Diseño de Software

Prototipo de Strava I

.....

- Xiang Chen Chen
- Adrián Feijoo Martínez
- Carlos García Cabezudo
- Emilio Gil del Río Pérez

.....

Índice

Diseño detallado del API del Servidor Central

A continuación se lista el diseño de APIs del servidor central, realizado a partir de la descripción y requisitos del proyecto, con ayuda parcial de IA generativa, si bien todo ha sido exhaustivamente revisado.

1. Operaciones de autenticación

1.1. Registration
Endpoint: `/register`
Method: **POST**
Descripción: Registra un nuevo usuario usando una cuenta de Google o Meta.
Parámetros de entrada (Cuerpo JSON):

- `email` (string, requerido)
- `password` (string, requerido)
- `name` (string, requerido)
- `date_of_birth` (string, formato YYYY-MM-DD, requerido)
- `weight` (number, kilogramos, opcional)
- `height` (number, centímetros, opcional)
- `max_heart_rate` (number, bpm, opcional)
- `resting_heart_rate` (number, bpm, opcional)
- `auth_provider` (string, requerido, valores: "google", "meta")

Parámetros de salida (JSON):

- `message` (string)
- `user_id` (string)

1.2. Login
Endpoint: `/login`
Method: **POST**
Descripción: Inicia sesión validando las credenciales del usuario con Google o Meta. Devuelve un token para solicitudes autenticadas.
Parámetros de entrada (Cuerpo JSON):

- `email` (string, requerido)
- `password` (string, requerido)
- `auth_provider` (string, requerido, valores: "google", "meta")

Parámetros de salida (JSON):

- `token` (string)

1.3. Logout
Endpoint: `/logout`
Method: **POST**
Descripción: Cierra la sesión invalidando el token del usuario.
Parámetros de entrada (Cuerpo JSON):

- `token` (string, requerido)

Parámetros de salida (JSON):

- `message` (string)

---

2. Sesiones entrenamiento

2.1 Crear sesión de entrenamiento
Endpoint: `/sessions`
Method: **POST**
Descripción: Crea manualmente una nueva sesión de entrenamiento.
Parámetros de entrada (Cuerpo JSON):

- `token` (string, requerido)
- `title` (string, requerido)
- `sport` (string, requerido, valores: "cycling", "running")
- `distance` (number, kilómetros, requerido)
- `start_date` (string, formato YYYY-MM-DD, requerido)
- `start_time` (string, formato HH:MM
  , requerido)
- `duration` (number, minutos, requerido)

Parámetros de salida (JSON):

- `message` (string)
- `session_id` (string)

2.2 Consultar sesiones de entrenamiento
Endpoint: `/sessions`
Method: **GET**
Descripción: Devuelve las sesiones de entrenamiento del usuario. Por defecto, devuelve las 5

últimas o filtra por un rango de fechas determinado.
Parámetros de entrada (Query Parameters):

- `token` (string, requerido)
- `start_date` (string, formato YYYY-MM-DD, opcional)
- `end_date` (string, formato YYYY-MM-DD, opcional)
- `limit` (number, opcional, el valor predeterminado es 5)

Parámetros de salida (JSON):

- `sessions` (array de objetos de sesión)

---

3. Retos

3.1 Crear reto
Endpoint: `/challenges`
Method: **POST**
Descripción: Crea un nuevo desafío al que solo los usuarios pueden unirse.
Parámetros de entrada (Cuerpo JSON):

- `token` (string, requerido)
- `name` (string, requerido)
- `start_date` (string, formato YYYY-MM-DD, requerido)
- `end_date` (string, formato YYYY-MM-DD, requerido)
- `objective_value` (number, requerido)
- `objective_type` (string, requerido, valores: "distance", "time")
- `sport` (string, requerido, valores: "cycling", "running")

Parámetros de salida (JSON):

- `message` (string)
- `challenge_id` (string)

3.2 Consultar retos activos
Endpoint: `/challenges`
Method: **GET**
Descripción: Recupera desafíos activos que aún no han terminado.
Parámetros de entrada (Query Parameters):

- `token` (string, requerido)

- `date` (string, formato YYYY-MM-DD, opcional, por defecto es la fecha actual)
- `sport` (string, opcional, valores: "cycling", "running")
- `limit` (number, opcional, el valor predeterminado es 5)

Parámetros de salida (JSON):

- `challenges` (array de objetos de desafío)

3.3 Aceptar reto
Endpoint: `/challenges/{challenge_id}/accept`
Method: **POST**
Descripción: Acepta y únete a un desafío.
Parámetros de entrada (Ruta URL & Cuerpo JSON):

- `challenge_id` (string, requerido en la ruta URL)
- `token` (string, requerido en el cuerpo JSON)

Parámetros de salida (JSON):

- `message` (string)

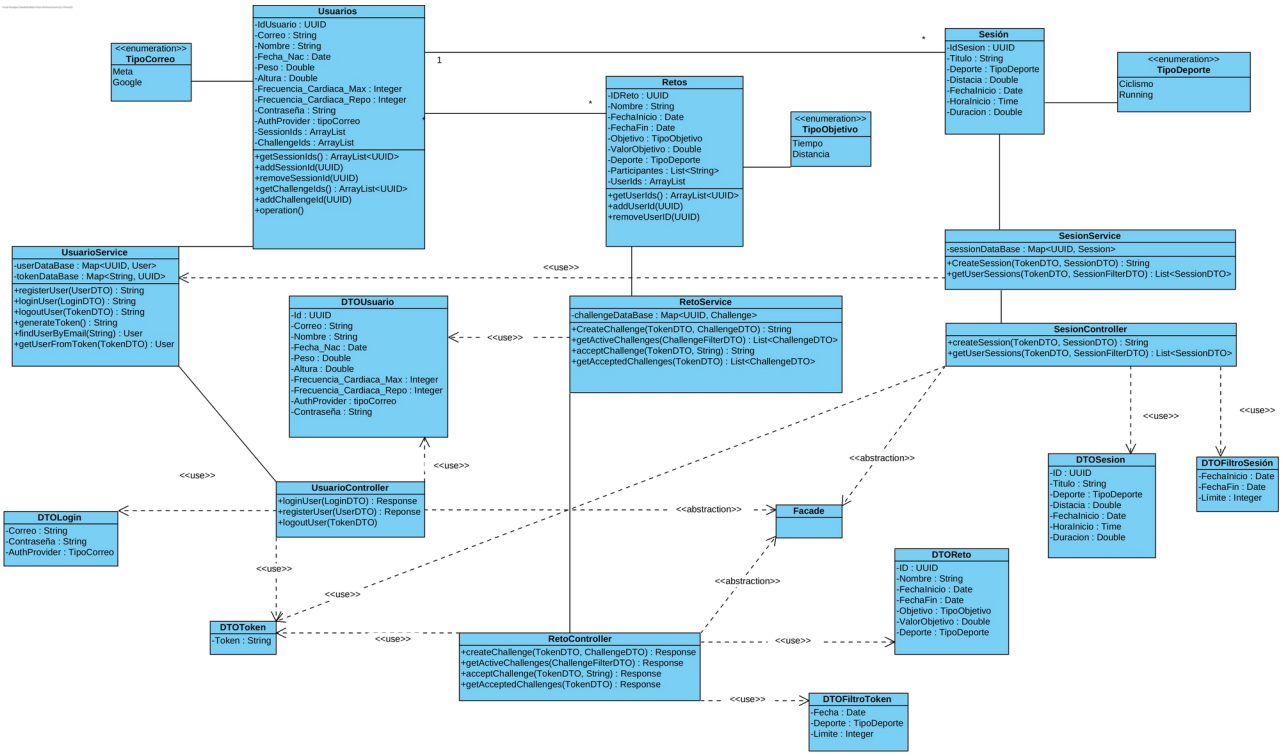
3.4 Consultar retos aceptados
Endpoint: `/challenges/accepted`
Method: **GET**
Descripción: Recupera los desafíos que el usuario ha aceptado.
Parámetros de entrada (Query Parameters):

- `token` (string, requerido)

Parámetros de salida (JSON):

- `challenges` (array de objetos de desafío)

Diagrama de clases y de secuencia



Nota: con el fin de no sobrecargar demasiado el diagrama, se omiten la mayor parte de los getters y los setters.

