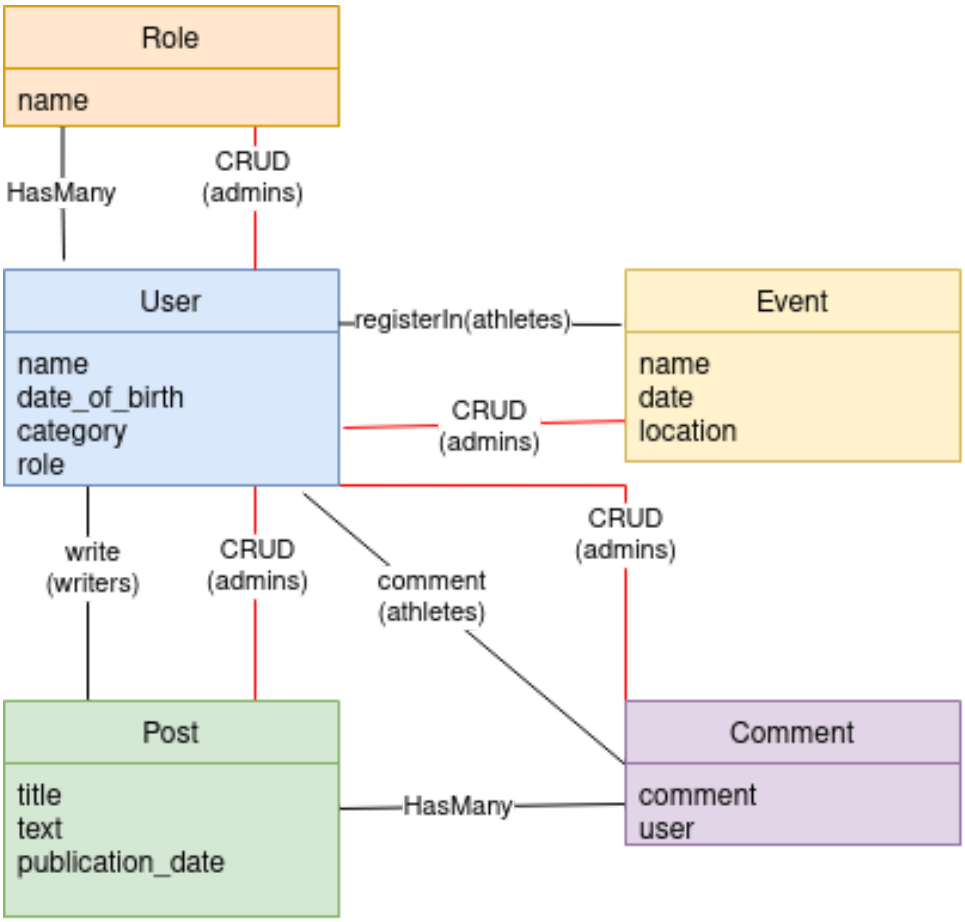
Práctica 1: Diseño e Implementación APIs REST

Fase 1: Diseño y especificación del API

En esta práctica desarrollaremos un API REST para la gestión de un equipo deportivo. El objetivo es facilitar a los gerentes del club algunas de las tareas repetitivas que se realizan a lo largo de la temporada. Asimismo, se aproevechará el sitio web a modo de 'blog' para entradas que puedan resultar de interés para el público o promoción para el club.

Los recursos con los que empezará a trabajará el API REST son, por ahora:



(ACTUALIZACIÓN: No se ha podido implementar la parte relativa al blog)

Entre paréntesis se reflejan los tipos de usuarios que pueden realizar cada acción; es decir, los *roles* que cuentan con los permisos para cada acción. El control de acceso basado en roles es una parte importante de la funcionalidad del API REST. Nos basamos en él para ampliar las posibilidades de nuestra aplicación.

Se trata de un diagrama poco detallado, aún hay muchos casos de uso que no están reflejados. Por ejemplo, las posibilidades que tienen los usuarios sin autenticar no están reflejados.

Una breve explicación de los casos de uso expuestos en el diagrama y algunos más son:

- Los usuarios no autenticados podrán consultar tanto las entradas del blog como los comentarios de los mismos. Está por decidir aún si consideramos interesante que puedan acceder a la información del calendario de competiciones (eventos) y/o integrantes del club (usuarios atletas).

  > De ahora en adelante, para simplificar la lectura nos referiremos a los distintos tipos de usuarios por el nombre de los roles identificados (**administradores, atletas, entrenadores y escritores,**).

