
06       current_user: session&.user
07     }
08
09   end
10 end
```

# Finalizando...

THE DEVELOPER'S CONFERENCE

app/graphql/types/query_type.rb

```
01 Types::QueryType = GraphQL::ObjectType.define do
02  name "Query”
03
04  field :user, Types::UserType do
05  argument :id, types.ID
06    description "Identificação do Usuário"
07
08    resolve -> (obj, args, ctx) {
09      Rails.logger.info "Logged in as #{ctx[:current_user]}"
10      User.where(id: args[:id]).first
11   }
12  end
13 end
```

# Finalizando...

THE DEVELOPER'S CONFERENCE

- Desvantagens.
- Quando não usar?

# Finalizando...

THE DEVELOPER'S CONFERENCE

- API servindo dados para aplicativo mobile.
- Este aplicativo não terá sua interface modificada.

# Finalizando...

THE DEVELOPER'S CONFERENCE

- Cultura da empresa ou equipe.
- Pode existir uma app híbrida:
- REST / GraphQL

# Finalizando...

THE DEVELOPER'S CONFERENCE

SQL databases x NO SQL databases

# Finalizando...

THE DEVELOPER'S CONFERENCE

RESTfull APIs x NoRESTfull APIs

# Finalizando...

THE DEVELOPER'S CONFERENCE

NoRESTfull API

# Perguntas?

THE DEVELOPER'S CONFERENCE

**Autor da Palestra:**
**Sergio Lima**

**Escritor no OneBitCode**

onebitcode.com/author/sergio-lima

**#rubydev.rb**

@sergiosouzalima

**Systems Analyst / Developer**

/sergiosouzalima

THE DEVELOPER'S CONFERENCE

**Autor da Palestra:**
Sergio Lima

# Obrigado!

## Escritor no OneBitCode

onebitcode.com/author/sergio-lima

## #rubydev.rb

@sergiosouzalima

## Systems Analyst / Developer

/sergiosouzalima

# Referências

THE DEVELOPER'S CONFERENCE


- http://graphql.org
- http://graphql-ruby.org/
- https://graphql.org/users/
- https://onebitcode.com/graphql-introducao/
- https://www.howtographql.com/graphql-ruby
- https://www.moesif.com/blog/technical/graphql/REST-vs-GraphQL-APIs-the-good-the-bad-the-ugly/
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://en.wikipedia.org/wiki/Application_programming_interface