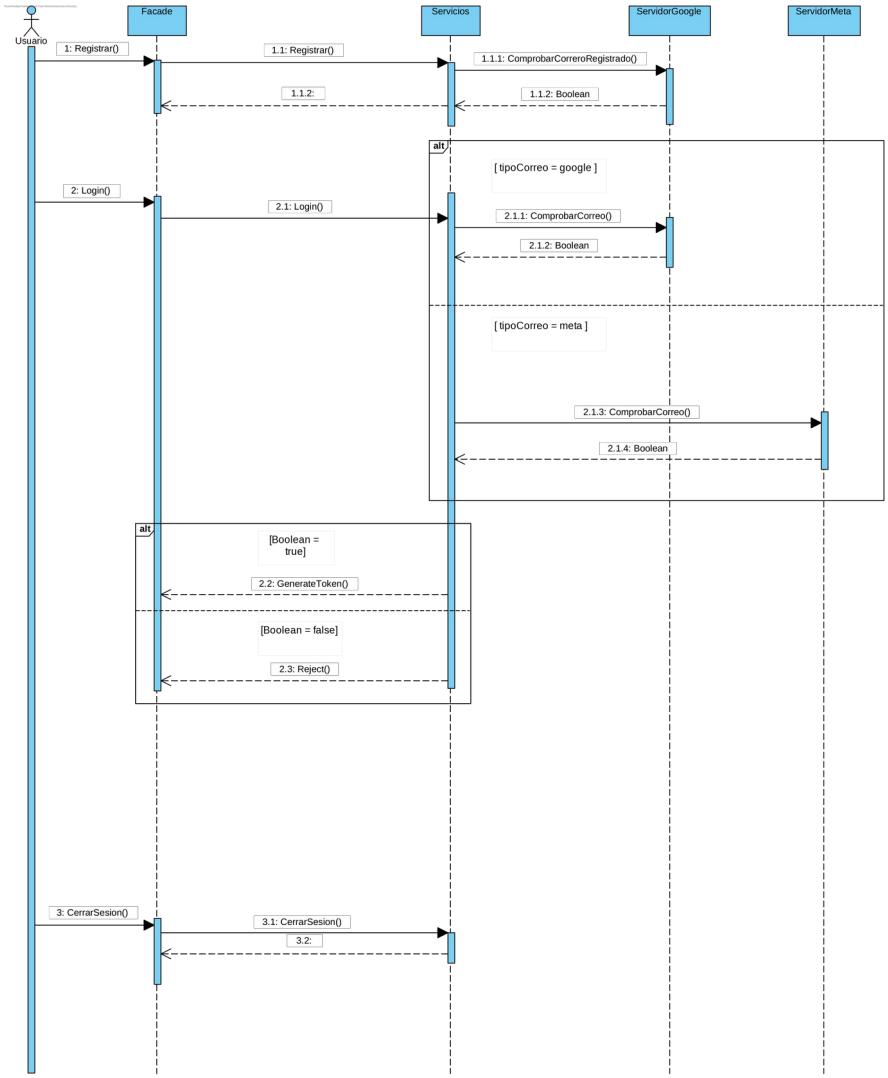
Patrones de diseño:

- Se implementa el patrón DTO para la transferencia de datos entre las capas de controlador, servicio y/o clases.

- Se implementa el patrón Façade en forma de controladores, que proporcionan las APIs al usuario.

- Se implementa el patrón AppService para gestionar la lógica interna del servidor.

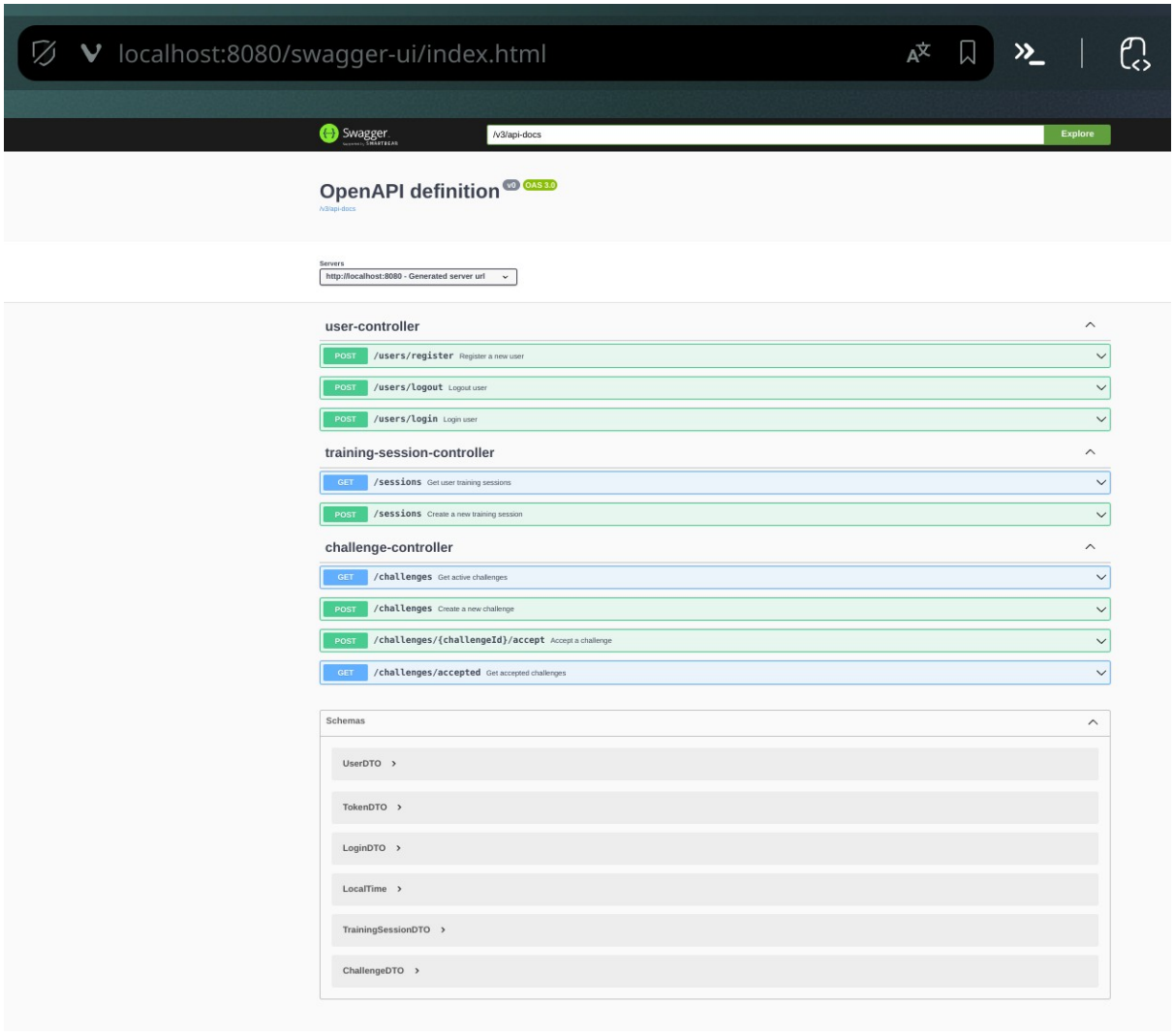
- Se implementa el patrón StateManagment en cuanto al uso de Tokens para la gestión del estado de usuario.



Implementación del Servidor Strava

El código del servidor de Strava ha sido implementado siguiendo los diseños previos de las APIs, diagramas de clases y secuencia como base. Mediante la integración de estas especificaciones con IA generativa, se logró desarrollar una parte importante de la estructura general del código. Posteriormente, se llevó a cabo una revisión exhaustiva que permitió refinar y detallar aún más la implementación, optimizando los métodos específicos en el proceso.

Además, se añadieron anotaciones de Swagger que permiten acceder a la interfaz interactiva de las APIS en http://localhost:8080/swagger-ui/index.html. A partir de la misma y siguiendo los pasos guiados por ChatGPT se generó la documentacion de las APIs (el documento adjunto llamado OpenAPI definition).

