Criando uma API completa com Rails
Leonardo Scorza (One Bit Code)

https://www.youtube.com/watch?v=EqOoElCjpNI

Resumo do curso feito por Roberto Pinheiro

Aula 01 - Planejando nossa API

O que é uma API?

É uma maneira de conversar com outros serviços.

Caso você precise de informações e dados que não possui em sua aplicação, você pode consumir uma API.

As API's podem ser pagas ou públicas.

O que vamos criar?

Vamos criar uma API de agenda de contatos.

O que vamos usar no projeto?

Planejando nossa API

SwaggerHub

Esse software permite documentar as API's.

Uma API normalmente é constituída de diversos endpoints. Cada URL é basicamente um endpoint. E cada endpoint tem uma função.

https://app.swaggerhub.com/apis/lscorza/OneBitContacts/1.0.0

OneBitContacts

1.0.0

[Base URL: virtserver.swaggerhub.com/lscorza/OneBitContacts/1.0.0]

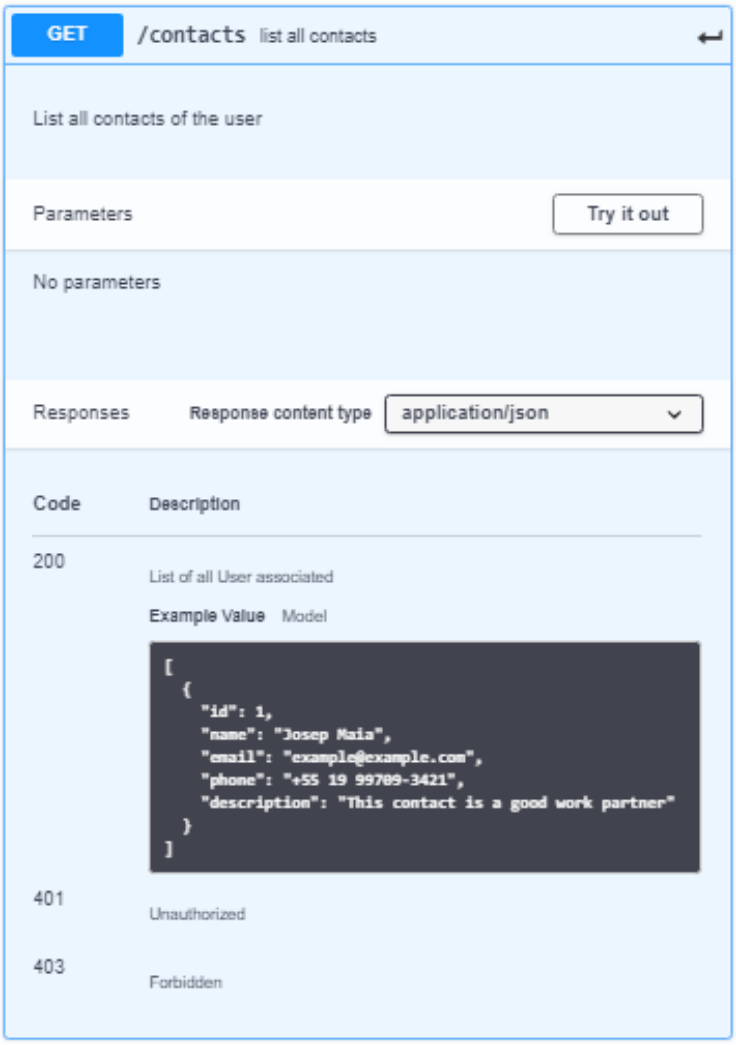
Api for OneBitContacts

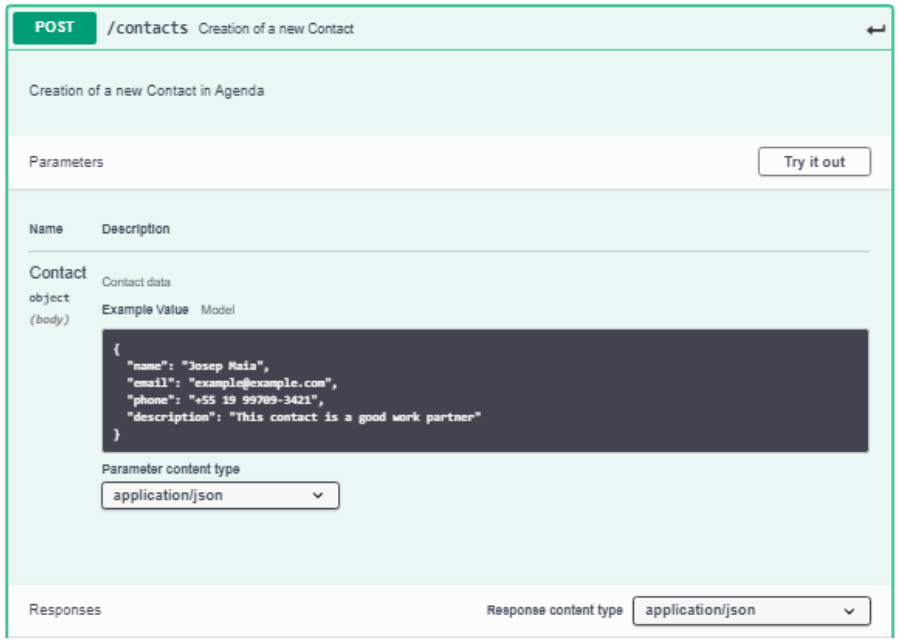
Contact the developer

Apache 2.0

Schemes

HTTPS


GET
/contacts list all contacts
List all contacts of the user
Parameters
Try it out
No parameters
Responses
Response content type
application/json
Code
Description
200
List of all User associated
Example Value
Model
[
{
"id": 1,
"name": "Josep Maia",
"email": "example@example.com",
"phone": "+55 19 99709-3421",
"description": "This contact is a good work partner"
}
]
401
Unauthorized
403
Forbidden


