Universidad ORT Uruguay
Facultad de Ingeniería

Obligatorio 1 de Diseño de Aplicaciones 2

Agustín Juárez - 236487, Agustín Campón - 233006

Tutor: Nicolas Fierro

Repositorio con la solución

2023

Introducción

Credenciales para el administrador del sistema:

Email: admin@hotmail.com

Usuario: admin

Contraseña: cleancode2023

Descripción del Diseño

Nuestro sistema implementa una API REST, la cual ofrece todas las funcionalidades requeridas para implementar una aplicación de Blogs.

Criterios de nuestra API REST

Para asegurar que nuestra API cumple con los criterios REST, tuvimos en cuenta los siguientes criterios a la hora de implementarla:

1) Identificamos los recursos: Definimos los recursos de la API (por ejemplo, usuarios, roles, artículos, comentarios, etc.) y utilizamos URLs consistentes para representarlos.
2) Utilizamos métodos HTTP apropiados: Implementamos los métodos HTTP correspondientes para describir las acciones que se pueden realizar sobre los recursos (GET, POST, PUT, DELETE, etc.).
3) Utilizamos códigos de estado HTTP adecuados: Retornamos los códigos de estado HTTP correctos según el resultado de la operación (200 OK, 201 Created, 400 Bad Request, 404 Not Found, etc.).
4) Implementar autenticación y autorización: De esta manera garantizamos que solo los clientes autorizados puedan acceder a los recursos de la API y realizar acciones específicas.
5) Documentar la API: Proveemos una documentación clara y completa de la API para un tercero sepa cómo utilizarla correctamente.

Siguiendo estos criterios, se puede asegurar que la API cumple con los principios fundamentales de REST.

Descripción del mecanismo de autenticación de las peticiones

La lógica detrás de la autenticación es poder mantener un estado entre las solicitudes HTTP, ya que el protocolo HTTP no tiene estado por naturaleza.

Una sesión nos permite identificar y rastrear a un usuario a lo largo de múltiples solicitudes, y así poder proporcionar una experiencia personalizada en función de los permisos y preferencias de cada usuario.

A) Procedimiento de autenticación :

1) User Ingresa sus credenciales:
    - El usuario hace login
2) Servidor Valida credenciales:
    - Consultamos la BD y validamos si existe un usuario que matchee las credenciales ingresadas.
3) Creación de la sesión:
    - Si las credenciales son válidas, creamos una nueva sesión para el usuario. (Esto implica generar un token de autenticación y asociarlo con el usuario ingresado.).
4) Almacenamiento de la sesión:
    - Almacenamos la sesión en la BD, esto permite al servidor rastrear y validar la sesión en futuras peticiones.

5) Envío del token de autenticación:
    - El servidor responde y envía el token generado al cliente. El cliente será responsable de almacenar este token y enviarlo en futuras solicitudes.

Una vez completado el proceso de autenticación, el usuario puede acceder a las vistas protegidas de la aplicación y el servidor es capaz de mantener estado entre solicitudes.

B) Verificación del token:

1) Recepción de las solicitudes:
    - Cuando el servidor recibe una petición que requiera autenticación (e.g: acceder a un área protegida de la aplicación) esta petición debe incluir el token de autenticación. Este es enviado dentro del header de la petición, en la propiedad Authorization.
2) El Servidor extrae el valor del token de la petición.
3) El servidor busca una sesión que contenga el token extraído.(Si encuentra una sesión eso indica que el token generado es válido).
4) Si la sesión es válida, el servidor puede extraer la información del usuario asociada a dicha sesión y asignarla al contexto de la petición actual. Esto permite que el servidor procese la petición en función de la identidad y permisos del usuario autenticado.
5) Una vez verificado el token y asociado el usuario, el servidor continúa procesando la solicitud, permitiendo o denegando el acceso al recurso, dependiendo del rol que tenga el usuario.

C) Cierre de sesión

1) Cuando el usuario decide cerrar la sesión (desloguearse), el servidor elimina la sesión almacenada en la bd y con ella el token de autenticación asociado.

   Esto garantiza que el token no podrá ser utilizado en futuras solicitudes.