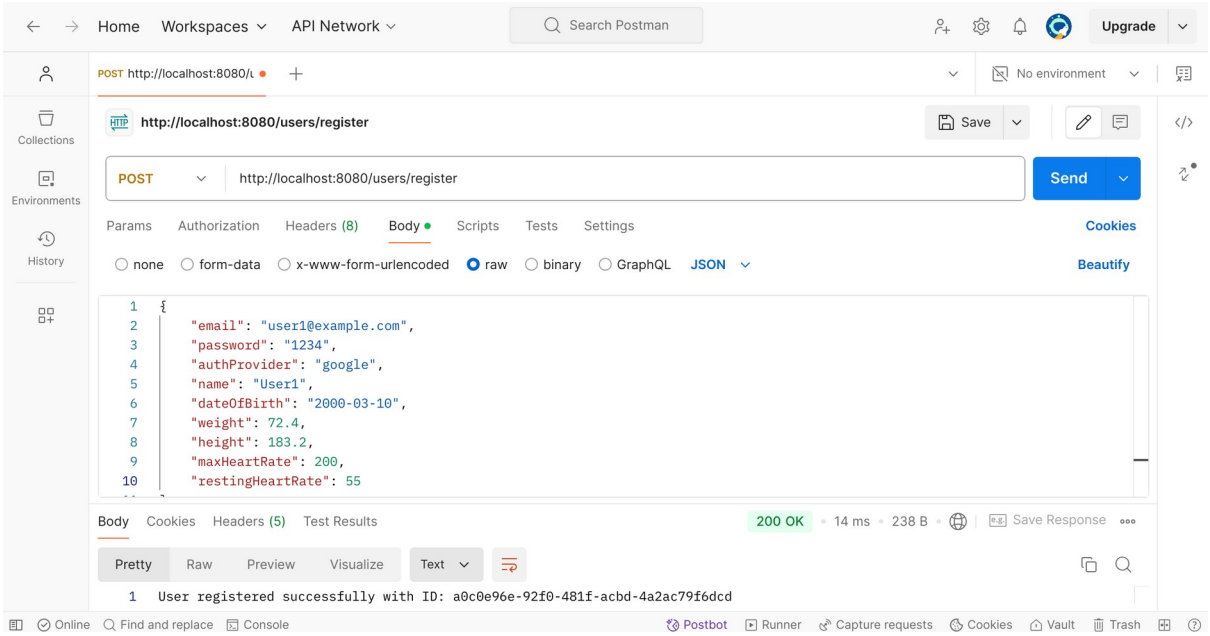


El código se encuentra junto a este documento, en la carpeta strava-proyecto-eclipse. Para ejecutarlo, se debe importar en Eclipse como proyecto de Gradle. Una vez iniciado el servidor con SpringBoot, será posible hacer uso de las APIs.

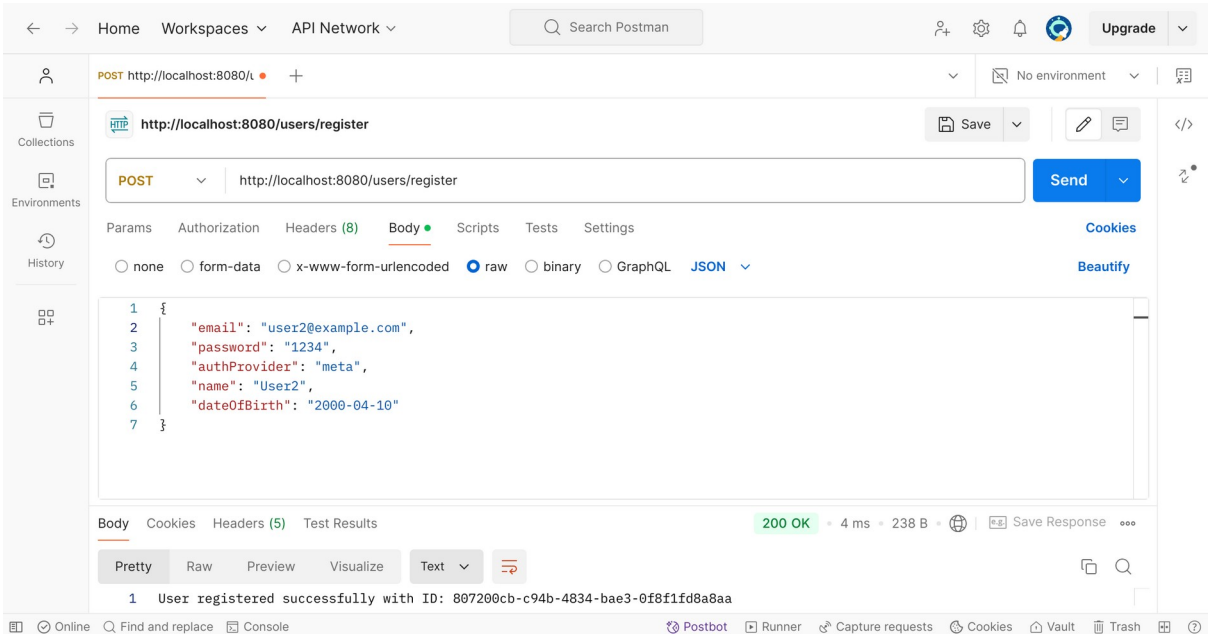
Validación de las APIs

A continuación se presenta una ilustración del funcionamiento básico de las APIs utilizando Postman. No obstante, para obtener un mayor nivel de detalle y comprender completamente su interacción, se recomienda realizar pruebas directamente con las APIs.

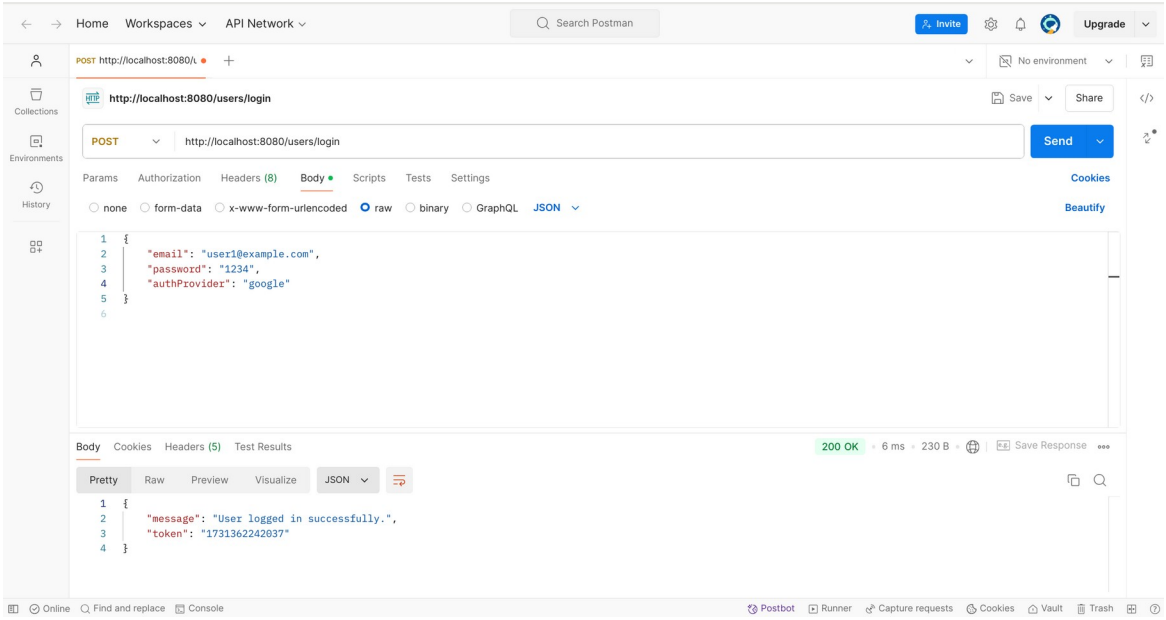
Registro de un usuario | NOTA: Con posterioridad a las capturas, solo se aceptan los valores “GOOGLE” y “META” como authProvider, tanto para el registro como para el login.



