

UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA



UNAN - León

FACULTAD DE CIENCIAS Y TECNOLOGIA

Carrera: Ingeniería en Telemática

Componente: Software como un servicio

Grupo:1

Docente: Erving Montes

Elaborado por:

- Harold Steven García Ramos
- Alli Gissie Herrera Chow.
- Ismael Noe Sánchez Guido

Fecha: 06 de octubre de 2024

Introducción

ReadHereAPI es una API diseñada para facilitar la búsqueda y descarga de libros. Permite a los usuarios encontrar el libro que desean leer y descargarlo en formato PDF. Esta API es de solo consumo, lo que significa que solo se admiten solicitudes HTTP GET.

Requisitos Previos

No se requiere autenticación para acceder a la API. Puedes realizar solicitudes utilizando herramientas como Postman, cURL o cualquier cliente HTTP de tu elección.

GET /libros

Permite buscar libros en la base de datos.

Parámetros

titulo (opcional): Título del libro que deseas buscar.

autor (opcional): Nombre del autor del libro.

genero(opcional): Categoría en la que deseas buscar el libro.

Ejemplo de Uso

GET <https://localhost:3000/books>

Respuesta

```
{
  "libros": [
    {
      "id": 1,
      "Titulo": "El Principito ",
      "Autor": " Antoine de Saint-Exupéry",
      "Descripcion": "cuento infantil"
```

```
"Genero": " literatura infantil, fabula, novela corta, ficción especulativa ",
"Fecha_publicacion": "1994-05-25"
"link": " https://drive.google.com/file/d/1-uTvhK-sFB8WHJ7ccPb\_zSTYZk\_i42Bn/view?usp=drive\_link"
```

```
"created_at": 2024:1003T22:29:17:501Z
"updated_at": 2024:1003T22:29:17:501Z
```

```
}
]
}
```

```
require 'net/http'
require 'json'
require 'uri'
```

```
uri = URI("https://localhost:3000/books")
response = Net::HTTP.get_response(uri)
```

```
if response.is_a?(Net::HTTPSuccess)
  libros = JSON.parse(response.body)
  puts libros
else
  puts "Error al buscar libros: #{response.code}"
end
```

Manejo de Errores

La API puede devolver diferentes códigos de estado HTTP. A continuación se presentan algunos de los errores más comunes:

404 Not Found: El recurso solicitado no fue encontrado.

500 Internal Server Error: Error en el servidor. Intenta nuevamente más tarde.

Normas de Conducta

Te invitamos a ser amable y respetuoso con otros desarrolladores que utilizan ReadHereAPI. La colaboración y el respeto son fundamentales para mantener una comunidad positiva.

Reportar Vulnerabilidades

Si detectas alguna vulnerabilidad de seguridad en la API, actúa de manera responsable y repórtala a nuestro equipo de soporte.

FAQ (Preguntas Frecuentes)

¿Es gratuita la API? Sí, ReadHereAPI es completamente gratuita.

¿Necesito autenticación para usar la API? No, no se requiere autenticación.

Contacto

Para más información o soporte, por favor contacta a: support@readhere.com.

Código

En resumen, este código configura rutas RESTful para un recurso de libros y proporciona una ruta de verificación de estado para la aplicación. Esto es fundamental para el manejo de las solicitudes en una aplicación Rails y para asegurar su disponibilidad.

```
Rails.application.routes.draw do
```

```
  # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html
```

```
  resources :books
```

```
  # Reveal health status on /up that returns 200 if the app boots with no exceptions, otherwise 500.
```

```
  # Can be used by load balancers and uptime monitors to verify that the app is live.
```

```
  get "up" => "rails/health#show", as: :rails_health_check
```

```
  # Defines the root path route ("/")
```

```
  # root "posts#index"
```