POST
/contacts Creation of a new Contact
Creation of a new Contact in Agenda
Parameters
Try it out
Name
Description
Contact
object
(body)
Contact data
Example Value
Model
{
"name": "Josep Maia",
"email": "example@example.com",
"phone": "+55 19 99709-3421",
"description": "This contact is a good work partner"
}
Parameter content type
application/json
Responses
Response content type
application/json

Code	Description
200	Item created Example Value Model

```
[
  {
    "id": 1,
    "name": "Josep Maia",
    "email": "example@example.com",
    "phone": "+55 19 99709-3421",
    "description": "This contact is a good work partner"
  }
]
```

401	Unauthorized
403	Forbidden

GET /contacts/:id Show Contact

Show contact information

Parameters Try it out

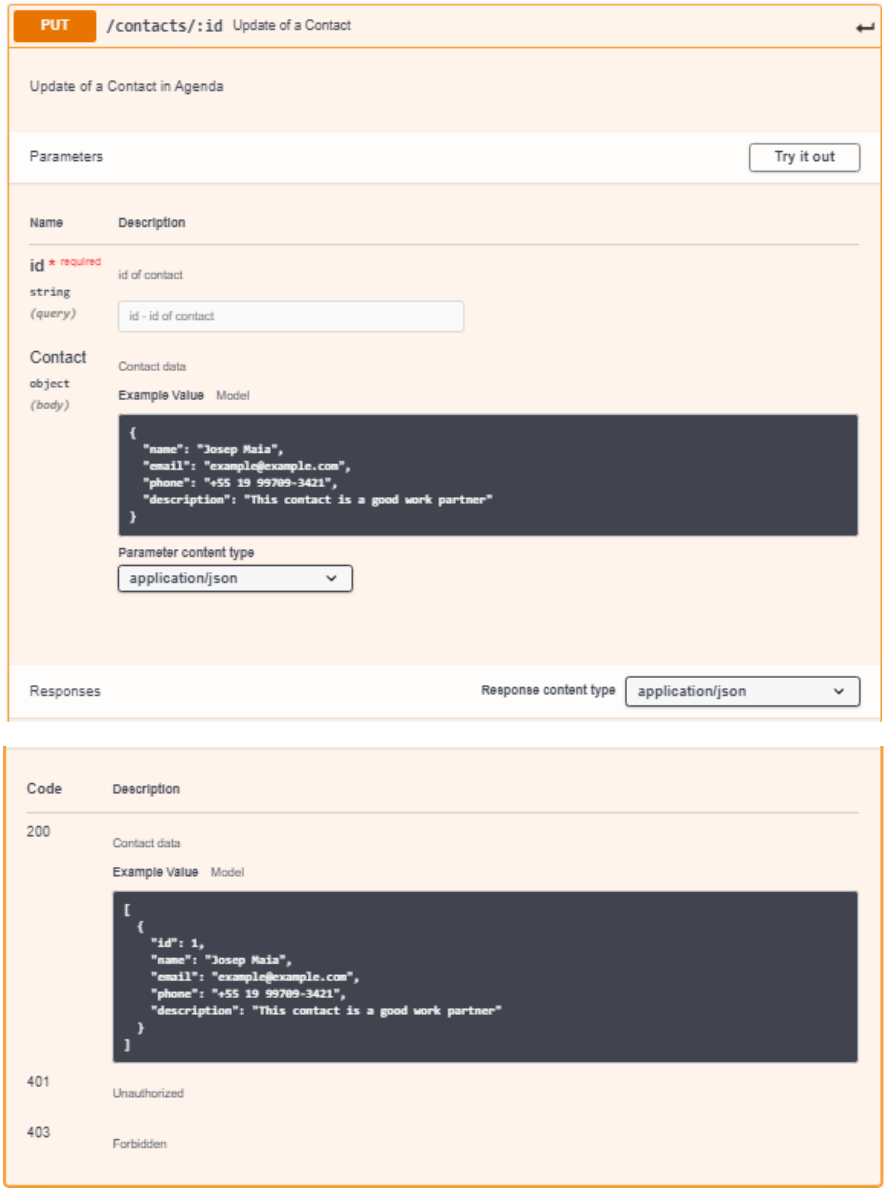
Name	Description
id * required string (query)	id of contact id - id of contact

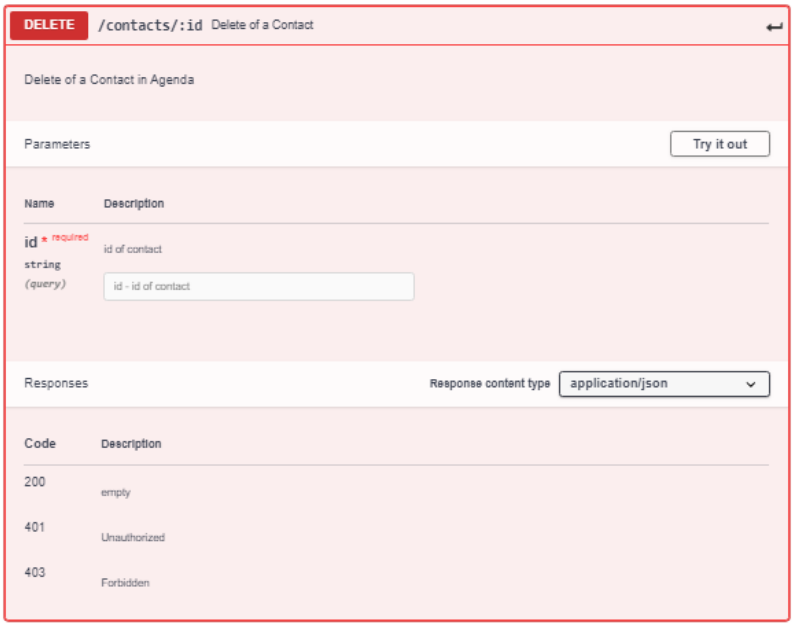
Responses Response content type application/json

