Universidad Nacional Autónoma de Nicaragua, León.

UNAN - León

Tema:

- API Datos Estudiantes

Nombre:

1. Luis Adolfo Diaz Mendoza
2. Esteban Josue Prado Tellez
3. Luis Miguel Meza Vega
4. Jonathan Santiago Suazo Montalvan
5. Desbye Ahinoa Tercero Blandon

Docente:

- Ing. Ervin Montes

Componente:

- Software como un servicio

API de Datos de Estudiante utilizando Rswag en Rails.
Rswag facilita la generación de documentación interactiva en formato Swagger/Open API

1. Instalar las Gemas Necesarias

Primero, asegúrate de tener instaladas las gemas Rswag y rswag-ui en tu aplicación Rails.

Agregar las Gemas al Gemfile

Abre tu Gemfile y añade las siguientes líneas:

- gem rswag
- gem rswag-ui

Instalar las Gemas

Ejecuta el siguiente comando en tu terminal para instalar las gemas:

```
bundle install
```

Generar los Archivos de Configuración de Rswag

Después de instalar las gemas, ejecuta el siguiente comando para generar los archivos de configuración necesarios:

```
rails generate rswag:install
```

Este comando creará varios archivos y directorios, incluyendo:

- swagger_helper.rb en spec/
- Rutas necesarias para acceder a la documentación
- Configuraciones para Rswag en config/initializers/

2. Configurar Rswag

Configurar swagger_helper.rb

Abre el archivo spec/swagger_helper.rb y configura los detalles básicos de tu API.

spec/swagger_helper.rb

```
RSpec.configure do |config|
 config.swagger_root = Rails.root.to_s + /swagger

 config.swagger_docs = {
  v1/swagger.yaml => {
  openapi: 3.0.1 ,
  info: {
  title: Datos Estudiante API ,
  version: v1
  },
  paths: {},
  components: {
  schemas: {
   Estudiante: {
   type: :object,
   properties: {
    id: { type: :integer },
    nombres: { type: :string },
    apellidos: { type: :string },
    carrera: { type: :string },
    carnet: { type: :string },
    created_at: { type: :string, format: date-time },
    updated_at: { type: :string, format: date-time }
   },
   required: [ nombres , apellidos , carrera , carnet ]
   },
   Error: {
   type: :object,
   properties: {
    errors: {
    type: :object,
    additionalProperties: {
     type: :array,
     items: { type: :string }
    }
    }
   },
   required: [ errors ]
   }
  }
  }
 }
 }
 config.swagger_format = :yaml
end
```

Configurar las Rutas de Swagger UI

Las rutas para acceder a la documentación de Swagger estén configuradas en tu config/routes.rb :

config/routes.rb

```
Rails.application.routes.draw do
    Otros endpoints de tu API

Rutas para Rswag
mount Rswag::Ui::Engine => /api-docs
mount Rswag::Api::Engine => /api-docs
end
```

Esto permitirá acceder a la documentación interactiva en http://localhost:3000/api-docs .

3. Escribir Especificaciones de Rswag para tus End points

Crear Archivos de Especificación

Crea archivos de especificación para cada recurso en spec/integration/ .

Por ejemplo, para el recurso estudiantes , se crea estudiantes_spec.rb :

```
mkdir -p spec/integration
touch spec/integration/estudiantes_spec.rb
```

Escribir las Especificaciones

Abre spec/integration/estudiantes_spec.rb y añade las especificaciones para cada end point.