End

En resumen, este controlador maneja las operaciones CRUD para el recurso Book en una aplicación Rails. Permite a los usuarios obtener una lista de libros, ver detalles de un libro específico, crear nuevos libros, actualizar libros existentes y eliminar libros. El uso de JSON como formato de respuesta facilita la integración con aplicaciones front-end o APIs externas.

```
class BooksController < ApplicationController
```

```
  #GET /books
```

```
  def index
```

```
    books= Book.filter_by(params)
```

```
    render json: books
```

```
  end
```

```
  #GET /books/:id
```

```
  def show
```

```
    book = Book.find(params[:id])
```

```
    render json: book
```

```
  end
```

```
  #POST /books
```

```
  def create
```

```
    book = Book.new(book_params)

    if book.save

      render json: book, status: :created

    else

      render json: book.errors, status: :unprocessable_entity

    end

  end

end
```

```
#PUT /books/:id
```

```
def update

  book = Book.find(params[:id])

  if book.update(book_params)

    render json: book

  else

    render json: book.errors, status: :unprocessable_entity

  end

end

end
```

```
#DELETE /books/:id
```

```
def destroy
```

```

    book = Book.find(params[:id])

    book.destroy

    head :no_content

  end

  private

  def book_params

    params.require(:book).permit(:Titulo, :Autor, :Descripcion, :Fecha_publicacion,
:Genero, :link)

  end

end

```

index:

```

def index
  books = Book.filter_by(params)
  render json: books
end

```

- Este método responde a una solicitud GET en /books.
- Llama al método filter_by en el modelo Book para obtener una lista de libros filtrados según los parámetros proporcionados.
- Renderiza la lista de libros en formato JSON.

show:

```

def show
  book = Book.find(params[:id])
  render json: book
end

```

- Este método responde a una solicitud GET en /books/:id.
- Busca un libro específico por su ID y renderiza el libro en formato JSON.

create:

```
def create
  book = Book.new(book_params)
  if book.save
    render json: book, status: :created
  else
    render json: book.errors, status: :unprocessable_entity
  end
end
```

- Este método maneja una solicitud POST a /books.
- Crea un nuevo libro con los parámetros permitidos.
- Si se guarda correctamente, devuelve el libro con un estado 201 Created.
- Si falla, devuelve los errores de validación con un estado 422 Unprocessable Entity.

update:

```
def update
  book = Book.find(params[:id])
  if book.update(book_params)
    render json: book
  else
    render json: book.errors, status: :unprocessable_entity
  end
end
```

- Este método maneja una solicitud PUT a /books/:id.
- Busca el libro por ID y actualiza sus atributos.
- Si la actualización es exitosa, devuelve el libro actualizado.
- Si falla, devuelve los errores.

destroy:

```
def destroy
  book = Book.find(params[:id])
  book.destroy
  head :no_content
end
```

- Este método maneja una solicitud DELETE a /books/:id.
- Busca el libro por ID y lo elimina de la base de datos.

- Devuelve un estado 204 No Content para indicar que la operación se realizó correctamente.

Método privado book_params:

```
private
```

```
def book_params
```

```
  params.require(:book).permit(:Titulo, :Autor, :Descripcion, :Fecha_publicacion, :Genero, :link)
end
```

- Este método se encarga de permitir solo los parámetros específicos que se pueden usar para crear o actualizar un libro.
- Asegura que no se puedan enviar parámetros no deseados, lo que ayuda a proteger la aplicación.

Este código define el modelo Book en una aplicación Rails. Se encarga de las validaciones para asegurar que ciertos campos estén presentes al crear o actualizar libros y proporciona un método para filtrar libros según los parámetros especificados en las consultas. Esto hace que la búsqueda de libros sea más flexible y eficiente, permitiendo a los usuarios encontrar libros que coincidan con criterios específicos.

```
class Book < ApplicationRecord
```

```
  #validaciones
```

```
  validates :Titulo, :Autor, :Fecha_publicacion, :Genero, presence: true
```

```
  #Metodo para filtrar
```

```
  def self.filter_by(params)
```

```
    books= Book.all
```

```
    books = books.where("Titulo LIKE ?", "%#{params[:Titulo]}") if params[:Titulo].present?
```

```
    books books.where("Autor LIKE ?", "%#{params[Autor]}") if params[:Autor].present?
```

```
books = books.where("Fecha publicacion LIKE ?", "%#{params[:Fecha publicacion]}") if  
params[:Fecha_publicacion]
```

```
books = books.where( "Genero LIKE ?", "%#{params: Genero}") if  
params[:Genero].present?
```

```
books
```

```
end
```

```
end
```