Code	Description
200	Contact data Example Value Model

```
[
  {
    "id": 1,
    "name": "Josep Maia",
    "email": "example@example.com",
    "phone": "+55 19 99709-3421",
    "description": "This contact is a good work partner"
  }
]
```

401	Unauthorized
403	Forbidden


PUT
/contacts/:id Update of a Contact
Update of a Contact in Agenda
Parameters
Try it out
Name
Description
id * required
string
(query)
id of contact
id - id of contact
Contact
object
(body)
Contact data
Example Value
Model
{
"name": "Josep Maia",
"email": "example@example.com",
"phone": "+55 19 99709-3421",
"description": "This contact is a good work partner"
}
Parameter content type
application/json
Responses
Response content type
application/json
Code
Description
200
Contact data
Example Value
Model
[
{
"id": 1,
"name": "Josep Maia",
"email": "example@example.com",
"phone": "+55 19 99709-3421",
"description": "This contact is a good work partner"
}
]
401
Unauthorized
403
Forbidden


DELETE
/contacts/:id Delete of a Contact
Delete of a Contact in Agenda
Parameters
Try it out
Name
Description
id * required
string
(query)
id of contact
id - id of contact
Responses
Response content type
application/json
Code
Description
200
empty
401
Unauthorized
403
Forbidden

Aula 02 - Preparando a base da API

API's são maneiras de conectar serviços, com elas é possível se conectar ao Google para traduzir uma frase ou ao Watson da IBM para realizar um processamento de linguagem natural. Vamos ver o que é essencial para criar uma usando Ruby On Rails.

O que vamos utilizar

- Ruby
- Ruby On Rails
- Devise
- Devise Simple Token
- Rack Cors
- Rack Attack
- Postman

Gerando nosso Projeto

1 – Rode no console

```
rails new OneBitContacts --api
```

*Se quiser utilizar o PostgreSQL adicione ao comando:

```
--database=postgresql
```

Versionando a API

1 – Dentro da pasta ***controllers*** crie uma pasta chamada ***api***
2 – Dentro da pasta criada crie uma nova pasta chamada ***v1***
3 – Dentro da pasta criada crie um arquivo chamado ***api_controller.rb*** e coloque nele:

```
module Api::V1
    class ApiController < ApplicationController
        # Métodos globais
    end
end
```

Habilitando o CORS

Acrescente no seu Gemfile:

```
gem 'rack-cors'
```

Instale a gem rodando:

```
bundle install
```

Acrescente no seu ***config/application.rb***:

```
config.middleware.insert_before 0, Rack::Cors do
   allow do
        origins '*'
        resource '*',
                headers: :any,
                methods: %i(get post put patch delete options head)
   end
end
```

config/application.rb

```
require_relative 'boot'

require "rails"
# Pick the frameworks you want:
require "active_model/railtie"
require "active_job/railtie"
require "active_record/railtie"
require "active_storage/engine"
require "action_controller/railtie"
require "action_mailer/railtie"
require "action_view/railtie"
require "action_cable/engine"
# require "sprockets/railtie"
require "rails/test_unit/railtie"

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(*Rails.groups)

module OneBitContacts
  class Application < Rails::Application
    # Initialize configuration defaults for originally generated Rails version.
    config.load_defaults 5.2
```

```
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration can go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded after loading
    # the framework and any gems in your application.

    # Only loads a smaller set of middleware suitable for API only apps.
    # Middleware like session, flash, cookies can be added back manually.
    # Skip views, helpers and assets when generating a new resource.
    config.api_only = true

    config.middleware.insert_before 0, Rack::Cors do
      allow do
        origins '*'
        resource '*',
                headers: :any,
                methods: %i(get post put patch delete options head)
      end
    end

  end
end
```

Configurando o Rack Attack

Acrescente no seu Gemfile:

gem 'rack-attack'

Instale a gem:

bundle install

Acrescente em config/application:

config.middleware.use Rack::Attack

config/application.rb

```
require_relative 'boot'

require "rails"
# Pick the frameworks you want:
require "active_model/railtie"
require "active_job/railtie"
require "active_record/railtie"
require "active_storage/engine"
require "action_controller/railtie"
require "action_mailer/railtie"
require "action_view/railtie"
require "action_cable/engine"
# require "sprockets/railtie"
require "rails/test_unit/railtie"

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(*Rails.groups)

module OneBitContacts
  class Application < Rails::Application
    # Initialize configuration defaults for originally generated Rails version.
    config.load_defaults 5.2

    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration can go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded after loading
    # the framework and any gems in your application.

    # Only loads a smaller set of middleware suitable for API only apps.
    # Middleware like session, flash, cookies can be added back manually.
    # Skip views, helpers and assets when generating a new resource.
    config.api_only = true
```

```
    config.middleware.insert_before 0, Rack::Cors do
      allow do
        origins '*'
        resource '*',
                 headers: :any,
                 methods: %i(get post put patch delete options head)
      end
    end

    config.middleware.use Rack::Attack

  end
end
```

Crie um arquivo chamado ***rack_attack.rb*** no seu ***config/initializers/*** e coloque nele:

config/initializers/rack_attack.rb

```
class Rack::Attack
  Rack::Attack.cache.store = ActiveSupport::Cache::MemoryStore.new

  # Allow all local traffic
  safelist('allow-localhost') do |req|
    '127.0.0.1' == req.ip || '::1' == req.ip
  end

  # Allow an IP address to make 10 requests every 10 seconds
  throttle('req/ip', limit: 5, period: 5) do |req|
    req.ip
  end


  # Throttle login attempts by email address
  #throttle("logins/email", limit: 5, period: 20.seconds) do |req|
  #  if req.path == '/users/sign_in' && req.post?
  #    req.params['email'].presence
  #  end
  #end
end
```

