Universidad ORT Uruguay
Facultad de Ingeniería

Obligatorio 1 -Diseño de aplicaciones 2 - Evidencia del diseño y especificación de la API.

Nicolas Hernandez – 229992
Francisco Aguilar – 230143

2023

Índice

Abstract

En este documento se encuentra toda la información necesaria acerca del diseño y especificación de la API creada para el obligatorio de diseño de aplicaciones 2. Aquí se puede encontrar los criterios REST utilizados durante la creación de la API, los diferentes códigos de estatus utilizados en la api para las respuestas a cada request y la descripción de los diferentes recursos de la API.

Link Documentacion Swagger

Utilizamos la herramienta de swagger, donde se pueden encontrar todos los endpoints de la api, junto con los diferentes schemas y el json de la api en el cuál se puede encontrar más información como por ejemplo el tipo de valor que utilizamos en cada elemento del body.
https://app.swaggerhub.com/apis/NicolasAHF/blog-web_api/1.0

Criterios seguidos para asegurar que la API cumple con los criterios REST

Utilizamos diferentes métodos HTTP para la solución de nuestra API, dentro de los métodos utilizados se encuentran, GET, POST, PUT y DELETE. También seguimos con el criterio REST de tener URLs orientadas a recursos osea nuestros recursos son acordes al uso de la API, ya que es una API sobre un blog y nuestros recursos son: users, articles, comments y auth. Cada recurso define explícitamente sobre que se trata

Descripción del mecanismo de autenticación

Para la autenticación de los usuarios creamos filtros, un filtro de encarga de verificar cuando un usuario está logueado y le permite realizar ciertas funciones en base de si esta logueado o no, Otro filtro utilizado fue el de chequeo de roles de los diferentes usuarios, ya que los usuarios pueden tener roles de admin y/o blogger,

este filtro se encarga de verificar que el usuario cuente con el rol correcto para realizar la acción que está queriendo hacer.

Descripción general de códigos de status

Lo códigos de estatus que utilizamos en nuestra API REST fueron los siguientes:

- 200 - Ok: este código lo utilizamos en los momentos en que un método GET generalmente retornaba un resultado válido. Aunque en nuestro caso lo utilizamos también en otros métodos como en el POST en el momento que se loguea un usuario exitosamente, al actualizar usuarios o artículos y también en los métodos de DELETE.
- 201 - Created: Este código lo utilizamos en diferentes métodos POST como por ejemplo, la creación de usuarios, artículos y comentarios.
- 400 - Bad Request: Este código fue utilizado en los momentos en los que el usuario ingresaba algún valor no válido en las diferentes request.
- 401 - Unauthorized: Este código de estatus lo utilizamos en los casos de logueo no exitoso por parte de los usuarios
- 404 - Not Found: Utilizamos este código de estatus para momentos en los que los diferentes elementos de la API Rest no eran encontrados, como por ejemplo al buscar un artículo con un id válido que no existe.
- 500 - Internal Server Error: Este código lo utilizamos en el filtro de excepciones como el código de error genérico para códigos que no soportamos en nuestra API.

Descripción de los recursos de la API

URL Base

https://localhost:7105/api

Recursos

Artículos

- GET - /api/articles/{id}
    - Este recurso se encarga de obtener un artículo a partir de su id

    - Se le pasa un id como parámetro
    - Respondió un código de estatus de 200 ok en caso válido y un 404 not found en caso de no encontrar un artículo
    - Se le manda un header un Authorization ya que hay que estar logueado para poder llamar al endpoint
- PUT - /api/articles/{id}
    - Se encarga de modificar un artículo a partir de su id
    - Se le pasa un id de artículo y el id de Authorization como parámetros, y en el body se le pasa el artículo con sus cambios.
    - Responde un 200 ok en caso válido, un 404 not found en caso de no encontrar el artículo y un 400 bad request en caso de mal enviado el body
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- DELETE - /api/articles/{id}
    - Se encarga de borrar un artículo a partir de su id
    - Se le para el id del artículo y el id de Authorization como parámetros
    - Responde un 200 ok en caso válido y un 404 not found en caso de no encontrar el artículo
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- GET - /api/articles/public
    - Obtiene todos los artículos públicos
    - No se le pasan parámetros
    - Responde un 200 ok en caso válido y un 404 not found en caso de no haber articulos
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- GET - /api/articles?username=
    - Obtiene todos los artículos de un usuario, si el usuario logueado es el usuario que hizo la request se agregan también los artículos privados
    - se le pasa por query string el username del usuario y por header el token de Authorization