□ **Validaciones:**

```
validates :Titulo, :Autor, :Fecha_publicacion, :Genero, presence: true
```

Esto asegura que cada libro debe tener un título, un autor, una fecha de publicación y un género. Si alguno de estos atributos falta, el libro no se guardará en la base de datos.

□ **Método de filtrado:**

```
def self.filter_by(params)
```

Este método permite filtrar los libros en función de los parámetros que se le pasen. Los parámetros pueden incluir el título, el autor, la fecha de publicación y el género.

□ **Lógica de filtrado:**

Se inicializa una colección de todos los libros:

```
books = Book.all
```

Luego, se aplican condiciones a esta colección dependiendo de los parámetros que estén presentes:

```
books = books.where("Titulo LIKE ?", "%#{params[:Titulo]}%") if  
params[:Titulo].present?
```

Aquí se busca en la columna Titulo de los libros aquellos que coincidan con el patrón proporcionado en `params[:Titulo]`.

Hay otros filtros para el autor, la fecha de publicación y el género, aunque hay errores en la sintaxis que deben corregirse.

En resumen, este código se utiliza para poblar la base de datos de la aplicación Rails con registros iniciales de libros. Esto es útil para el desarrollo y las pruebas, proporcionando una base de datos con datos significativos y variados desde el principio.

```
# This file should ensure the existence of records required to run the application in every environment (production,
```

```
# development, test). The code here should be idempotent so that it can be executed at any point in every environment.
```

```
# The data can then be loaded with the bin/rails db:seed command (or created alongside the database with db:setup).
```

```
#
```

```
# Example:
```

```
#
```

```
# ["Action", "Comedy", "Drama", "Horror"].each do |genre_name|
```

```
#   MovieGenre.find_or_create_by!(name: genre_name)
```

```
# end
```

```
Book.create([
```

```
{
```

```
  Titulo: "El duque y yo",
```

```
  Autor: "Julia Quinn",
```

Descripcion: "Novela de romance historico",

Fecha_publicacion: "2000-05-01",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-BCFCbOtYDMcdrS9Xm64ZF7BWdsV00Je/view?usp=drive_link"

},

{

Titulo:"El vizconde que me amo",

Autor: "Julia Quinn",

Descripcion: "Novela de romance de ficcion",

Fecha_publicacion: "2000-12-05",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-BIckoHd1XBxuldS_1WH5j9zw_HYKmjw/view?usp=drive_link "

},

{

Titulo: "Te doy mi corazon",

Autor: "Julia Quinn",

Descripcion: "Novela de Romance de regerencia",

Fecha_publicacion: "2001-06-03",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-K6T8yc-4zVo2OblQjbLhx67nZ3gZcR7/view?usp=drive_link "

},

{

Titulo: "Seduciendo a Mr.Bridgerton",

Autor: "Julia Quinn",

Descripcion: "Novela de ficcion sobre un romance prohibido",

Fecha_publicacion: "2002-10-18",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-hDsHVyF2c_BcEsWVcCC1c1woFThtd_0/view?usp=drive_link "

},

{

Titulo: "A sir Phillip, con amor",

Autor: "Julia Quinn",

Descripcion: "El amor de una joven florece cada dia mas con cada carta",

Fecha_publicacion: "2003-07-01",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-D5iaqrBfO3Q-ztmamwHwQQjziU2IZia/view?usp=drive_link "

},

{

Titulo: "El corazon de una Bridgerton",

Autor: "Julia Quinn",

Descripcion: "Los sentimientos de una joven saldran a la luz",

Fecha_publicacion: "2004-06-29",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-msRCaqakvNpYEFcXIT-5WhYA42S5I9A/view?usp=drive_link "