Instalando o Devise

Acrescente no seu Gemfile:

gem 'devise'

Instale rodando:

bundle install

Em config/environments/development.rb coloque:

config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }

config/environments/development.rb

```
Rails.application.configure do
  # Settings specified here will take precedence over those in config/application.rb.

  # In the development environment your application's code is reloaded on
  # every request. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.cache_classes = false

  # Do not eager load code on boot.
  config.eager_load = false

  # Show full error reports.
  config.consider_all_requests_local = true

  # Enable/disable caching. By default caching is disabled.
  if Rails.root.join('tmp/caching-dev.txt').exist?
    config.action_controller.perform_caching = true

    config.cache_store = :memory_store
    config.public_file_server.headers = {
      'Cache-Control' => "public, max-age=#{2.days.seconds.to_i}"
    }
  else
    config.action_controller.perform_caching = false

    config.cache_store = :null_store
  end

  # Don't care if the mailer can't send.
  config.action_mailer.raise_delivery_errors = false

  config.action_mailer.perform_caching = false

  # Print deprecation notices to the Rails logger.
  config.active_support.deprecation = :log
```

```
  # Raise an error on page load if there are pending migrations.
  config.active_record.migration_error = :page_load

  config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }


  # Raises error for missing translations
  # config.action_view.raise_on_missing_translations = true

  # Use an evented file watcher to asynchronously detect changes in source code,
  # routes, locales, etc. This feature depends on the listen gem.
  config.file_watcher = ActiveSupport::EventedFileUpdateChecker
end
```

Rode no console:

```
rails generate devise:install
```

```
C:\sites\OneBitContacts (master)
λ rails generate devise:install
      create  config/initializers/devise.rb
      create  config/locales/devise.en.yml
===============================================================================

Some setup you must do manually if you haven't yet:

  1. Ensure you have defined default url options in your environments files. Here
     is an example of default_url_options appropriate for a development environment
     in config/environments/development.rb:

       config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }

     In production, :host should be set to the actual host of your application.

  2. Ensure you have defined root_url to *something* in your config/routes.rb.
     For example:

       root to: "home#index"

  3. Ensure you have flash messages in app/views/layouts/application.html.erb.
     For example:

       <p class="notice"><%= notice %></p>
       <p class="alert"><%= alert %></p>

  4. You can copy Devise views (for customization) to your app by running:

       rails g devise:views

===============================================================================
```

Rode no console:

```
rails generate devise User
```

```
C:\sites\OneBitContacts (master)
λ rails generate devise User
      invoke  active_record
      create    db/migrate/20191112014801_devise_create_users.rb
      create    app/models/user.rb
      invoke    test_unit
      create      test/models/user_test.rb
      create      test/fixtures/users.yml
      insert    app/models/user.rb
       route  devise_for :users
```

rails g migration addNameToUser name:string

```
C:\sites\OneBitContacts (master)
λ rails g migration addNameToUser name:string
      invoke  active_record
      create    db/migrate/20191112015013_add_name_to_user.rb
```

Crie o banco de dados:

rails db:create

```
C:\sites\OneBitContacts (master)
λ rails db:create
Created database 'OneBitContacts_development'
Created database 'OneBitContacts_test'
```

Gere as migrations:

rails db:migrate

```
C:\sites\OneBitContacts (master)
λ rails db:migrate
== 20191112014801 DeviseCreateUsers: migrating ================================
-- create_table(:users)
   -> 0.3243s
-- add_index(:users, :email, {:unique=>true})
   -> 0.4103s
-- add_index(:users, :reset_password_token, {:unique=>true})
   -> 0.2749s
== 20191112014801 DeviseCreateUsers: migrated (1.0132s) =======================

== 20191112015013 AddNameToUser: migrating ====================================
-- add_column(:users, :name, :string)
   -> 0.6277s
== 20191112015013 AddNameToUser: migrated (0.6286s) ===========================
```

Devise Simple Token

Adicione ao seu Gemfile:

gem 'simple_token_authentication', '~> 1.0'

Instale rodando:

bundle install

Adicione ao seu User Model:

acts_as_token_authenticatable

app/models/user.rb

```
class User < ApplicationRecord

  acts_as_token_authenticatable

  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable

  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable
end
```

Rode no console:

rails g migration add_authentication_token_to_users "authentication_token:string{30}:uniq"

```
C:\sites\OneBitContacts (master)
λ rails g migration add_authentication_token_to_users "authentication_token:string{30}:uniq"
      invoke  active_record
      create    db/migrate/20191112020700_add_authentication_token_to_users.rb
```

rake db:migrate

```
C:\sites\OneBitContacts (master)
λ rake db:migrate
== 20191112020700 AddAuthenticationTokenToUsers: migrating ====================
-- add_column(:users, :authentication_token, :string, {:limit=>30})
   -> 0.4549s
-- add_index(:users, :authentication_token, {:unique=>true})
   -> 0.2297s
== 20191112020700 AddAuthenticationTokenToUsers: migrated (0.6866s) ===========
```

Coloque no seu controler ***api/v1/api_controller.rb***:

app/controllers/api/v1/api_controller.rb

```
module Api::V1
  class ApiController < ApplicationController
    acts_as_token_authentication_handler_for User
    before_action :require_authentication!

    private
    def require_authentication!
      throw(:warden, scope: :user) unless current_user.presence
    end
  end
end
```

Testando

rails c

User.create(email: 'bootcamp@onebitcode.com', password: '123456')