- Los atletas se podrán apuntar a las competiciones que puedan. Esta funcionalidad es la que dota de mayor utilidad a la aplicación. El club es el encargado de inscribir a los deportistas a las competiciones, pero recopilar los nombres de las personas que quieren apuntarse resulta complicado en un club descentralizado de cientos de personas (atletas que avisan por correo, otros por WhatsApp, Instagram, etc; otros que avisan en persona al entrenador -queda al amparo de la memoria del entrenador- y demás situaciones).
- Los atletas podrán también dejar comentarios en las entradas del blog.
- Los escritores podrán, como su propio nombre indica, escribir las entradas del blog.
- Un post puede tener comentarios.
- Los administradores pueden realizar las operaciones básicas CRUD de cualquier recurso. Pero, por ejemplo, puede ser interesante conceder permisos de edición o creación de eventos a los entrenadores (para reducir la carga de trabajo a los administradores).

> Cabe recalcar que los roles no son excluyentes. Un usuario puede contar con varios roles.

Especificación de llamadas al API (actualización)

User

GET /users *(Lista todos los usuarios)*

Respuesta:

Cabecera	
Content-Type	application/json

Código	**Descripción**
200	Operación exitosa, devuelve los recursos
500	Error de servidor

GET /users/{id} (*Lista el usuario según el parámetro {id} enviado en la url*)

Petición:

Parámetro	**Descripción**
id	Identificador del usuario

Respuesta:

Cabecera	
Content-Type	application/json

Código	**Descripción**
200	Devuelve el recurso solicitado
401	Parámetro inválido
404	No existe

POST /users/ (*Creamos un nuevo usuario*)

Petición:

Cabecera	
Content-Type	application/json
Cuerpo	

Información del recurso a crear	

Respuesta:

Cabecera	
Location	URL del recurso creado
Cuerpo	
Información del recurso creado	

Código	**Descripción**
201	Recurso creado correctamente.
400	Petición incorrecta
401	Sin autenticar o credenciales incorrectas
500	Error de servidor

PUT /users/{id} (*Actualiza un usuario*)

Petición:

Parámetro	**Descripción**
id	Identificador del usuario a actualizar

Cabecera	
Content-Type	application/json
Cuerpo	
Información del recurso a actualizar	

Respuesta:

Cabecera	
Content-Type	application/json

Código	**Descripción**
204	Recurso modificado correctamente
404	No existe
500	Error de servidor

DELETE /users/{id} (*Elimina un usuario*)

Petición:

Parámetro	Descripción
id	Identificación del usuario a actualizar

Cabecera	
Content-Type	application/json

Respuesta:

Cabecera	
Content-Type	application/json

Código	Descripción
204	Recurso eliminado correctamente
404	No existe
500	Error de servidor

Event

GET /events *(Lista todos los eventos)*

Respuesta:

Cabecera	
Content-Type	application/json

Código	Descripción
200	Operación exitosa, devuelve los recursos
500	Error de servidor

POST /events/ (*Creamos un nuevo evento*)

Petición:

Cabecera	
Content-Type	application/json
Cuerpo	
Información del recurso a crear	

Respuesta:

Cabecera	

Location	URL del recurso creado
Cuerpo	
Información del recurso creado	

Código	**Descripción**
201	Recurso creado correctamente.
400	Petición incorrecta
401	Sin autenticar o credenciales incorrectas
500	Error de servidor

DELETE /events/{event_id} (*Elimina un evento*)

Petición:

Parámetro	**Descripción**
event_id	Identificación del evento a eliminar

Cabecera	
Content-Type	application/json

Respuesta:

Cabecera	
Content-Type	application/json

Código	**Descripción**
204	Recurso eliminado correctamente
404	No existe
500	Error de servidor

Role

GET /roles *(Lista todos los roles)*

Respuesta:

Cabecera	
Content-Type	application/json

Código	**Descripción**

200	Operación exitosa, devuelve los recursos
500	Error de servidor

POST /role/ (*Creamos un nuevo rol*)

Petición:

Cabecera	
Content-Type	application/json
Cuerpo	
Información del recurso a crear	

Respuesta:

Cabecera	
Location	URL del recurso creado
Cuerpo	
Información del recurso creado	

Código	**Descripción**
201	Recurso creado correctamente.
400	Petición incorrecta
401	Sin autenticar o credenciales incorrectas
500	Error de servidor