},

{

Titulo: "Por un beso",

Autor: "Julia Quinn",

Descripcion: "Es una historia de amor en un entorno ficticio",

Fecha_publicacion: "2007-03-25",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-FFfe1ulYdzZpxercshz76KS_bZnExFO/view?usp=drive_link "

},

{

Titulo: "Buscando esposa",

Autor: "Julia Quinn",

Descripcion: "Es una novela de romance con estadia en inglaterra",

Fecha_publicacion: "2006-06-27",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-KtzxztVyfwFpMqAnyyX17yadjVi9ViW/view?usp=drive_link"

},

{

Titulo: "Brigerton felices para siempre",

Autor: "Julia Quinn",

Descripcion: "Una pareja busca superar los obstaculos y ser felices",

Fecha_publicacion: "2013-04-02",

Genero: "Romance",

link: "https://drive.google.com/file/d/1-EKyK5uy9vAgdj7AP-c_60O4aJBnUaft/view?usp=drive_link "

},

{

Titulo: "Diarios",

Autor: "Alejandra Pizarnmik",

Descripcion: "Un libro lleno de poemas",

Fecha_publicacion: "2010-12-01",

Genero: "Poesia",

link: "https://drive.google.com/file/d/1--dihTleD_BKfpUNGv9SrU8A6hvccFE1/view?usp=drive_link "

},

{

Titulo: "Poesia Completa",

Autor: "Alejandra Pizarnmik",

Descripcion: "Un libro el cual lleva muchos poemas bonitos",

Fecha_publicacion: "2001-12-01",

Genero: "Poesia",

link: "https://drive.google.com/file/d/1-1X8Aj6Ff7-SR7B_gBIomNc1HMxsoGSA/view?usp=drive_link "

},

{

Titulo: "Diario de Ana Frank",

Autor: "Ana Frank",

Descripcion: "Una historia de los dias que paso una joven durante tiempos oscuros",

Fecha_publicacion: "1947-06-25",

Genero: "Autobiografia",

link: "https://drive.google.com/file/d/1-p6vIbTJSiu6tpmj7sW8wWiuxIH0DW6i/view?usp=drive_link "

},

{

Titulo: "Noches Blancas",

Autor: "Fiodor Dostoyevsk",

Descripcion: "Una historia de ficcion muy entretenida",

Fecha_publicacion: "1848-06-25",

Genero: "Ficcion",

link: "https://drive.google.com/file/d/1-5TCbVIVeYivyxYIn-YDU9eIdeUooFmy/view?usp=drive_link "

},

{

Titulo: "Las mujeres que aman demasiado",

Autor: "Robin Norwood",

Descripcion: "Libro que planea ayudar a las mujeres",

Fecha_publicacion: "1985-10-25",

Genero: "Libro de autoayuda",

link:

"https://drive.google.com/file/d/17blpUdQ1XoCR1v8sYBADjzCPvhUYUSOp/view?usp=drive_link "

},

{

Titulo: "Harry Potter y la piedra filosofal",

Autor: "J.K.Rowling",

Descripcion: "Una historia de magia, ambientada en un mundo donde los magos viven en secreto",

Fecha_publicacion: "1997-06-26",

Genero: "Fantasia",

link:

"https://drive.google.com/file/d/1-qxxAKfUv-MAIK42DMGyPX5IWCLRFr4O/view?usp=drive_link "

},

{

Titulo: "Cincuenta sombras liberadas",

Autor: "E.L.James",

Descripcion: "Una historia de un hombre millonaria y su vida sexual",

Fecha_publicacion: "2012-04-17",

Genero: "Erotico",

link: "https://drive.google.com/file/d/1-uKCMYdPlil_woY870Uw5iSoWcYhDvBF/view?usp=drive_link "

},

{

Titulo: "El principito",

Autor: "Antoine de Saint-Exupery",

Descripcion: "Es un niño que habita en un asteroide lejano",

Fecha_publicacion: "1943-04-06",

Genero: "Infantil",

link: "https://drive.google.com/file/d/1-uTvhK-sFB8WHJ7ccPb_zSTYZk_i42Bn/view?usp=drive_link "