Aula 03 - Desenvolvendo e testando nossa API

Gerando os models

Rode no console:

rails g model Contact name:string email:string phone:string description:text user:references

```
C:\sites\OneBitContacts (master)
λ rails g model Contact name:string email:string phone:string description:text user:references
      invoke  active_record
      create    db/migrate/20191112105700_create_contacts.rb
      create    app/models/contact.rb
      invoke    test_unit
      create      test/models/contact_test.rb
      create      test/fixtures/contacts.yml
```

rails db:migrate

```
C:\sites\OneBitContacts (master)
λ rails db:migrate
== 20191112105700 CreateContacts: migrating ===================================
-- create_table(:contacts)
   -> 0.2611s
== 20191112105700 CreateContacts: migrated (0.2630s) ==========================
```

Adicione ao seu Model Contact:

validates :name, :user, presence: true

app/models/contact.rb

```
class Contact < ApplicationRecord
  belongs_to :user
  validates :name, :user, presence: true
end
```

Adicione ao seu Model User:

has_many :contacts, dependent: :destroy

app/models/user.rb

```
class User < ApplicationRecord
  acts_as_token_authenticatable
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  has_many :contacts, dependent: :destroy
end
```

Gerando os controllers

Rode no console:

rails g controller api/v1/contacts

```
C:\sites\OneBitContacts (master)
λ rails g controller api/v1/contacts
      create  app/controllers/api/v1/contacts_controller.rb
      invoke  test_unit
      create    test/controllers/api/v1/contacts_controller_test.rb
```

Coloque no controller gerado:

app/controllers/api/v1/contacts_controller.rb

```
class Api::V1::ContactsController < Api::V1::ApiController
  before_action :set_contact, only: [:show, :update, :destroy]
  before_action :require_authorization!, only: [:show, :update, :destroy]

  # GET /api/v1/contacts
  def index
    @contacts = current_user.contacts

    render json: @contacts
  end

  # GET /api/v1/contacts/1
  def show
    render json: @contact
  end

  # POST /api/v1/contacts
  def create
    @contact = Contact.new(contact_params.merge(user: current_user))
```

```
    if @contact.save
      render json: @contact, status: :created
    else
      render json: @contact.errors, status: :unprocessable_entity
    end
  end

  # PATCH/PUT /api/v1/contacts/1
  def update
    if @contact.update(contact_params)
      render json: @contact
    else
      render json: @contact.errors, status: :unprocessable_entity
    end
  end

  # DELETE /api/v1/contacts/1
  def destroy
    @contact.destroy
  end

  private
  # Use callbacks to share common setup or constraints between actions.
  def set_contact
    @contact = Contact.find(params[:id])
  end

  # Only allow a trusted parameter "white list" through.
  def contact_params
    params.require(:contact).permit(:name, :email, :phone, :description)
  end

  def require_authorization!
    unless current_user == @contact.user
      render json: {}, status: :forbidden
    end
  end
end
```

Ajustando as rotas

config/routes.rb

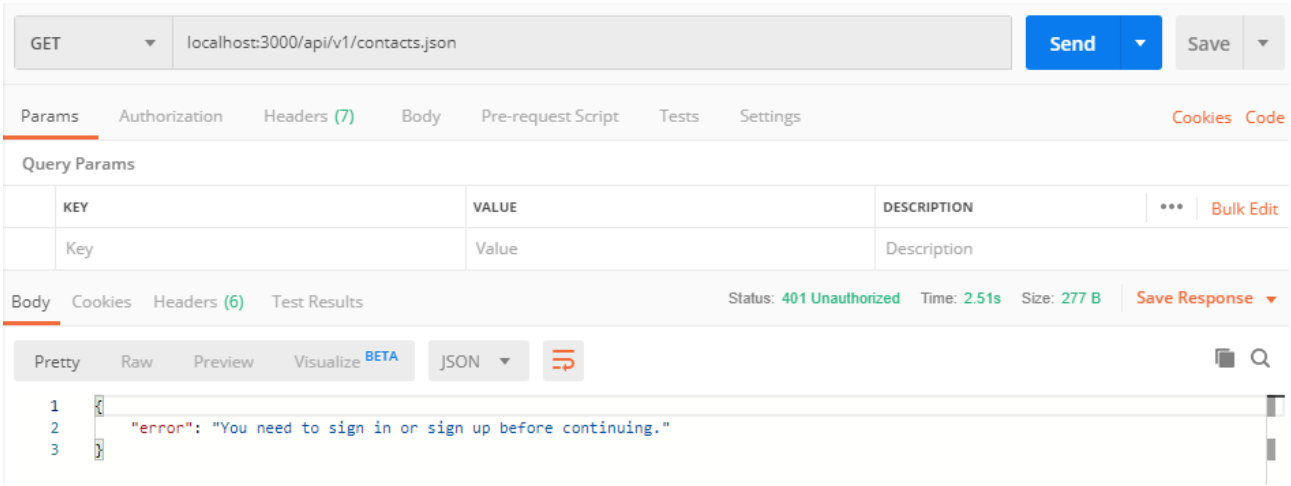
```
Rails.application.routes.draw do
  devise_for :users
  namespace :api do
    namespace :v1 do
      resources :contacts
    end
  end

  # constraints subdomain: 'api' do
  #   scope module: 'api' do
  #     namespace :v1 do
  #       resources :contacts
  #     end
  #   end
  # end
end
```

Testando os endpoints

Teste usando Postman ou CURL como no vídeo.