Ejemplo Completo para el Recurso Estudiante

spec/integration/estudiantes_spec.rb

```
require swagger_helper

RSpec.describe Estudiantes API , type: :request do
 path /estudiantes do

 get Listar estudiantes do
  tags Estudiantes
  produces application/json

  response 200 , lista de estudiantes do
```

```
  schema type: :array,
    items: { $ref => /components/schemas/Estudiante }

  run_test!
  end
 end

 post Crear un estudiante do
  tags Estudiantes
  consumes application/json
  produces application/json
  parameter name: :estudiante, in: :body, schema: {
   $ref => /components/schemas/Estudiante
  }

  response 201 , estudiante creado do
  let(:estudiante) { { nombres: Gema Mercedes , apellidos: Bravo Lanzas , carrera: Ingeniería
Telemática , carnet: 21-03968-1 } }
  run_test!
  end

  response 422 , errores de validación do
  let(:estudiante) { { nombres: , apellidos: , carrera: , carnet: } }
  run_test!
  end
 end
 end

 path /estudiantes/{id} do
 parameter name: id , in: :path, type: :integer, description: ID del estudiante

 get Obtener un estudiante do
  tags Estudiantes
  produces application/json

  response 200 , estudiante encontrado do
  let(:id) { Estudiante.create(nombres: Gema Mercedes , apellidos: Bravo Lanzas , carrera: Ingeniería
Telemática , carnet: 21-03968-1 ).id }
  schema $ref => /components/schemas/Estudiante

  run_test!
  end

  response 404 , estudiante no encontrado do
  let(:id) { invalid }
  run_test!
  end
 end
```

```
 put Actualizar un estudiante do
  tags Estudiantes
  consumes application/json
  produces application/json
  parameter name: :estudiante, in: :body, schema: {
   $ref => /components/schemas/Estudiante
  }

  response 200 , estudiante actualizado do
  let(:id) { Estudiante.create(nombres: Gema Mercedes , apellidos: Bravo Lanzas , carrera: Ingeniería
Telemática , carnet: 21-03968-1 ).id }
  let(:estudiante) { { nombres: Gema M. , apellidos: Bravo L. , carrera: Ingeniería de Sistemas } }
  run_test!
  end

  response 404 , estudiante no encontrado do
  let(:id) { invalid }
  let(:estudiante) { { nombres: Gema M. , apellidos: Bravo L. , carrera: Ingeniería de Sistemas } }
  run_test!
  end

  response 422 , errores de validación do
  let(:id) { Estudiante.create(nombres: Gema Mercedes , apellidos: Bravo Lanzas , carrera: Ingeniería
Telemática , carnet: 21-03968-1 ).id }
  let(:estudiante) { { nombres: , apellidos: , carrera: } }
  run_test!
  end
 end

 delete Eliminar un estudiante do
  tags Estudiantes
  produces application/json

  response 204 , estudiante eliminado do
  let(:id) { Estudiante.create(nombres: Gema Mercedes , apellidos: Bravo Lanzas , carrera: Ingeniería
Telemática , carnet: 21-03968-1 ).id }
  run_test!
  end

  response 404 , estudiante no encontrado do
  let(:id) { invalid }
  run_test!
  end
 end
 end
end
```

Explicación de las Especificaciones

- path : Define la ruta del endpoint.
- get , post , put , delete : Especifica el método HTTP.
- tags : Agrupa los endpoints en categorías.
- produces y consumes : Define los tipos de contenido que maneja el endpoint.
- parameter : Define los parámetros de entrada, como los de path o body.
- response : Define los posibles códigos de respuesta y sus esquemas.
- schema : Hace referencia a los esquemas definidos en swagger_helper.rb
- run_test! : Ejecuta la prueba y genera la documentación.

4. Generar y Visualizar la Documentación

Ejecutar las Pruebas para Generar la Documentación

Para generar la documentación Swagger, debes ejecutar las pruebas de Rspec que has definido.

```
bundle exec rspec
```

Este comando ejecutará las pruebas y generará los archivos de documentación en el directorio swagger/ . Por ejemplo, swagger/v1/swagger.yaml .

Iniciar el Servidor Rails

Asegúrate de que tu servidor Rails esté corriendo:

```
rails s
```

Acceder a la Documentación Interactiva

Abre tu navegador y navega a:

http://localhost:3000/api-docs

Verás la interfaz de Swagger UI con toda la documentación generada automáticamente. Desde aquí, puedes interactuar con tus endpoints, ver detalles de las solicitudes y respuestas, y probar la API directamente desde la interfaz.