},

{

Titulo: "La guerra de los mundos",

Autor: "H.G.Wells",

Descripcion: "Equipados con terribles maquinas, incia una invasion",

Fecha_publicacion: "1898-08-23",


Genero: "Ciencia Ficción",

link: "https://drive.google.com/file/d/1-xZFyQb7I97Ry6RQsdfHzu7MCyLMQoXx/view?usp=drive_link "

}

D)

Funcionamiento de API en POSTMAN

 http://localhost:3000/books

GET

http://localhost:3000/books

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

	Key	Value	Bulk Edit
	Key	Value	

Body

Cookies

Headers (13)

Test Results

200 OK

347 ms

6.89 KB

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "id": 1,
3    "Titulo": "El duque y yo",
4    "Autor": "Julia Quinn",
5    "Descripcion": "Novela de romance historico",
6    "Fecha_publicacion": "2000-05-01",
7    "Genero": "Romance",
8    "link": "https://drive.google.com/file/d/1-BCFCb0tYDMcdrS9Xm64ZF7BWdsV00Je/view?usp=drive_link",
9    "created_at": "2024-10-03T22:29:17.501Z",
10   "updated_at": "2024-10-03T22:29:17.501Z"
11  },
12  {
13  }
```

GET

http://localhost:3000/up

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

	Key	Value
	Key	Value

BodyCookiesHeaders (13)Test Results

200 OK317 ms65

PrettyRawPreviewVisualizeHTML

```
1 <!DOCTYPE html>
2 <html>
3
4 <body style="background-color: green"></body>
5
6 </html>
```

GET

http://localhost:3000/books/16

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

	Key	Value	Bulk Edit
	Key	Value	

BodyCookiesHeaders (13)Test Results

200 OK41 ms979 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 16,
3   "Titulo": "Cincuenta sombras liberadas",
4   "Autor": "E.L.James",
5   "Descripcion": "Una historia de un hombre millonaria y su vida sexual",
6   "Fecha_publicacion": "2012-04-17",
7   "Genero": "Erotico",
8   "link": "https://drive.google.com/file/d/1-uKCMYdPl1l_woY870Uw5iSoWcYhDvBF/view?usp=drive_link ",
9   "created_at": "2024-10-03T22:29:17.543Z",
10  "updated_at": "2024-10-03T22:29:17.543Z"
11 }
```

POST

▼

http://localhost:3000/books

ParamsAuthorizationHeaders (8)Body ●Pre-request ScriptTestsSettings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

JSON ▼

10 }
11

bodyCookiesHeaders (13)Test Results

🌐 201 Created 100 ms

PrettyRawPreviewVisualizeJSON ▼

≡

1
2 "id": 19,
3 "Titulo": "Nuevo Libro",
4 "Autor": "Autor Ejemplo",
5 "Descripcion": "Descripción del libro",
6 "Fecha_publicacion": "2023-01-01",
7 "Genero": "Ficción",
8 "link": "https://drive.google.com/uc?id=ID_DEL_LIBRO",
9 "created_at": "2024-10-04T18:43:31.855Z",
10 "updated_at": "2024-10-04T18:43:31.855Z"
11

HTTP http://localhost:3000/books

PUT

http://localhost:3000/books/19

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ▼

3	"Titulo": "Nuevo Libro Actualizado",
4	"Autor": "Autor Ejemplo",

Body Cookies Headers (13) Test Results

200 OK

Pretty

Raw

Preview

Visualize

JSON ▼



```
1 {
2   "Titulo": "Nuevo Libro Actualizado",
3   "Autor": "Autor Ejemplo",
4   "Descripcion": "Descripción del libro",
5   "Fecha_publicacion": "2023-01-01",
6   "Genero": "Ficción",
7   "link": "https://drive.google.com/uc?id=ID_DEL_LIBRO",
8   "id": 19,
9   "created_at": "2024-10-04T18:43:31.855Z",
10  "updated_at": "2024-10-04T18:45:32.024Z"
11 }
```