rails s

localhost:3000/api/v1/contacts.json



rails c

User.create(email: 'bootcamp@onebitcode.com', password: '123456')

```
irb(main):002:0> User.create(email: 'bootcamp@onebitcode.com', password: '123456')
   (0.3ms)  BEGIN
  User Exists (0.5ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` = BINARY 'bootcamp@onebitcode.com' LIMIT 1
   (0.5ms)  SELECT COUNT(*) FROM `users` WHERE `users`.`authentication_token` = 'vX3mo7Z19jm3o3Titp_T'
  User Create (0.6ms)  INSERT INTO `users` (`email`, `encrypted_password`, `created_at`, `updated_at`, `authentication
_token`) VALUES ('bootcamp@onebitcode.com', '$2a$11$sTAwOKB4j5q84FYi9E8OB.PPheneZgpp54lXqy8bFTiiZ2A/ZbPgK', '2019-11-1
2 14:04:55', '2019-11-12 14:04:55', 'vX3mo7Z19jm3o3Titp_T')
   (31.5ms)  COMMIT
=> #<User id: 2, email: "bootcamp@onebitcode.com", created_at: "2019-11-12 14:04:55", updated_at: "2019-11-12 14:04:55
", name: nil, authentication_token: "vX3mo7Z19jm3o3Titp_T">
irb(main):003:0>
```

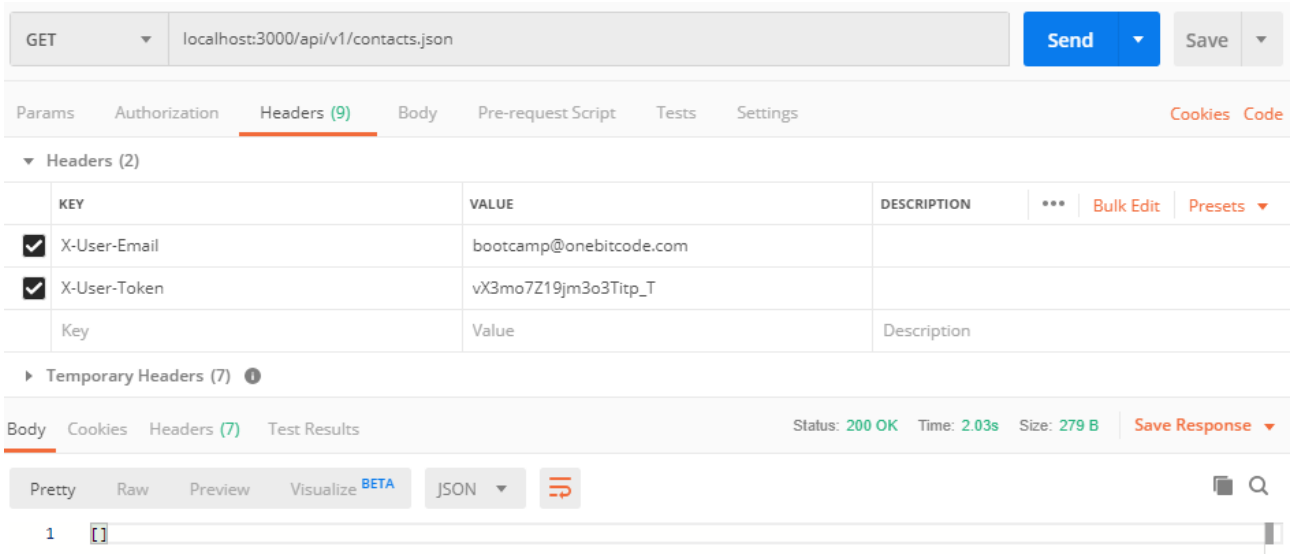
email: bootcamp@onebitcode.com
authentication_token: vX3mo7Z19jm3o3Titp_T

Em header, entre com:

Key	Value
X-User-Email	bootcamp@onebitcode.com
X-User-Token	vX3mo7Z19jm3o3Titp_T

Reinicie com:

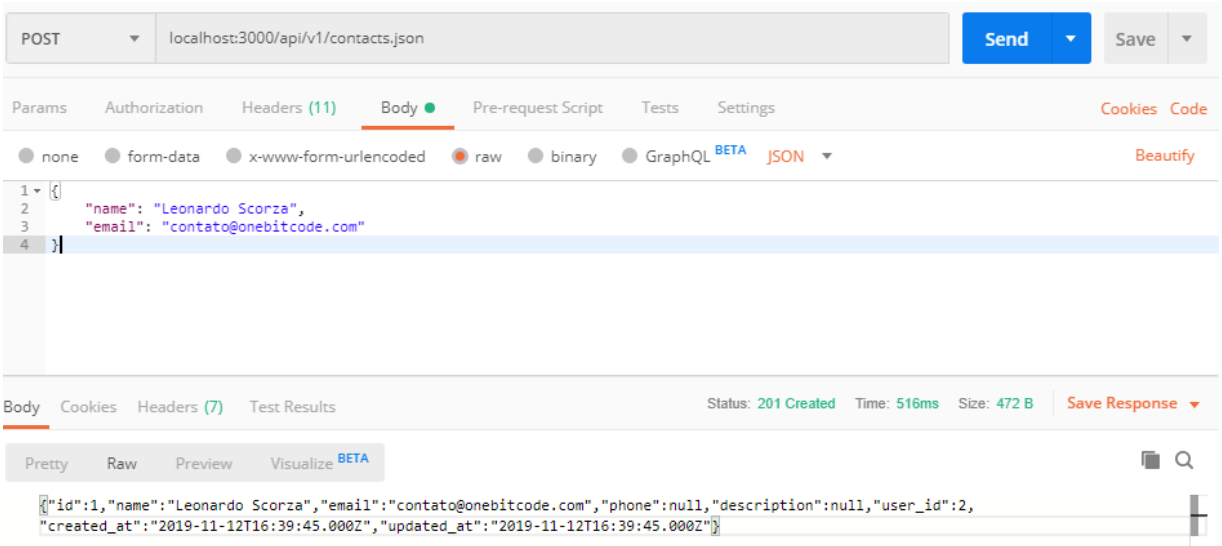
rails s



Criando usuários

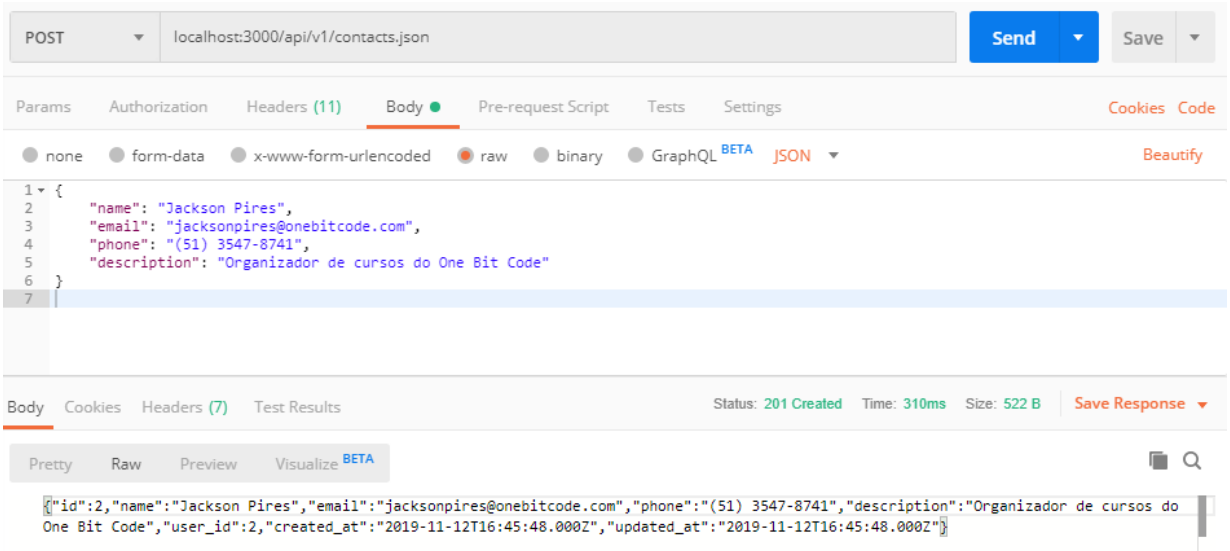
Em body entre com:

```
{
    "name": "Leonardo Scorza",
    "email": "contato@onebitcode.com"
}
```



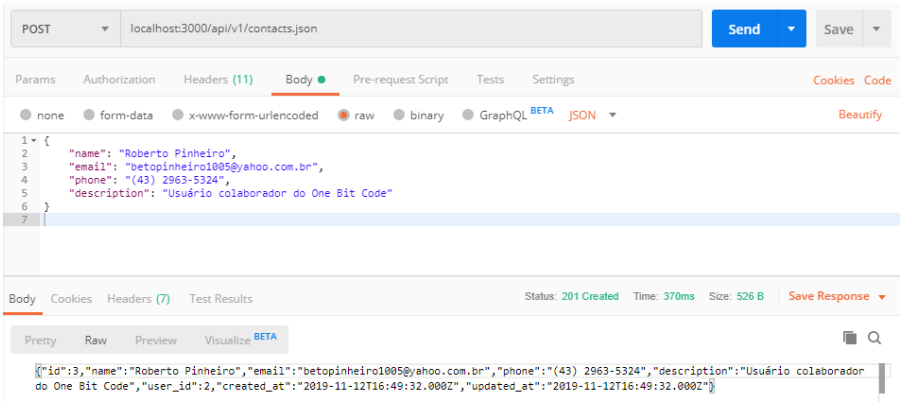
Em body entre com:

```
{
    "name": "Jackson Pires",
    "email": "jacksonpires@onebitcode.com",
    "phone": "(51) 3547-8741",
    "description": "Organizador de cursos do One Bit Code"
}
```

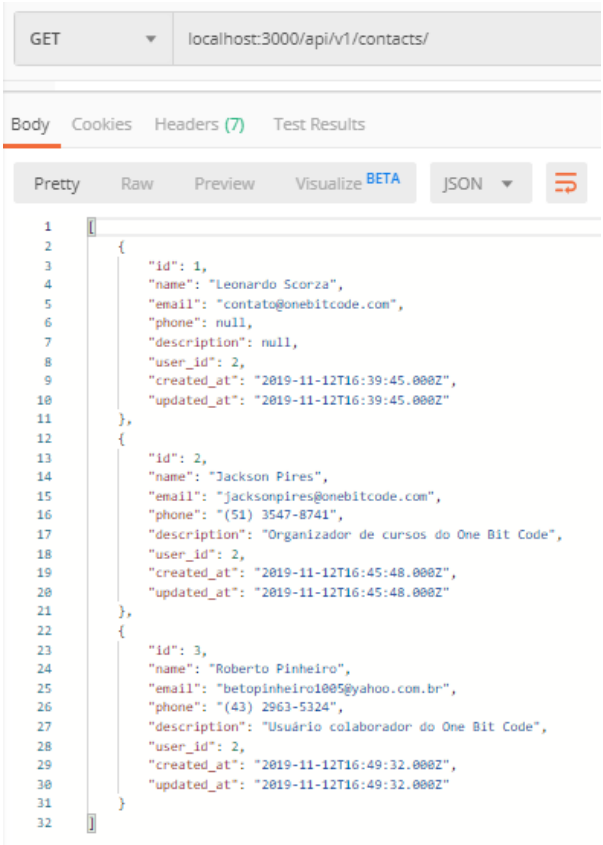


Em body entre com:

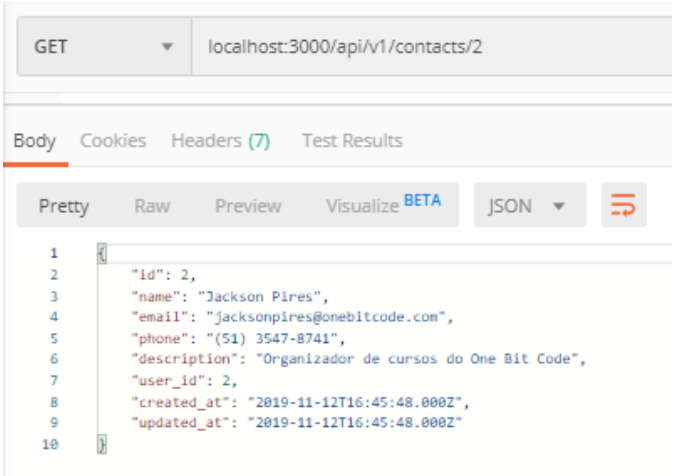
```
{
    "name": "Roberto Pinheiro",
    "email": "betopinheiro1005@yahoo.com.br",
    "phone": "(43) 2963-5324",
    "description": "Usuário colaborador do One Bit Code"
}
```



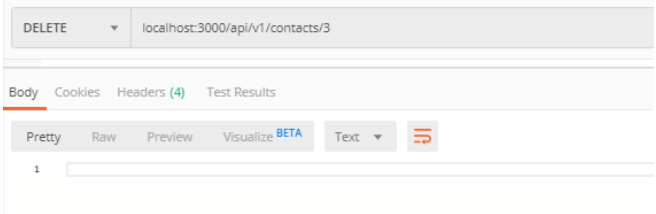
Listando os usuários

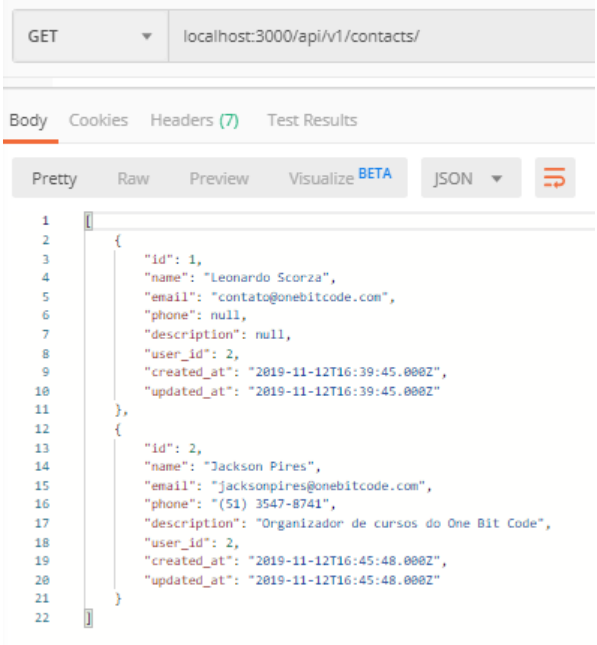


Exibindo detalhes de um usuário

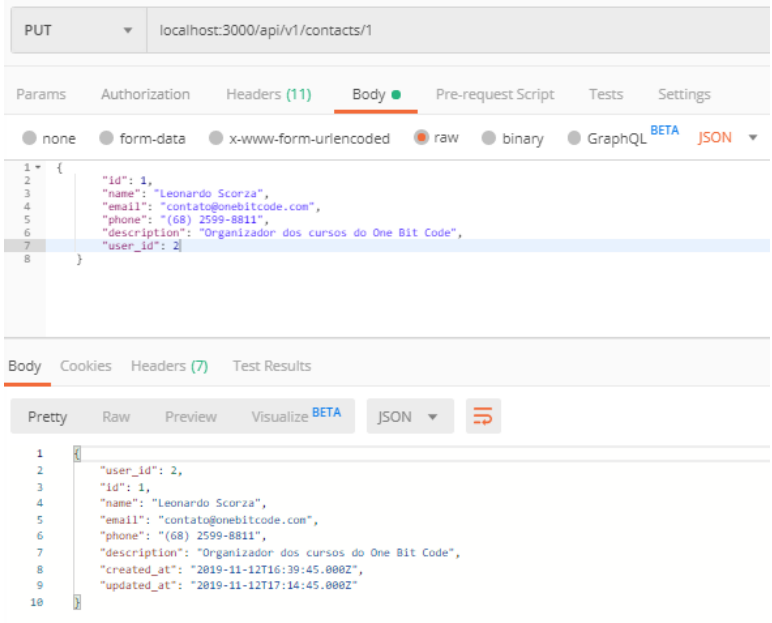


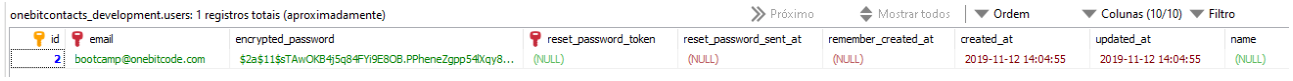
Excluindo um usuário

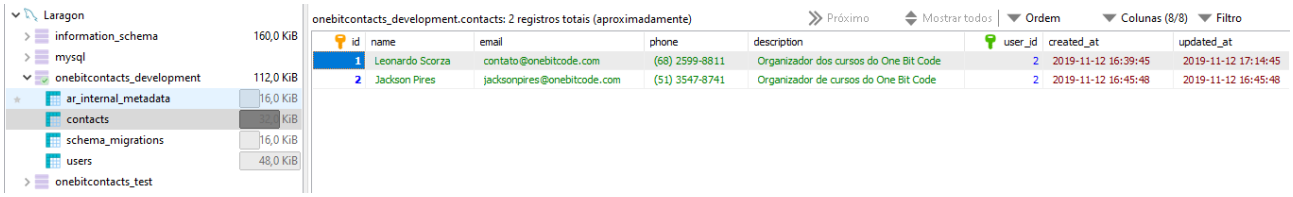




Atualizando os dados de um usuário







Desafios

1 - Paginar os retornos do endpoint /contacts

2 - Criar a estrutura para permitir a inclusão de multiplos endereços nos Contatos