Especificación de la API

Importante: Todos los endpoints de nuestra API a excepción del endpoint de login requieren autenticación. Es decir esto implica que en todas las request subsecuentes, en el header de cada solicitud debe ir el autorization token.

e.g : En el header de cada petición : Authorization : {{AUTH_TOKEN}}

Session Endpoints

Descripción General : La API "api/sessions" permite a los usuarios iniciar y cerrar sesión en el sistema. Se puede utilizar para autenticarse y obtener un token de acceso, así como para cerrar sesión y revocar el token.

a. Iniciar sesión

Método HTTP: POST

Ruta del endpoint: /api/sessions

Respuestas posibles:

- 200 OK: Devuelve un objeto que contiene el token de acceso.
- 400 Bad Request: La solicitud contiene datos incorrectos o incompletos, o las credenciales no son válidas.

Body Solicitud:

```
{
  "email": "admin@email.com",
  "password": "cleancode2023"
}
```

Respuesta:

```
{
  "token": "35a7efe8-99be-455d-950a-15a5c6b6d48f"
}
```

a. Cerrar sesión

Método HTTP: DELETE

Ruta del endpoint: /api/sessions

Respuestas posibles:

- 204 No Content: La sesión se cerró correctamente y el token de acceso fue revocado.
- 400 Bad Request: La solicitud contiene datos incorrectos o incompletos, o el token no es válido.

Header Solicitud:

```
{
  "Authorization": "35a7efe8-99be-455d-950a-15a5c6b6d48f"
}
```

Respuesta:

```
204 No Content
```

User Endpoints

Descripción General : La API "api/users" permite a los usuarios administrar usuarios y sus roles en el sistema. Se puede utilizar para crear, leer, actualizar y eliminar usuarios, así como para asignar roles a los usuarios.

a. Obtener todos los usuarios

Método HTTP: GET

Endpoint: /api/users

Respuestas posibles:

- 200 OK: Devuelve una lista de usuarios que coinciden con los criterios de búsqueda.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.

Respuesta:

```
[
  {
    "id": 1,
    "firstName": "admin",
    "lastName": "admin",
    "username": "admin",
    "email": "email@email.com",
    "password": "cleancode2023",
    "roles": [ 1 ]
  }
]
```

b. Obtener un usuario específico

Método HTTP: GET

Endpoint: /api/users/{id}

Endpoint example: /api/users/1

Respuestas posibles:

- 200 OK: Devuelve una lista de usuarios que coinciden con los criterios de búsqueda.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontró un usuario con el ID especificado.

Respuesta:


```
{
  "id": 1,
  "firstName": "admin",
  "lastName": "admin",
  "username": "admin",
  "email": "email@email.com",
  "password": "cleancode2023",
  "roles": [ 1 ]
}
```


c. Crear un nuevo usuario

Método HTTP: POST

Ruta del endpoint: /api/users

Respuestas posibles:

- 201 Created: El usuario ha sido creado con éxito
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 400 Bad Request: Los datos proporcionados son inválidos.
- 409 Conflict: Ya existe un usuario con el mismo nombre de usuario o correo electrónico.

Body Solicitud:


```
{
  "firstName": "blogger",
  "lastName": "blogger",
  "username": "blogger",
  "email": "blogger@email.com",
  "password": "cleancode2023",
  "roles": [ 2 ]
}
```


Respuesta:


```
{
  "id": 2,
  "firstName": "blogger",
  "lastName": "blogger",
  "username": "blogger",
  "email": "blogger@email.com",
  "password": "cleancode2023",
  "roles": [ 2 ]
}
```

d. Actualizar un usuario específico

Método HTTP: PUT

Endpoint: /api/users

Endpoint parameter example: /api/users/2

Body de la solicitud: Un objeto que contiene los campos necesarios para actualizar un usuario

Respuestas posibles:

- 200 OK: El usuario ha sido actualizado con éxito.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 400 Bad Request: Los datos proporcionados son inválidos.
- 409 Conflict: Ya existe un usuario con el mismo nombre de usuario o correo electrónico.