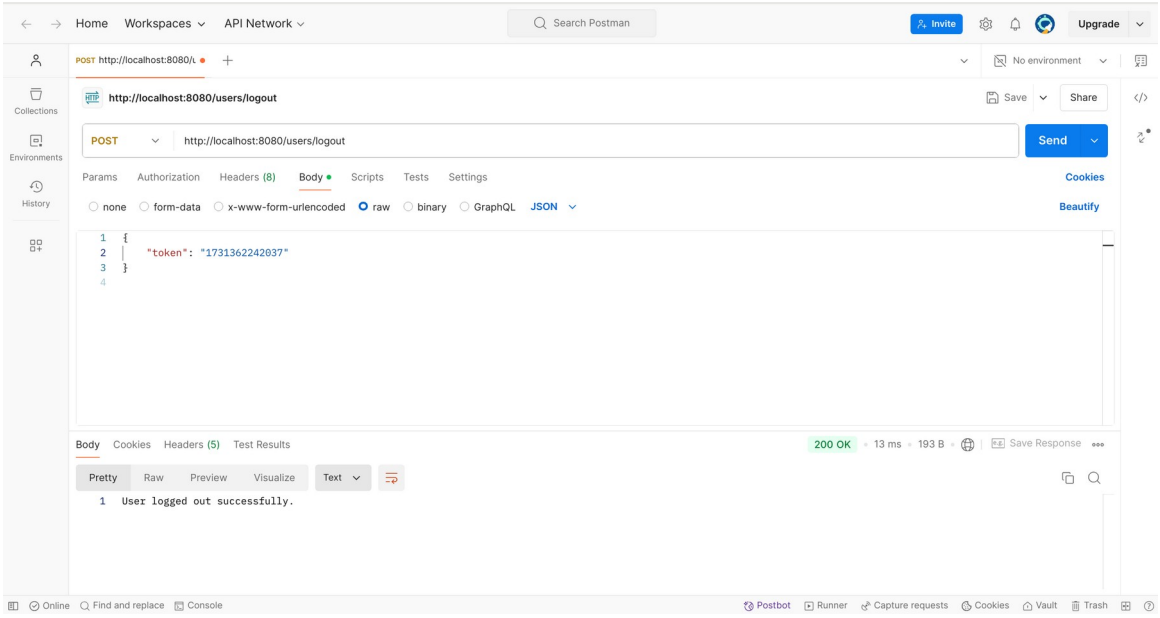
Se pueden omitir los parámetros opcionales.



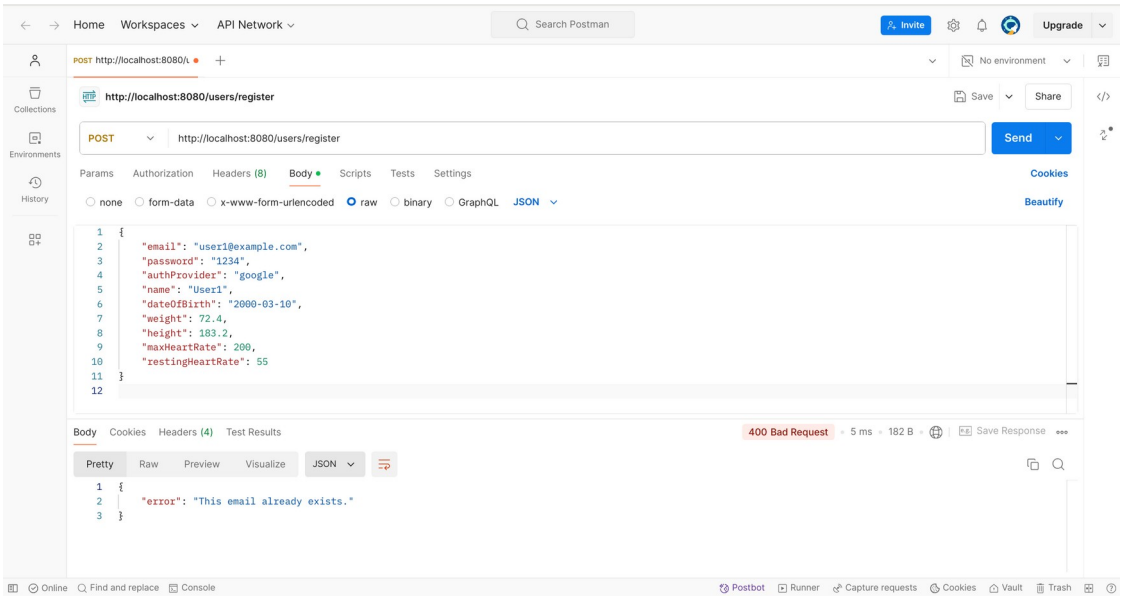
Login de un usuario (ya registrado). Se permiten múltiples logins por usuario, ya que podría ser útil, por ejemplo, para que un mismo usuario tenga diferentes sesiones en múltiples dispositivos, cada una con un token independiente.