DELETE

http://localhost:3000/books/19

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ▼

3	"Titulo": "Nuevo Libro Actualizado",
4	"Autor": "Autor Ejemplo",

Body Cookies Headers (9) Test Results

204 No Content 40 r

Pretty

Raw

Preview

Visualize

Text ▼



1

GET

http://localhost:3000/books

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryJSON

3

"Titulo": "Nuevo Libro Actualizado",

4

"Autor": "Autor Ejemplo",

odyCookiesHeaders (13)Test Results200 OK29 ms6.88 KBSave Response

PrettyRawPreviewVisualizeJSON

188

}

189

{

190

"id": 18,

191

"Titulo": "La guerra de los mundos",

192

"Autor": "H.G.Wells",

193

"Descripcion": "Equipados con terribles maquinas, incia una invasion",

194

"Fecha_publicacion": "1898-08-23",

195

"Genero": "Ciencia Ficcion",

196

"link": "https://drive.google.com/file/d/1-xZFyQb7I97Ry6RQsdfHzu7MCyLMQoXx/view?usp=drive_link",

197

"created_at": "2024-10-03T22:29:17.548Z",

198

"updated_at": "2024-10-03T22:29:17.548Z"

199

}

200

GET

http://localhost:3000/books?Titulo=El%20duque%20y%20yo

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Titulo	El%20duque%20y%20yo	

BodyCookiesHeaders (13)Test Results200 OK1112 ms947 BSave Respons

PrettyRawPreviewVisualizeJSON

1

2

{

3

"id": 1,

4

"Titulo": "El duque y yo",

5

"Autor": "Julia Quinn",

6

"Descripcion": "Novela de romance historico",

7

"Fecha_publicacion": "2000-05-01",

8

"Genero": "Romance",

9

"link": "https://drive.google.com/file/d/1-BCFCb0tYDMcdR59Xm64ZF7BWdsV00Je/view?usp=drive_link",

10

"created_at": "2024-10-03T22:29:17.501Z",

11

"updated_at": "2024-10-03T22:29:17.501Z"

12

}

13

GET

http://localhost:3000/books?Autor=Julia%20Quinn

Params • Authorization Headers (8) Body • Pre-request Script Tests Settings

Query Params

	Key	Value
<input checked="" type="checkbox"/>	Autor	Julia%20Quinn

body Cookies Headers (13) Test Results 200 OK 261 ms 3.69 KB

Pretty Raw Preview Visualize JSON

```
3      "id": 1,
4      "Titulo": "El duque y yo",
5      "Autor": "Julia Quinn",
6      "Descripcion": "Novela de romance historico",
7      "Fecha_publicacion": "2000-05-01",
8      "Genero": "Romance",
9      "link": "https://drive.google.com/file/d/1-BCFCb0tYDMcdiS9Xm64ZF7BWdsV00Je/view?usp=drive_link",
10     "created_at": "2024-10-03T22:29:17.501Z",
11     "updated_at": "2024-10-03T22:29:17.501Z"
12   },
13   {
14     "id": 2,
15     "Titulo": "El vizconde que me amo",
```

GET

http://localhost:3000/books?Genero=Fantasia

Send

Params • Authorization Headers (8) Body • Pre-request Script Tests Settings

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Genero	Fantasia	

body Cookies Headers (13) Test Results 200 OK 38 ms 1016 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    {
3      "id": 15,
4      "Titulo": "Harry Potter y la piedra filosofal",
5      "Autor": "J.K.Rowling",
6      "Descripcion": "Una historia de magia, ambientada en un mundo donde los magos viven en secreto",
7      "Fecha_publicacion": "1997-06-26",
8      "Genero": "Fantasia",
9      "link": "https://drive.google.com/file/d/1-qxxAKfUv-MA1K42DMGyPX5IWCLRFr40/view?usp=drive_link",
10     "created_at": "2024-10-03T22:29:17.540Z",
11     "updated_at": "2024-10-03T22:29:17.540Z"
12   }
13 }
```