Body Solicitud:

```
{
  "id": 2,
  "firstName": "firstNameUpdated",
  "lastName": "blogger",
  "username": "blogger",
  "email": "blogger@email.com",
  "password": "cleancode2023",
  "roles": [ 2 ]
}
```

Respuesta:

```
{
  "id": 2,
  "firstName": "firstNameUpdated",
  "lastName": "blogger",
  "username": "blogger",
  "email": "blogger@email.com",
  "password": "cleancode2023",
  "roles": [ 2 ]
}
```

d. Eliminar un usuario específico

Método HTTP: DELETE

Endpoint: /api/users/{id}

Endpoint parameter example : /api/users/2

Respuestas posibles:

- 200 OK: El usuario ha sido eliminado con éxito.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontró un usuario con el ID especificado.

Respuesta:

```
200 OK
```

Role Endpoints

Descripción General : La API "api/roles" permite a los usuarios obtener los roles en el sistema. Se puede utilizar para obtener todos los roles disponibles.

a. Obtener todos los roles

Método HTTP: GET

Ruta del endpoint: /api/roles

Respuestas posibles:

- 200 OK: Devuelve una lista de usuarios que coinciden con los criterios de búsqueda.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontraron roles en el sistema.

Respuesta:

```
{
   "roles": [1,2]
}
```

Article Endpoints

Descripción General : La API "api/articles" permite a los usuarios administrar artículos en el sistema. Se puede utilizar para crear, leer, actualizar y eliminar artículos.

a. Obtener todos los artículos

Método HTTP: GET

Ruta del endpoint: /api/articles

Respuestas posibles:

- 200 OK : Devuelve una lista de todos los artículos disponibles en el sistema.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontraron artículos en el sistema.

Respuesta:


```
[
  {
    "id": 1,
    "authorId":  1,
    "title": "Article 1",
    "private":  false,,
    "comments":  null,
    "content": "Public",
    "image": "https://file-saver.com/article.jpg",
     "template":  1,
    "createdAt":  "2023-04-30T17:34:30.8897458-03:00",
    "updatedAt": null,
    "deletedAt":  null
  }
]
```


b. Obtener un artículo específico

Método HTTP: GET

Endpoint: /api/articles/{articleId}

Endpoint parameter example : /api/articles/1

Respuestas posibles:

- 200 OK : Devuelve el artículo solicitado.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontró un artículo con el ID especificado.

Respuesta:


```
  {
    "id": 1,
    "authorId":  1,
    "title": "Article 1",
    "private":  false,
    "comments":  null,
    "content": "Public",
    "image": "https://file-saver.com/article.jpg",
     "template":  1,
    "createdAt":  "2023-04-30T17:34:30.8897458-03:00",
    "updatedAt": null,
    "deletedAt":  null
```

```
}
```

c. Crear un nuevo artículo

Método HTTP: POST

Ruta del endpoint: /api/articles

Respuestas posibles:

- 201 Created: El artículo ha sido creado con éxito.
- 400 Bad Request: La solicitud contiene datos incorrectos o incompletos.
- 404 Not Found: No se encontró el autor especificado.

Body solicitud:

```
{
   "authorId":  1,
   "title": "Article 1",
   "private":  false,
   "content": "Public",
   "image": "https://file-saver.com/article.jpg",
    "template":   1,
}
```

Respuesta:

```
{
   "id": 1,
   "authorId":  1,
   "title": "Article 1",
   "private":  false,
   "comments":  null,
   "content": "Public",
   "image": "https://file-saver.com/article.jpg",
    "template":   1,
   "createdAt":  "2023-04-30T17:34:30.8897458-03:00",
   "updatedAt": null,
   "deletedAt":  null
}
```

c. Actualizar un nuevo artículo

Método HTTP: PUT

Endpoint: /api/articles/{articleId}

Endpoint parameter example : /api/articles/1

Respuestas posibles:

- 200 OK: El artículo ha sido actualizado con éxito.
- 400 Bad Request: La solicitud contiene datos incorrectos o incompletos.
- 401 Unauthorized: El usuario no tiene permiso para actualizar este artículo.
- 404 Not Found: No se encontró un artículo con el ID especificado.