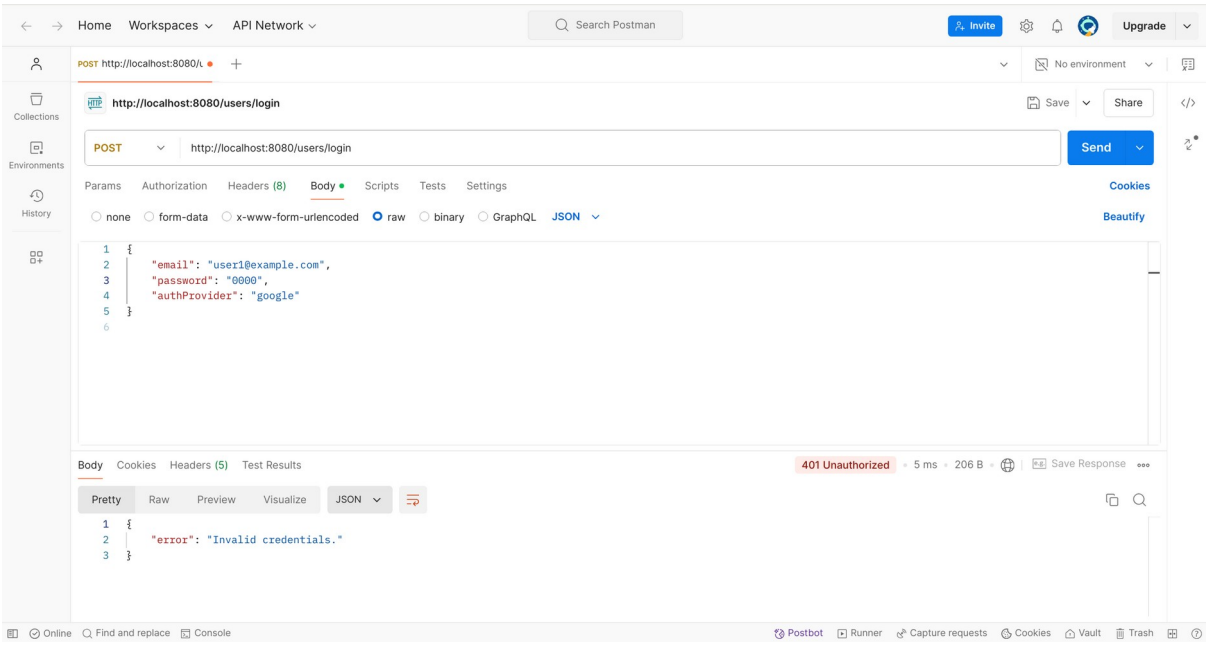
Logout



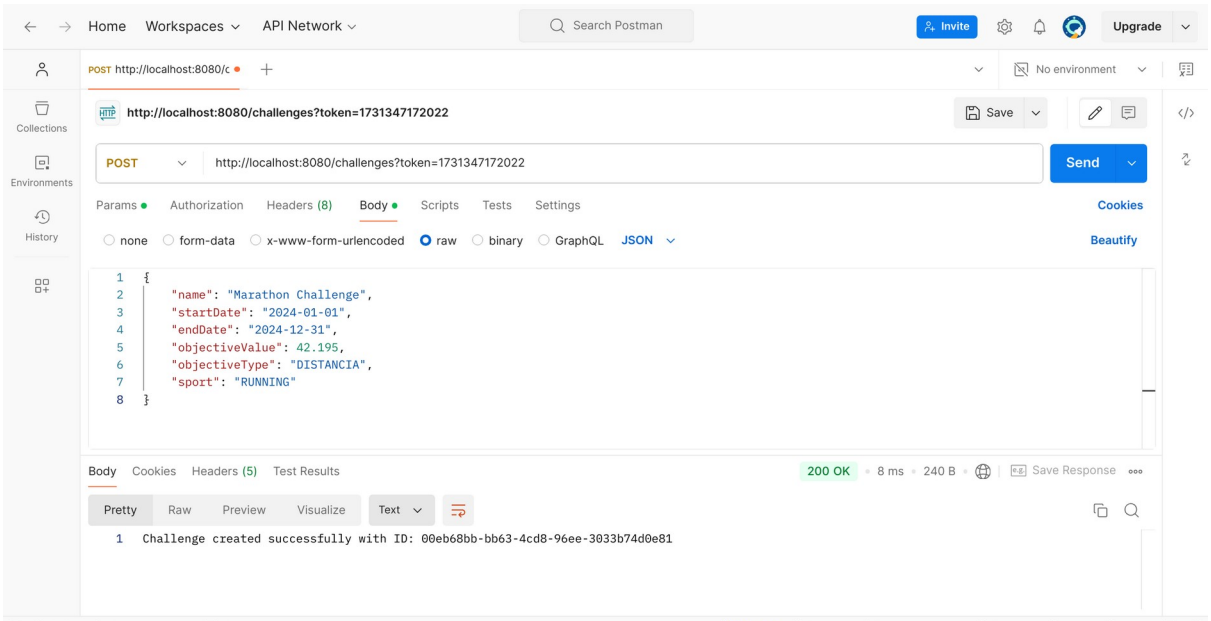
No se permite la creación de usuarios con el mismo email.

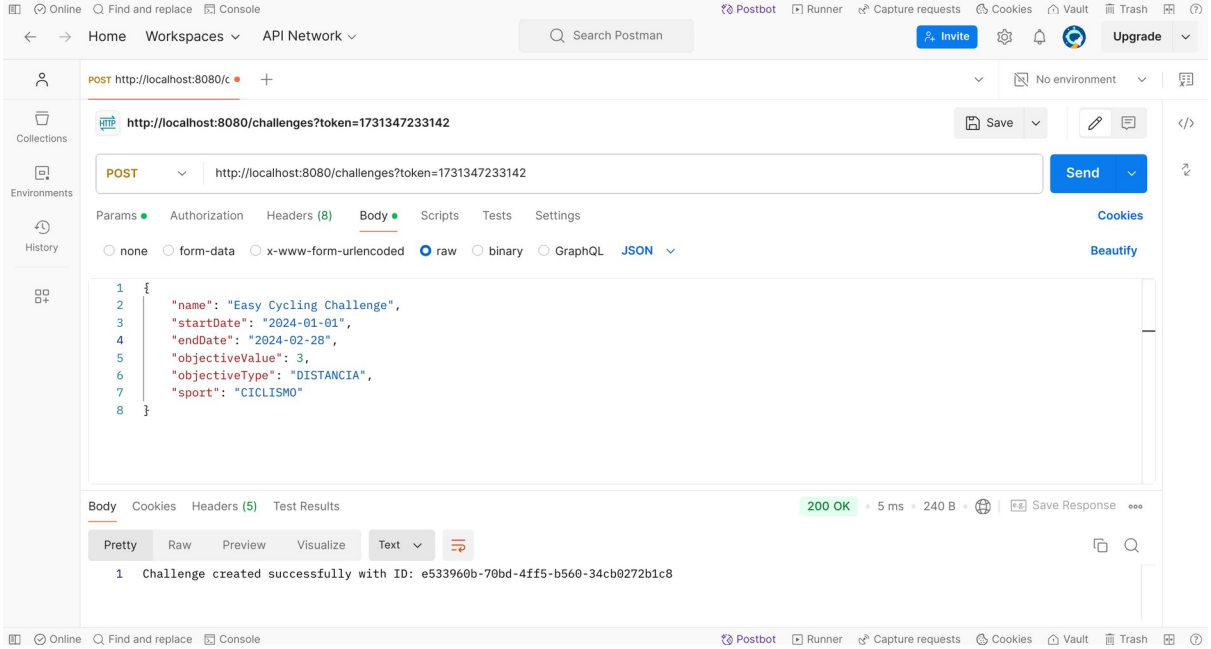


Las credenciales para el login deben ser las del registro.

