Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de Aplicaciones 2

Obligatorio 1

BlogsApp

Fernando Spillere - 274924

Gimena Alamón - 243518

Carmela Sotuyo - 186554

Mayo 2023

Índice

Diseño general de la API

Para cumplir con el desarrollo de una API RESTFul se siguen una serie de criterios que sugiere este estilo arquitectónico. Nos apegamos a los mismos de la siguiente manera:

La interfaz que se expone al usuario es uniforme en cuanto a verbos HTTP utilizados y URIs definidas para cada recurso. Se utilizan los verbos Post ,Get, Put, Patch y Delete.

Es ”stateless”, lo cual significa que no guarda un estado, sino que cada request realizada por el cliente es independiente de la anterior. Es importante destacar que en cada petición (Menos cuando se trata de crear al usuario) en el header de la request se debe incluir el campo ”Autorización” y especificar del Token del usuaurio.

En base a este token la aplicación identifica cuál es el usuario que está realizando la petición y evalúa si dispone de los permisos necesarios para las operaciones que intenta realizar.

A su vez, se tuvo en cuenta que las URIs no tengan más e 3 niveles, que los nombres de los recursos sean siempres los mismos, es decir, no se usan sinónimos, y siempre se retorna el recurso que está más a la derecha.

Se puede concluír que se desarrolló la entonces el ”Backend”, es decir, la parte de servidor, de la arquitectura solicitada ”Cliente-Servidor”

Como pre-requisito para poder ejecutar la aplicación se deberá configurar el Connection string a la base de datos que se utilizará. Para esto se deberá agregar el mismo en el archivo de configuración (-../WebAPI/appsettings.json)

Registro de usuarios

Para registrarse en la aplicación, es decir, para crear un usuario se deberá realizar un POST al endpoint de user con la información necesaria. En caso de que alguno de los valores tenga un formato incorrecto, o que el json esté incompleto (falte algún atributo obligatorio), la aplicación devolverá un mensaje de error detallando el error. Una vez que el usuario es creado, podrá obtener de la response su Token de autorización, con el cual podrá loguearse y realizar peticiones.

Un usuario puede crearse a si mismo, así como también el usuario Admin podrá crear tantos usuarios desee.

Autorización de usuarios

Para obtener su token, un cliente deberé realizar un POST al endpoint de Sessions, lo cual devolverá en la response el token que deberá utilizar para sus siguientes peticiones.

Por tanto, para realizar peticiones a la aplicación, en cada request el cliente deberá incluir un header con key ”Authorization” y el valor del token.

Si el usuario desea cerrar su sesión deberá realizar un PATCH al endpoint de Sessions. En este caso el token generado anteriormente ya no le servirá para las próximas peticiones, sino que deberá volver a iniciar sesión.

Filtros - Manejo de errores

La aplicación tiene definidos una serie de mensajes de error, que se disparan en diversos momentos, dependiendo de si el error ocurre a nivel de servidor o a nivel de usuario. En todos los casos se trata de brindar al usuario la información necesaria para identificar lo sucedido, y si es posible, solucionarlo.

Los códigos y mensajes de error se detallan a continuación:

- 200 - Ok. Este código se retorna cuando la operación fue exitosa.

- 400 - Bad Request. Es un error a nivel del cliente. Se dispara cuando el usuario ingresa algún valor inválido.

- 401 - Unauthorized. Este mensajes es devuelto cuando el usuario que realiza la petición no tiene los permisos para realizar dicha acción.

- 403 - Forbidden Se dispara cuando el usuario intenta crear un objeto que ya existe.

- 404 - Not Found Este error es retornado cuando no se encontró la información solicitada por el usuario.

- 500 - Internal Server Error Este error es a nivel del servidor, sucede cuando existe algún fallo inesperado en la aplicación o base de datos.

DTOs

Se definen DTOs para el manejo de los recursos desde la especificación de la API. Se implementan tanto para Requests como para responses. El uso de DTOs facilitó la comunicación entre las peticiones realizadas por el cliente y la lógica de negocio, brindando flexibilidad a las mismas.

Se crean los siguientes:

- CreateUserRequestDTO

- UpdateUserRequestDTO

- ArticleConverter

- CreateArticleRequestDTO

- LoginRequestDTO

- LoginResponseDTO

- UpdateArticleRequestDTO

- UpdateUserRequestDTO

URIs

A continuación se detallan los endpoints existentes de la API con la información necesaria para poder ejecutarlos:

URI aplicable a todos: http://localhost:5050

6.1 User

- **POST /api/users**

  - Objetivo: Crear un usuario

  - Parámetros: Sin parámetros - No aplica Authorization Header con token.

  - Body:

  {

  ”username”: ”username”,

  ”email”: ”email@mail.com”,

  ”name”: ”name”,

  ”lastName”: ”lastName”,

  ”blogger”: true,

  ”admin”: false

  }

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Todos los campos anteriores son obligatorios.

  - Se controla que el email contenga un ”@” y finalice en ”.com”

  - Se controla que el username sea menor a 12 caracteres.

- Si el campo ”Blogger” tiene valor True implica que el usuario tendrá este rol. Si el campo ”Admin” tiene valor True implica que el usuario tendrá este rol. Si tiene ambos valores en True significa que tendrá ambos roles.

- **PATCH /api/users/{id}**

  - Objetivo: Modificar un usuario

  - Parámetros: Sin parámetros

  - Aplica Authorization Header con token.

  - Body:

  {

  ”username”: ”username”,

  ”email”: ”email@mail.com”,

  ”name”: ”name”,

  ”lastName”: ”lastName”,

  ”blogger”: true,

  ”admin”: false

  }

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Se puede enviar solo el atributo que se desea modificar, el resto no es necesario.