    - Responde un 200 ok en caso válido y un 404 not found en caso de no haber artículos y un 400 bad request en caso de un username no válido
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- POST - /api/articles
    - Crea un artículo
    - Se le pasa por header el token de authorization y por body el articulo a crear
    - Responde un 200 ok en caso de creación válida y un 400 bad request en caso de que se falle en alguno de los campos
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- GET - /api/articles/search?text=
    - Obtiene todos los artículos que contienen un texto en el título o en el contenido del mismo
    - Se le pasa el texto por query string
    - Responde un 200 ok en caso de encontrar artículos y un 404 not found en caso de que no existan artículos con ese texto
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- GET - /api/articles/last-articles
    - Obtiene los ultimos de articulos públicos creados
    - No se le pasan parámetros
    - Responde un 200 ok en caso de encontrar artículos y un 404 not found en caso de que no existan artículos
    - No se le manda un header ya que no requiere authorization

Auth

- POST - /api/auth/register
    - Se encarga del registro de usuarios
    - Se le pasa por body el primer nombre, segundo nombre, username, contraseña y email.

    - Responde un 200 ok en caso de registro exitoso y un 400 bad request en caso de que el usuario no haya ingresado un campo válido
    - No se le manda un header ya que no requiere authorization
- POST - api/auth/login
    - Se encarga de logueo del usuario
    - Se le pasa por body el username y la contraseña
    - Responde un 200 ok en caso de logueo válido y un 401 unauthorize en caso de credenciales inválidas
    - No se le manda un header ya que no requiere authorization
- POST - /api/auth/logout
    - Se encarga de desloguear al usuario del sistema
    - Como parametro se le pasa por header el token de Authorization
    - Responde 200 ok en caso de deslogueo valido y 400 bad request en caso de error en el token
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso

Comment

- POST - /api/comments
    - Añade un comentario a un artículo
    - Se le pasa como parámetro por header el token de authorization y por body los campos articleId y body que es el cuerpo del comentario
    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar al artículo
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- PUT - /api/comments
    - Se utiliza para responder a cierto comentario
    - Se le pasa por parámetro por el body el commentId y la reply
    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar al comentario
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso

- DELETE - /api/comments/{id}
    - Se utiliza para borrar un comentario
    - Se le pasa por url el id del comentario
    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar al comentario
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso

User

- GET - /api/users/{id}
    - Obtiene a un usuario a partir de su id
    - Se le pasa por ruta el id del usuario y por heder el token de authorization
    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar al usuario
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- PUT - /api/users/{id}
    - Actualiza un usuario a partir de su id, si el usuario logueado no es el mismo que el que manda la request se verifica si es admin, en caso de que sea el mismo usuario logueado se permite modificación
    - Se le pasa por ruta el id del usuario y por heder el token de authorization, además por el body se le pasa todo lo necesario para modificar al usuario
    - Responde 200 ok en caso de deslogueo válido, 404 not found en caso de no encontrar al usuario y 400 bad request en caso de que algun campo no sea valido
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- DELETE - /api/users/{id}
    - Borra un usuario a partir de su id, este método solo es permitido para admins
    - Se le pasa por ruta el id del usuario a borrar

    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar al usuario.
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- GET - /api/users/ranking
    - Obtiene a los usuarios con mas comentarios y articulos creados entre 2 fechas especificas, se requiere ser admin para utilizar este recurso
    - Se le pasa por query la fecha de inicio y la fecha de fin del filtrado
    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar usuarios en el sistema
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- GET - /api/users
    - Obtiene todos los usuarios existentes en el sistema
    - No se le pasan parámetros
    - Responde 200 ok en caso de deslogueo válido y 404 not found en caso de no encontrar usuarios en el sistema
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso
- POST - /api/users
    - Crea un usuario, solo los admins pueden usar este método
    - Se le pasa por body el usuario con los campos de primer nombre, apellido, username, contraseña, roles y email
    - Responde 200 ok en caso de deslogueo válido y 400 bad request en caso de que se ingrese un campo no válido
    - Se le manda un header de Authorization ya que hay que estar logueado para usar el recurso