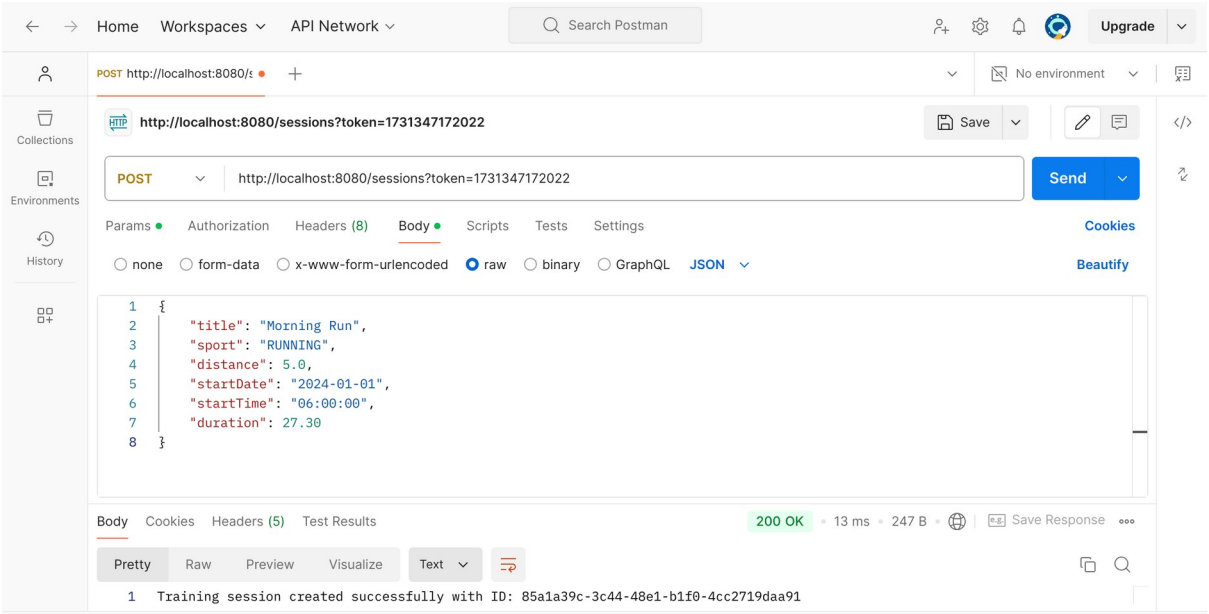


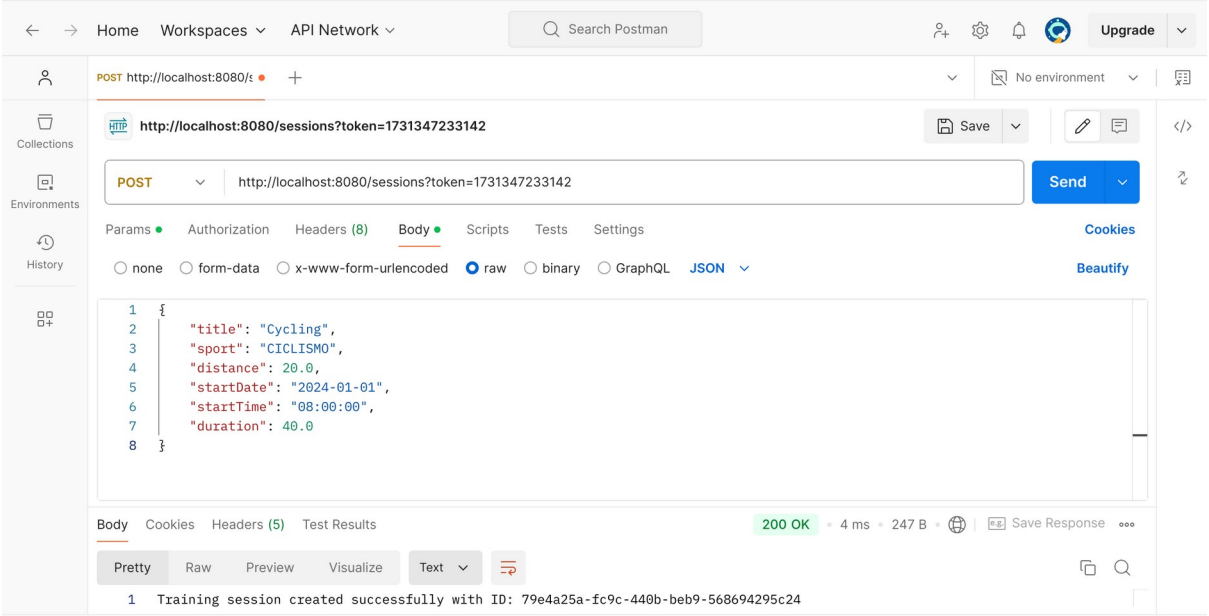
Creación de retos de distintos usuarios.

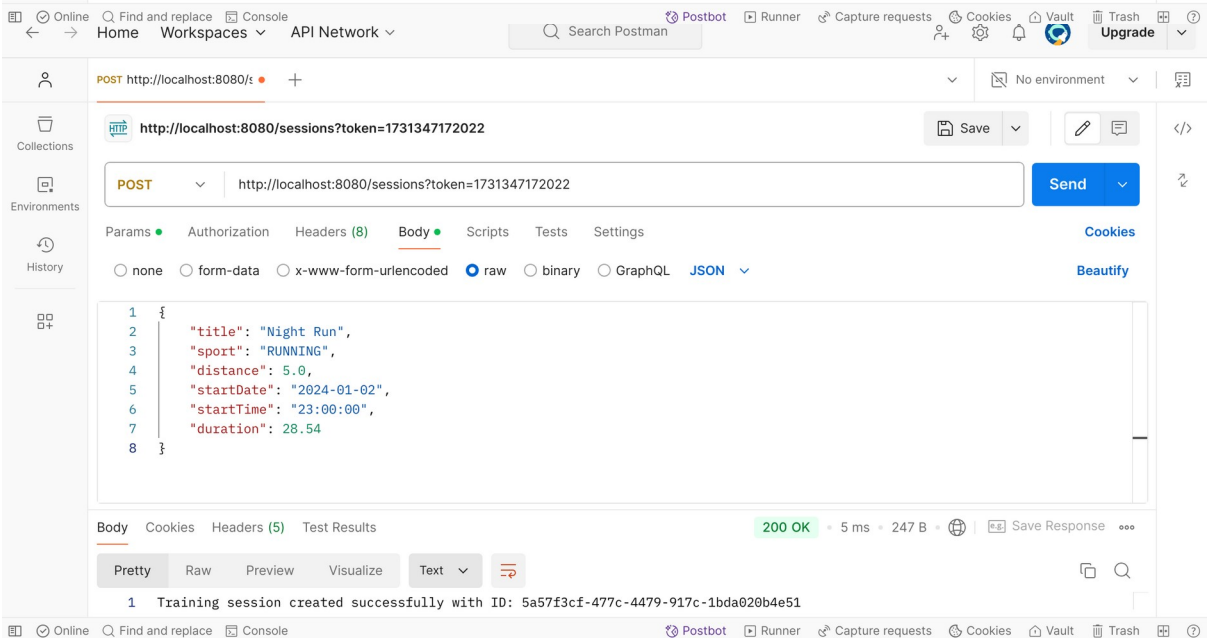




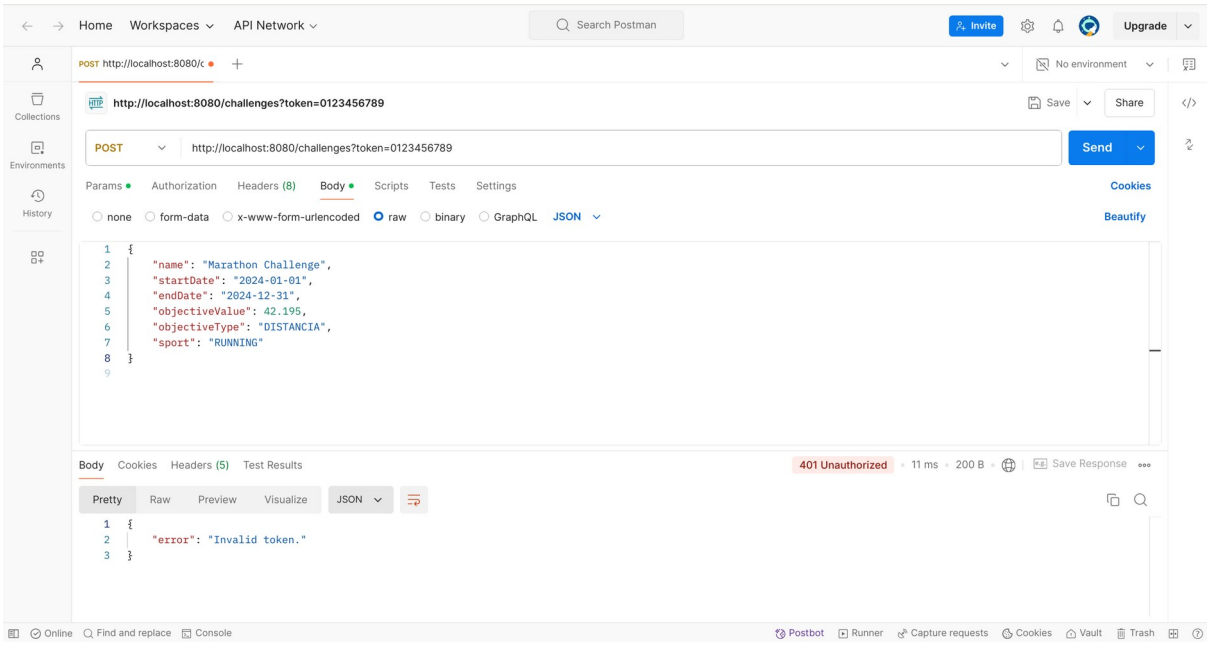
Creación de varias sesiones de entrenamiento.