- **DELETE /api/users/{id}**

  - Objetivo: Eliminar un usuario

  - Parámetros: Sin parámetros - Aplica Authorization Header con token.

  - Body: Sin body

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

○ Consideraciones:

- Un usuario puede eliminarse a si mismo. Un admin puede eliminar a cualquier usuario.

- **GET /api/usuarios/ranking?fechaInicio=fechaInicio&fechaFin=fechaFin**

  - Objetivo: Obtener el ranking de usuarios con más actividad (artículos publicados más comentarios realizados) entre dos fechas.

  - Parámetros: fechaInicio y fechaFin - Aplica Authorization Header con token.

  - Body: Sin body

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Ambos parámetros son obligatorios.

  - Esta consulta solo para puede realizar un usuario admin.

6.2 Session

- **POST /api/sessions**

  - Objetivo: Log in.

  - Parámetros: Sin parámetros. - No aplica Authorization Header con token.

  - Body:

  {

  ”username”: ”aUser”,

  ”password”: ”aPass123”

  }

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

- En la response se incluirá el token con el cual el usuario podrá interacturar con el sistema. - Al loguearse, el usuario será notificado si tiene nuevos comentarios en alguno de sus artículos, es decir, comentarios que se hayan realizado a entre su último log out y su inicio de sesión actual. - Para responder a un comentario facilmente, basta con tomar su id, y ejecutar el endpoint de reply.

- **PATCH /api/sessions/id**

  - Objetivo: Log out.

  - Parámetros: Sin parámetros. - Aplica Authorization Header con token.

  - Body: Sin body

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Luego de hacer log out, el token anterior queda inválido. - El id que se debe incluir en la URI es de la session.

6.3 Article

- **GET /api/users/{id}/articles**

  - Objetivo: Obtener los articulos de un usuario.

  - Parámetros: fechaInicio y fechaFin - Aplica Authorization Header con token.

  - Body: Sin body

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Ambos parámetros son obligatorios. - Esta consulta solo para puede realizar un usuario admin.

- **GET /api/articles/{id}**

  - Objetivo: Obtener un artículo mediante su id.

- Parámetros: Sin parámetros. - Aplica Authorization Header con token.

- Body: Sin body

○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

○ Consideraciones:

- El artículo se devolverá siempre que el usuario tenga permisos para verlo, es decir, que sea el propierario o que el artículo sea público.

- **GET /api/articles/stats?year=year**

  - Objetivo: Obtener la distribución mensual de artículos en un año dado

  - Parámetros: year - Aplica Authorization Header con token.

  - Body: Sin body

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Solo el administrador tiene autorización para hacer esta consulta. - En la respuesta se muestra la cantidad de artículos, agrupados por mes.

- **GET /api/articles?search=search**

  - Objetivo: Obtener la lista de articulos que contienen un determinado texto

  - Parámetros: texto a buscar (search) - Aplica Authorization Header con token.

  - Body: Sin body

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - El artículo puede contener en cualquier parte de su definición el texto.

- **DELETE /api/articles/{id}**

  - Objetivo: Eliminar el artículo dado mediante el id.

- Parámetros: Sin parámetros - Aplica Authorization Header con token.

- Body: Sin body

○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

○ Consideraciones:

- Se realiza un borrado lógico en la Base de datos, agregando valor al atributo correspondiente de Fecha de eliminado. - Solo el propietario del artículo podrá eliminarlo.

- **PUT /api/articles/{id}**

- Objetivo: Actualizar el contenido de un artículo existente.

- Parámetros: Sin parámetros. - Aplica Authorization Header con token.

- Body:

{ ”name”: ”ArticleName”,

”body”: ”This is an article body”,

”template”: 1,

”image”: ”../img.png”,

}

○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

○ Consideraciones:

- Template es un enumerado que acepta los valores 1,2,3 y corresponde al tipo de plantilla que desea seleccionar.

- **POST /api/articles**

- Objetivo: Crear un artículo.

- Parámetros: Sin parámetros. - Aplica Authorization Header con token.

- Body:

{ ”name”: ”ArticleName”,

”body”: ”This is an article body”,

”private”: true

”template”: 1,

”image”: ”../img.png”,

}

○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

○ Consideraciones:

- Si el artículo es privado, implica que solo el usuario creador podrá verlo.

- **GET /api/articles**

- Objetivo: Crear un artículo.

- Parámetros: Sin parámetros. - Aplica Authorization Header con token.

- Body:

{ ”name”: ”ArticleName”,

”body”: ”This is an article body”,

”private”: true

”template”: 1,

”image”: ”../img.png”,

}

○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

○ Consideraciones:

- Si el artículo es privado, implica que solo el usuario creador podrá verlo.

6.4 Comment

- **POST /api/comments**

- Objetivo: Realizar un comentario en un artículo.

- Parámetros: Sin parámetros. - Aplica Authorization Header con token.

- Body:

{ ”body”: ”This is a comment” }

○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

6.5 Reply

- **POST /api/replies/id**

  - Objetivo: Responder a un comentario realizado.

  - Parámetros: Sin parámetros. - Aplica Authorization Header con token.

  - Body:

  { ”body”: ”This is a reply” }

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Solo el usuario creador del artículo podrá responder a los comentarios realizados.

6.6 Logs

- **GET /api/logs?desde=fecha_desde&hasta=fecha_hasta**

  - Objetivo: Listar el log de entradas al sistema y de búsquedas realizadas de todos los usuarios entre dos fechas.

  - Parámetros: FechaDesde y FechaHasta. - Aplica Authorization Header con token.

  - Body: Sin body.

  ○ Responses: - 200 Success - 401 Unauthorized - 403 Forbidden

  ○ Consideraciones:

  - Solo el usuario administrador podrá realizar esta petición.