Body Solicitud:

```
{
   "authorId": 1,
   "title": "Modified Article 1",
   "private": false,
   "comments": null,
   "content": "Contenido del artículo",
   "image": "https://file-saver.com/article.jpg",
    "template": 1
}
```

Respuesta:

```
{
   "id": 1,
   "authorId": 1,
   "title": "Modified Article 1",
   "private": false,,
   "comments": null,,
   "content": "Contenido del artículo",
   "image": "https://file-saver.com/article.jpg",
    "template": 1,
   "createdAt": "2023-04-30T17:34:30.8897458-03:00",
   "updatedAt": "2023-04-30T23:45:11.0269279-03:00",
   "deletedAt": null
}
```

d. Eliminar un Articulo especifico

Método HTTP: DELETE

Endpoint: /api/articles/{id}

Endpoint parameter example: /api/article/1

Respuestas posibles:

- 200 OK: El artículo ha sido eliminado con éxito.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontró un artículo con el ID especificado.

Respuesta:

```
200 OK
```

Comments Endpoints

Descripción General : La API "api/comments" permite a los usuarios administrar comentarios en los artículos del sistema. Se puede utilizar para crear, leer y responder a los comentarios.

a. Obtener todos los comentarios

Método HTTP: GET

Endpoint: /api/comments

Endpoint parameter example: FIXME

Respuestas posibles:

- 200 OK : Devuelve una lista de todos los artículos disponibles en el sistema.
- 401 Unauthorized: El usuario no tiene permiso para acceder a este recurso.
- 404 Not Found: No se encontraron artículos en el sistema.

Respuesta:


```
[
  {
    "id": 1,
    "articleId": 1,
    "authorId": 1,
    "replyId" : null,
    "content": "This is a comment for article 1",
    "createdAt": "2023-04-30T17:34:30.8897458-03:00",
    "updatedAt": null,
    "deletedAt": null
  }
]
```


b. Crear un nuevo comentario

Método HTTP: POST

Endpoint: /api/comments

Respuestas posibles:

- 201 Created: El comentario ha sido creado con éxito.
- 400 Bad Request: La solicitud contiene datos incorrectos o incompletos.
- 401 Unauthorized: El usuario no tiene permiso para comentar este artículo.
- 404 Not Found: No se encontró el artículo o autor especificado.

Body solicitud:

```
{
  "articleId": 1,
   "authorId": 1,
   "content": "This is a comment for article 1",
}
```

Respuesta:

```
{
   "id": 1,
  "articleId": 1,
   "authorId": 1,
   "replyId" : null,
   "content": "This is a comment for article 1",,
   "createdAt": "2023-04-30T17:34:30.8897458-03:00",
   "updatedAt": null,
   "deletedAt": null
}
```

c. Obtener un comentario específico

Método HTTP: GET

Endpoint: /api/comments/{commentId}

Endpoint parameter example: /api/comments/1

Respuestas posibles:

- 200 OK: Devuelve el comentario solicitado.
- 404 Not Found: No se encontró un comentario con el ID especificado.

Respuesta:

```
{
   "id": 1,
  "articleId": 1,
   "authorId": 1,
   "replyId" : 1,
   "content": "This is a comment for article 1",,
   "createdAt": "2023-04-30T17:34:30.8897458-03:00",
   "updatedAt": null,
   "deletedAt": null
}
```

d. Crear una respuesta a un comentario

Método HTTP:POST

Endpoint :/api/comments/{commentId}/reply

Endpoint parameter example : /api/comments/1/reply

Respuestas posibles:

- 201 Created: La respuesta al comentario ha sido creada con éxito.
- 400 Bad Request: La solicitud contiene datos incorrectos o incompletos, o se intenta responder a un comentario que ya tiene respuesta.
- 401 Unauthorized: El usuario no tiene permiso para responder a este comentario.
- 404 Not Found: No se encontró el comentario, artículo o autor especificado.

Body Solicitud:

```
{
   "authorId": 1,
   "content": "This is a reply for comment 1"
}
```

Respuesta:

```
{
   "id": 2,
  "articleId": 1,
   "authorId": 1,
   "replyId" : null,
   "content": "This is a reply for comment 1",
   "createdAt": "2023-04-30T17:34:30.8897458-03:00",
   "updatedAt": null,
   "deletedAt": null
}
```