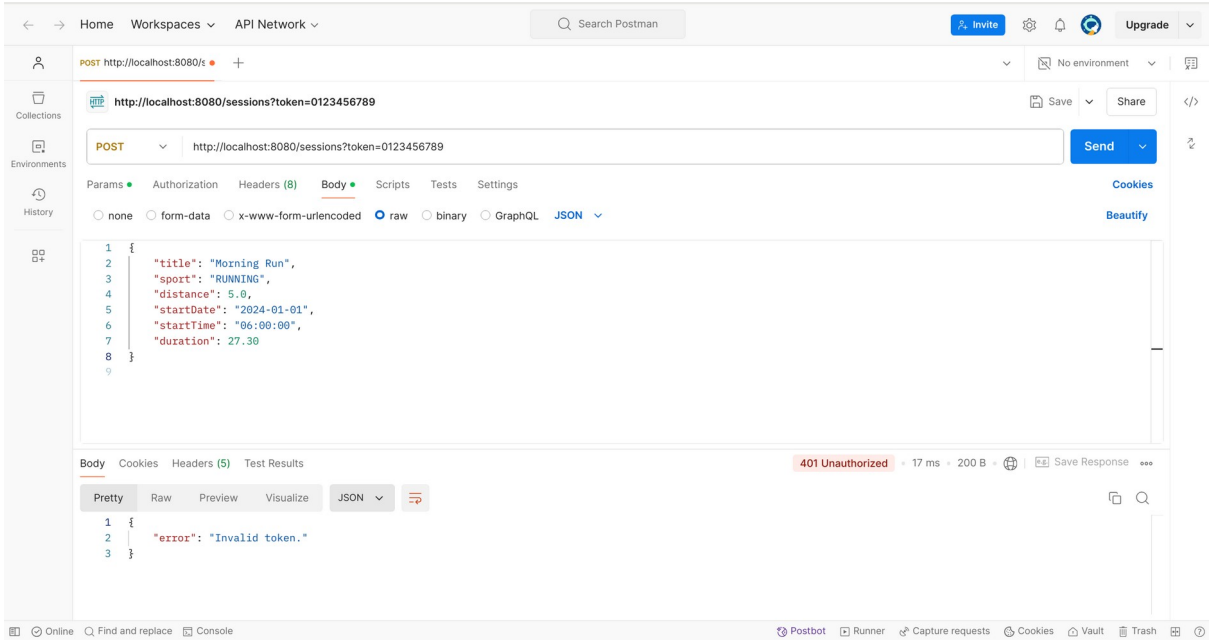




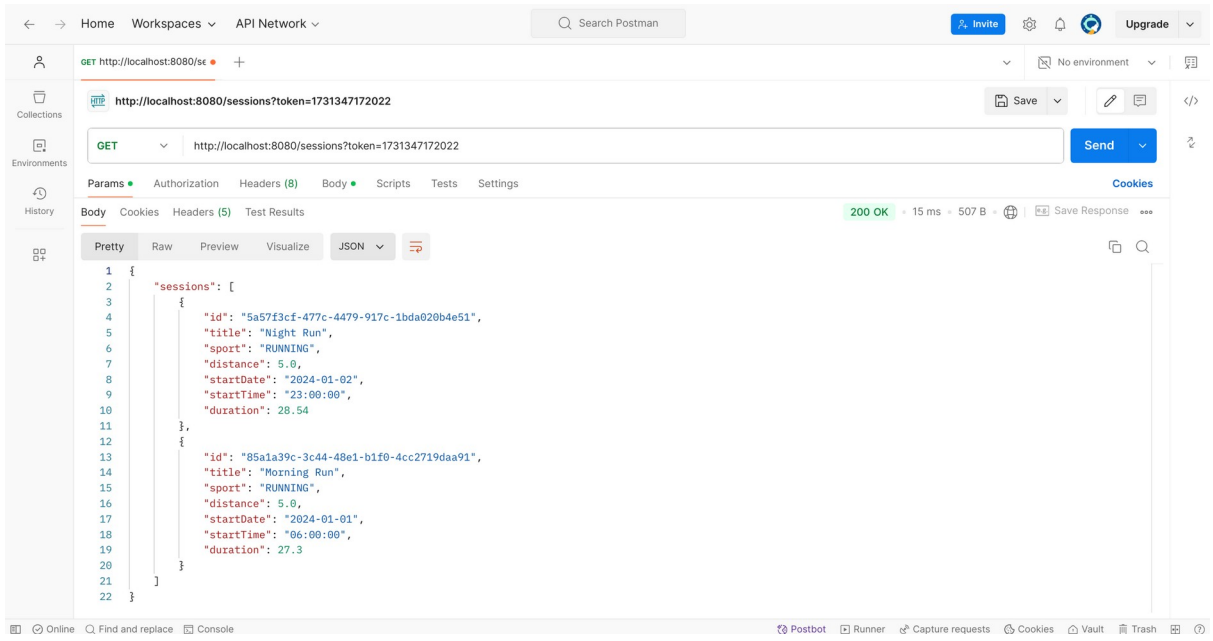


Las operaciones asociadas a un usuario requieren la validación del token.

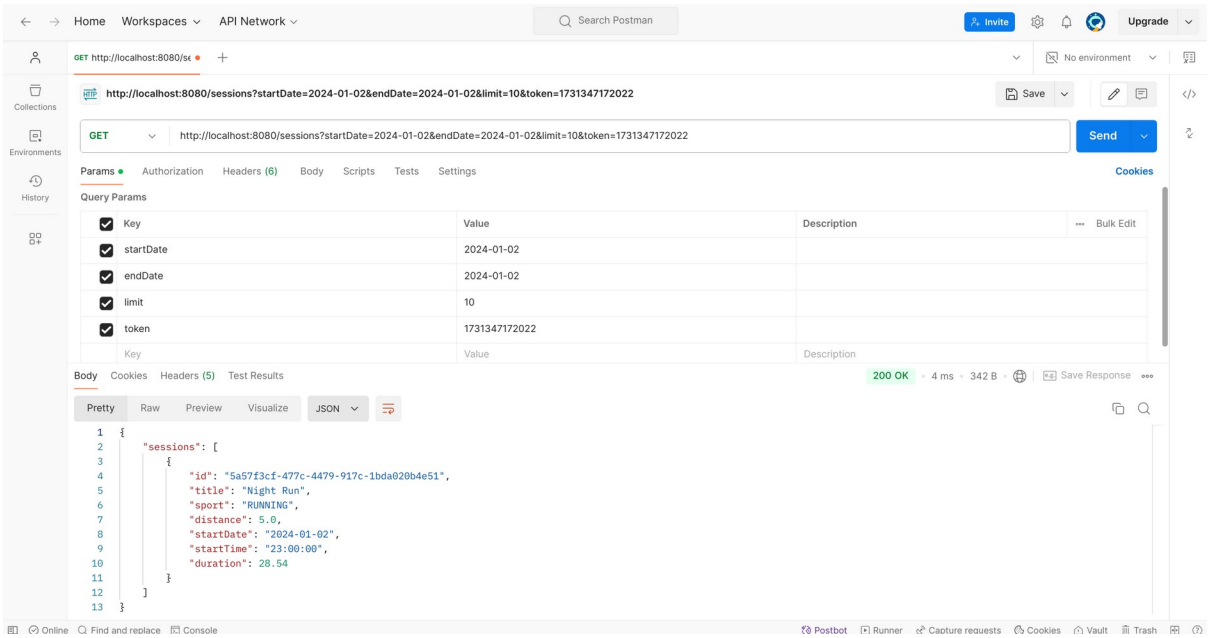




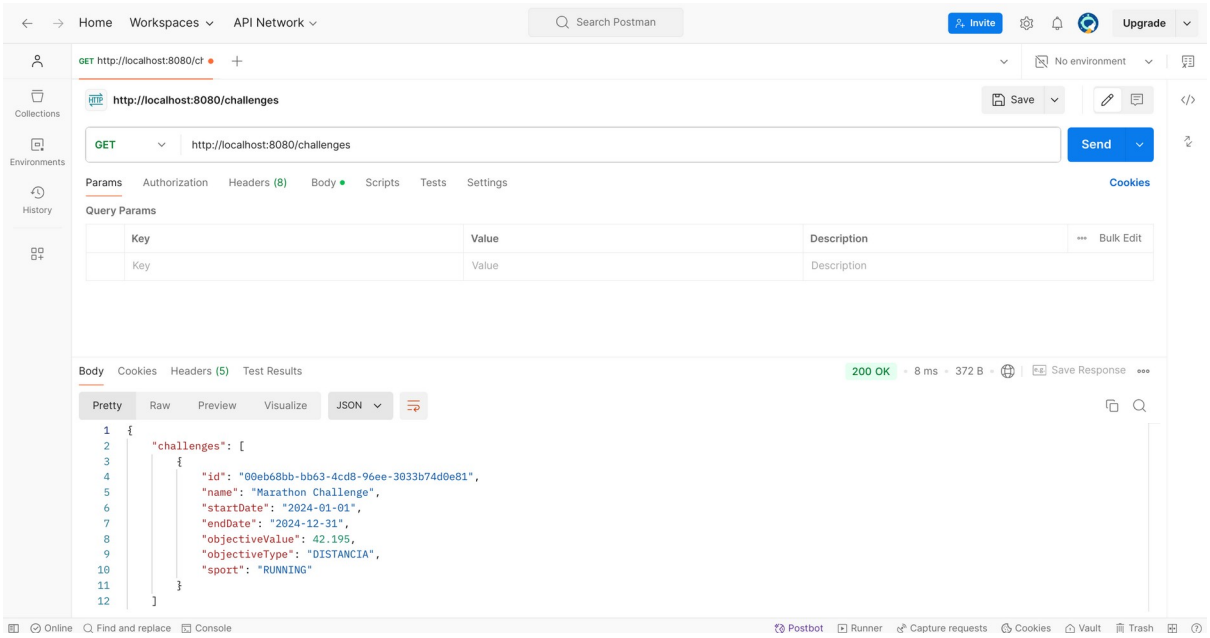
Obtener sesiones de entrenamiento.