Documentación para API de Estudiantes

Endpoints principales de la API de estudiantes:

1. Listar Estudiantes

- Método: GET
- URL: /estudiantes
- Descripción: Devuelve una lista de todos los estudiantes registrados.
- Respuesta Exitosa (200 OK):
  json

```
[
 {
  "id": 1,
  "nombres": "Gema Mercedes",
  "apellidos": "Bravo Lanzas",
  "carrera": "Ingeniería Telemática",
  "carnet": "21-03968-1",
  "created_at": "2024-10-10T15:00:00.000Z",
  "updated_at": "2024-10-10T15:00:00.000Z"
 },
 {
  "id": 2,
  "nombres": "Carlos Alberto",
  "apellidos": "González López",
  "carrera": "Ingeniería Electrónica",
  "carnet": "21-03968-2",
  "created_at": "2024-10-10T15:05:00.000Z",
  "updated_at": "2024-10-10T15:05:00.000Z"
 }
]
```

2. Crear un Estudiante

- Método: POST
- URL: /estudiantes
- Descripción: Crea un nuevo estudiante en la base de datos.
- Parámetros de Entrada (Body - JSON):
  json

```
{
 "nombres": "Gema Mercedes",
 "apellidos": "Bravo Lanzas",
 "carrera": "Ingeniería Telemática",
 "carnet": "21-03968-1"
}
```

- Respuesta Exitosa (201 Created):
  json

{
"id": 1,
"nombres": "Gema Mercedes",
"apellidos": "Bravo Lanzas",
"carrera": "Ingeniería Telemática",
"carnet": "21-03968-1",
"created_at": "2024-10-10T15:00:00.000Z",
"updated_at": "2024-10-10T15:00:00.000Z"
}

- Errores Posibles:
 - 422 Unprocessable Entity:
  json
 {
  "errors": {
  "carnet": ["ya está en uso"],
  "nombres": ["no puede estar vacío"]
  }
 }

3. Obtener un Estudiante por ID

- Método: GET
- URL: /estudiantes/{id}
- Descripción: Devuelve los datos de un estudiante específico identificado por su id .
- Parámetros de Entrada:
 - Path Parameter: id (integer) - ID del estudiante.
- Respuesta Exitosa (200 OK):
 json
 {
 "id": 1,
 "nombres": "Gema Mercedes",
 "apellidos": "Bravo Lanzas",
 "carrera": "Ingeniería Telemática",
 "carnet": "21-03968-1",
 "created_at": "2024-10-10T15:00:00.000Z",
 "updated_at": "2024-10-10T15:00:00.000Z"
 }

- Errores Posibles:
 - 404 Not Found:
  json
 {
  "errors": {
  "id": ["No se encontró el estudiante"]
  }
 }

4. Actualizar un Estudiante

- Método: PUT
- URL: /estudiantes/{id}
- Descripción: Actualiza los datos de un estudiante existente.
- Parámetros de Entrada:
 - Path Parameter: id (integer) - ID del estudiante.
 - Body (JSON):
  json
 {
  "nombres": "Gema M.",
  "apellidos": "Bravo L.",
  "carrera": "Ingeniería de Sistemas"
 }

- Respuesta Exitosa (200 OK):
 json
 {
 "id": 1,
 "nombres": "Gema M.",
 "apellidos": "Bravo L.",
 "carrera": "Ingeniería de Sistemas",
 "carnet": "21-03968-1",
 "created_at": "2024-10-10T15:00:00.000Z",
 "updated_at": "2024-10-10T16:00:00.000Z"
 }

- Errores Posibles:
 - 404 Not Found:
  json
 {
  "errors": {
  "id": ["No se encontró el estudiante"]
  }
 }

 - 422 Unprocessable Entity:
  json
 {
  "errors": {
  "nombres": ["no puede estar vacío"]
  }
 }

5. Eliminar un Estudiante

- Método: DELETE
- URL: /estudiantes/{id}
- Descripción: Elimina un estudiante de la base de datos.
- Parámetros de Entrada:
 - Path Parameter: id (integer) - ID del estudiante.
- Respuesta Exitosa (204 No Content):
 - No hay contenido en la respuesta.
- Errores Posibles:
 - 404 Not Found:
 json
 {
 "errors": {
 "id": ["No se encontró el estudiante"]
 }
 }

6. Consideraciones Adicionales

a. Autenticación (Si Aplica)

Si tu API requiere autenticación, debes documentar cómo los usuarios deben autenticarse. Por ejemplo, si usas tokens JWT:

- Método de Autenticación: Bearer Token
- Encabezado Requerido:

Authorization: Bearer <token>