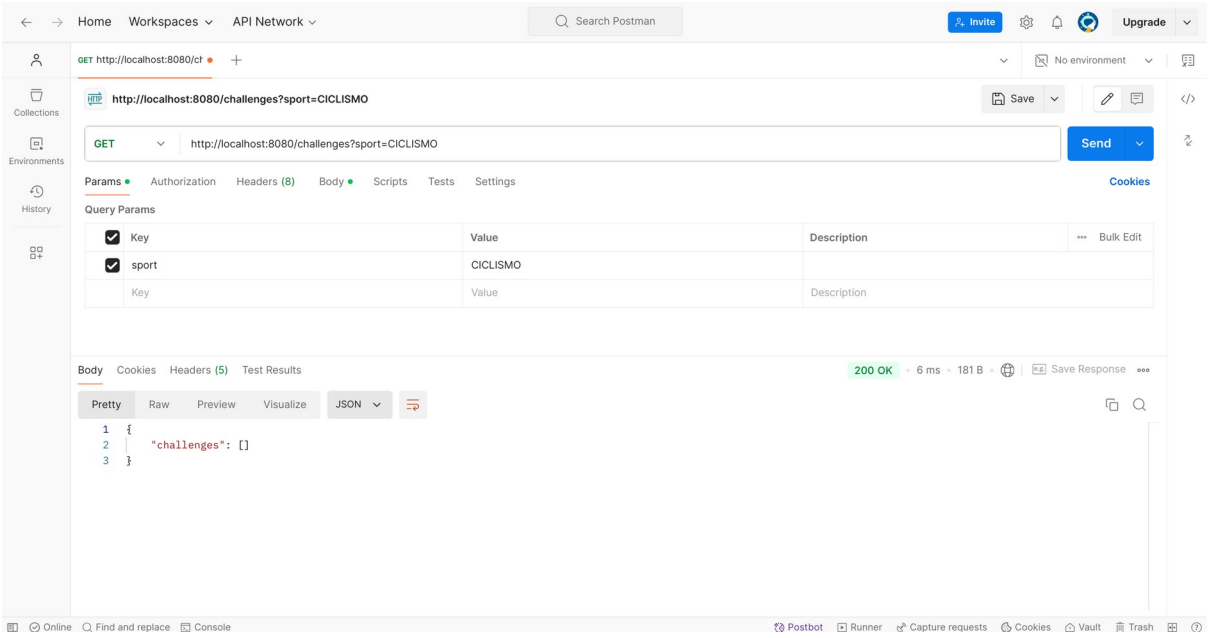
Filtrar y obtener sesiones de entrenamiento.



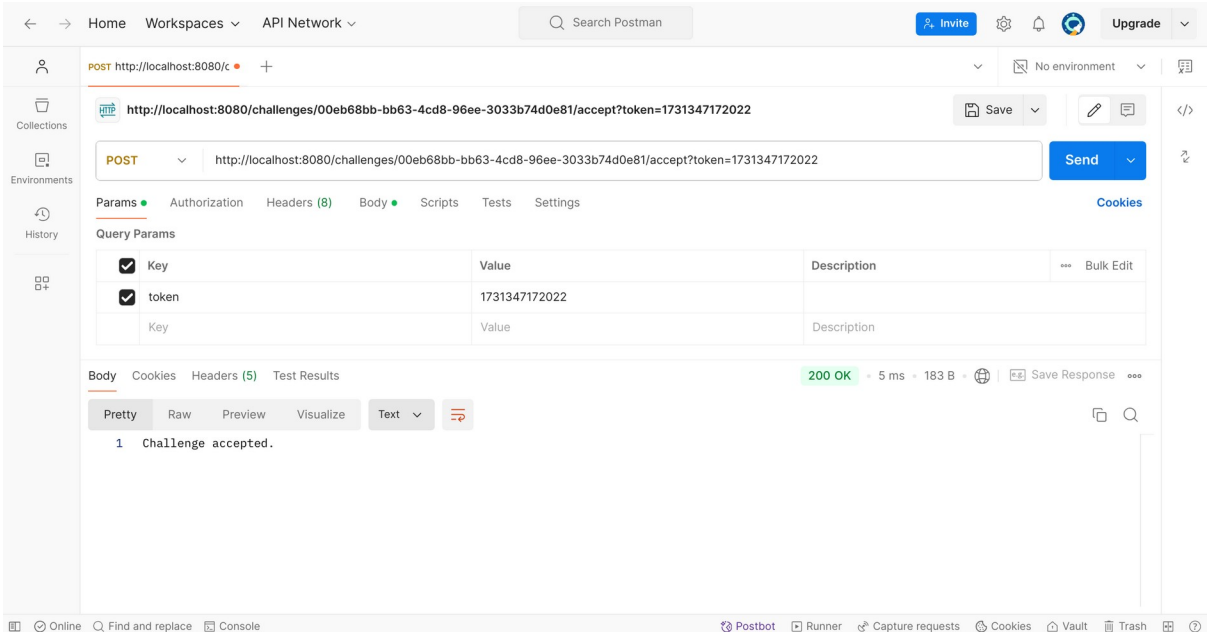
Obtener retos activos.



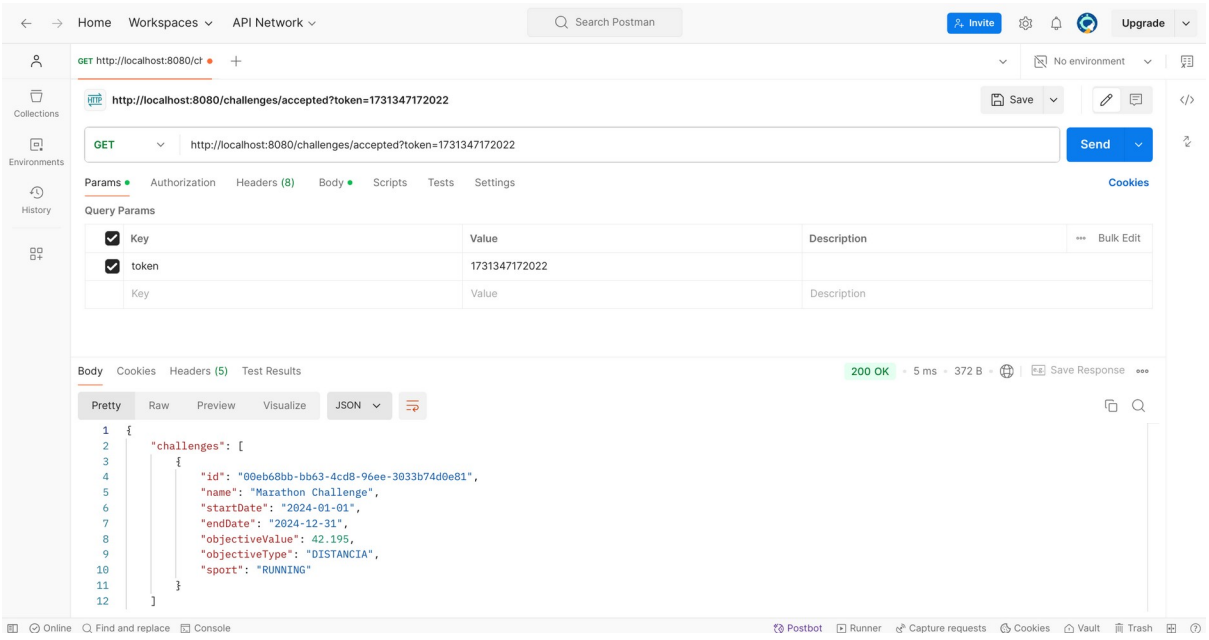
Filtrar y obtener retos activos.



Aceptar un reto.



Retos aceptados.



Véase que solo se asocia al usuario correspondiente; los demás no lo tienen como aceptado.