



Entrega 1
Grupo DS-5
Iker Alvis Veloso
Mikel Garduño Garcia

Diseño detallado del API del Servidor Central.....	2
API de Control de Usuarios.....	2
API de Simulación de Strava.....	4
Modelado del primer diagrama de clases.....	8
Modelado de un diagrama de secuencia a nivel de componentes.....	9
Login Usuario.....	9
Implementación del primer prototipo del Servidor Central.....	10
Validación del Prototipo utilizando Postman.....	10
Registrar Usuario.....	10
Login.....	10
Logout.....	11
Consultar usuarios.....	11
Crear un reto.....	12
Aceptar reto.....	12
Consultar retos.....	13
Crear un entrenamiento.....	13
Consultar retos activos (sin parámetros).....	14
Consultar retos activos (con parámetros).....	14
Consultar retos activos (con parámetros).....	15
Consultar retos aceptados (con funcionalidad del progreso).....	15
Consultar entrenamientos del usuario.....	16
Uso de Inteligencia Artificial en el proyecto.....	16

Diseño detallado del API del Servidor Central

API de Control de Usuarios

1. Registro de Usuario

Endpoint: `/auth/registroUsuario`

Método: POST

Descripción: Permite registrar un nuevo usuario en el sistema mediante Google o Meta.

Parámetros de Entrada:

- **Query Parameters:**
 - `nombre` (string, obligatorio): Nombre del usuario.
 - `fechaNacimiento` (string, obligatorio): Fecha de nacimiento (formato DD/MM/AAAA).
 - `peso` (float, obligatorio): Peso en kg.
 - `altura` (float, obligatorio): Altura en cm.
 - `frecuenciaCardiacaMax` (int, obligatorio): Frecuencia cardíaca máxima.
 - `frecuenciaCardiacaReposo` (int, obligatorio): Frecuencia cardíaca en reposo.

Cuerpo (JSON):

```
{  
  "email": "string",  
  "contrasena": "string",  
  "tipoLogin": "GOOGLE | META"  
}
```

Respuestas:

- **200:** Usuario registrado exitosamente.
 - **409:** El usuario ya existe.
 - **500:** Error interno del servidor.
 - **401:** Credenciales incorrectas.
-

2. Login

Endpoint: `/auth/login`

Método: POST

Descripción: Permite a un usuario iniciar sesión con su email y contraseña, devolviendo un token en caso de éxito.

Cuerpo (JSON):

```
{  
  "email": "string",  
  "contrasena": "string",  
  "tipoLogin": "GOOGLE | META"  
}
```

Respuestas:

- **200:** Inicio de sesión exitoso (devuelve un token).
 - **401:** Credenciales inválidas.
 - **409:** Usuario no registrado.
 - **500:** Error interno del servidor.
-

3. Logout

Endpoint: `/auth/logout`

Método: POST

Descripción: Cierra la sesión del usuario.

Cuerpo (JSON):

`"string"` (Token de autorización)

Respuestas:

- **204:** Sesión cerrada exitosamente.
 - **401:** Token inválido.
-

4. Consultar Usuarios

Endpoint: `/auth/usuarios`

Método: GET

Descripción: Devuelve la lista de usuarios registrados.

Respuestas:

- **200:** Lista obtenida exitosamente.
- **204:** No hay usuarios registrados.
- **500:** Error interno del servidor.

API de Simulación de Strava

1. Crear Reto

Endpoint: `/api/reto`

Método: POST

Descripción: Permite crear un nuevo reto.

Parámetros de Entrada (Query):

- **nombre** (string, obligatorio): Nombre del reto.
- **deporte** (string, obligatorio): Tipo de deporte.
- **objetivoDistancia** (float, obligatorio): Distancia objetivo.
- **objetivoTiempo** (int, obligatorio): Tiempo objetivo (en minutos).
- **fechaInicio** (string, obligatorio): Fecha de inicio (formato DD/MM/AAAA).
- **fechaFin** (string, obligatorio): Fecha de fin (formato DD/MM/AAAA).

Respuestas:

- **200:** Reto creado exitosamente.
 - **409:** Reto ya existe.
 - **500:** Error interno del servidor.
-

2. Aceptar Reto

Endpoint: `/api/retos/{nombreReto}/aceptar`

Método: POST

Descripción: Permite aceptar un reto.

Parámetros de Entrada:

- **Path Parameter:**
 - `nombreReto` (string, obligatorio): Nombre del reto.

Cuerpo (JSON):

`"string"` (Token de autorización)

-

Respuestas:

- **200:** Reto aceptado exitosamente.
 - **409:** Reto ya aceptado.
 - **500:** Error interno del servidor.
-

3. Consultar Retos

Endpoint: `/api/retos`

Método: GET

Descripción: Devuelve la lista de todos los retos creados.

Respuestas:

- **200:** Lista obtenida exitosamente.
 - **500:** Error interno del servidor.
-

4. Crear Entrenamiento

Endpoint: `/api/entrenamiento`

Método: POST

Descripción: Crea un nuevo entrenamiento para el usuario.

Parámetros de Entrada (Query):

- **titulo** (string, obligatorio): Nombre del entrenamiento.
- **deporte** (string, obligatorio): Tipo de deporte.
- **distancia** (float, obligatorio): Distancia recorrida (en km).
- **duracion** (int, obligatorio): Duración (en minutos).
- **fechaInicio** (string, obligatorio): Fecha del entrenamiento.
- **horaInicio** (string, obligatorio): Hora de inicio.
- **token** (string, obligatorio): Token de autorización.

Respuestas:

- **200**: Entrenamiento creado exitosamente.
 - **401**: Usuario no autorizado.
 - **500**: Error interno del servidor.
-

5. Consultar Retos Activos

Endpoint: `/api/retosActivos`

Método: GET

Descripción: Devuelve la lista de retos activos (no finalizados).

Parámetros de Entrada (Query):

- **fecha** (string, opcional): Fecha de consulta (formato DD/MM/AAAA).
- **deporte** (string, opcional): Filtrar por tipo de deporte.

Respuestas:

- **200**: Lista de retos activos obtenida exitosamente.
 - **400**: Fecha o deporte inválidos.
 - **500**: Error interno del servidor.
-

6. Consultar Retos Aceptados

Endpoint: `/api/retosAceptados`

Método: GET

Descripción: Devuelve la lista de retos aceptados por el usuario junto con su progreso.

Parámetros de Entrada (Query):

- **token** (string, obligatorio): Token de autorización.

Respuestas:

- **200:** Lista de retos aceptados obtenida exitosamente.
 - **400:** Usuario no especificado.
 - **409:** Usuario no existe.
 - **500:** Error interno del servidor.
-

7. Consultar Entrenamientos

Endpoint: `/api/entrenamientos/{fechaInicio}/{fechaFin}`

Método: GET

Descripción: Devuelve la lista completa de entrenamientos realizados por el usuario dentro del rango de fechas.

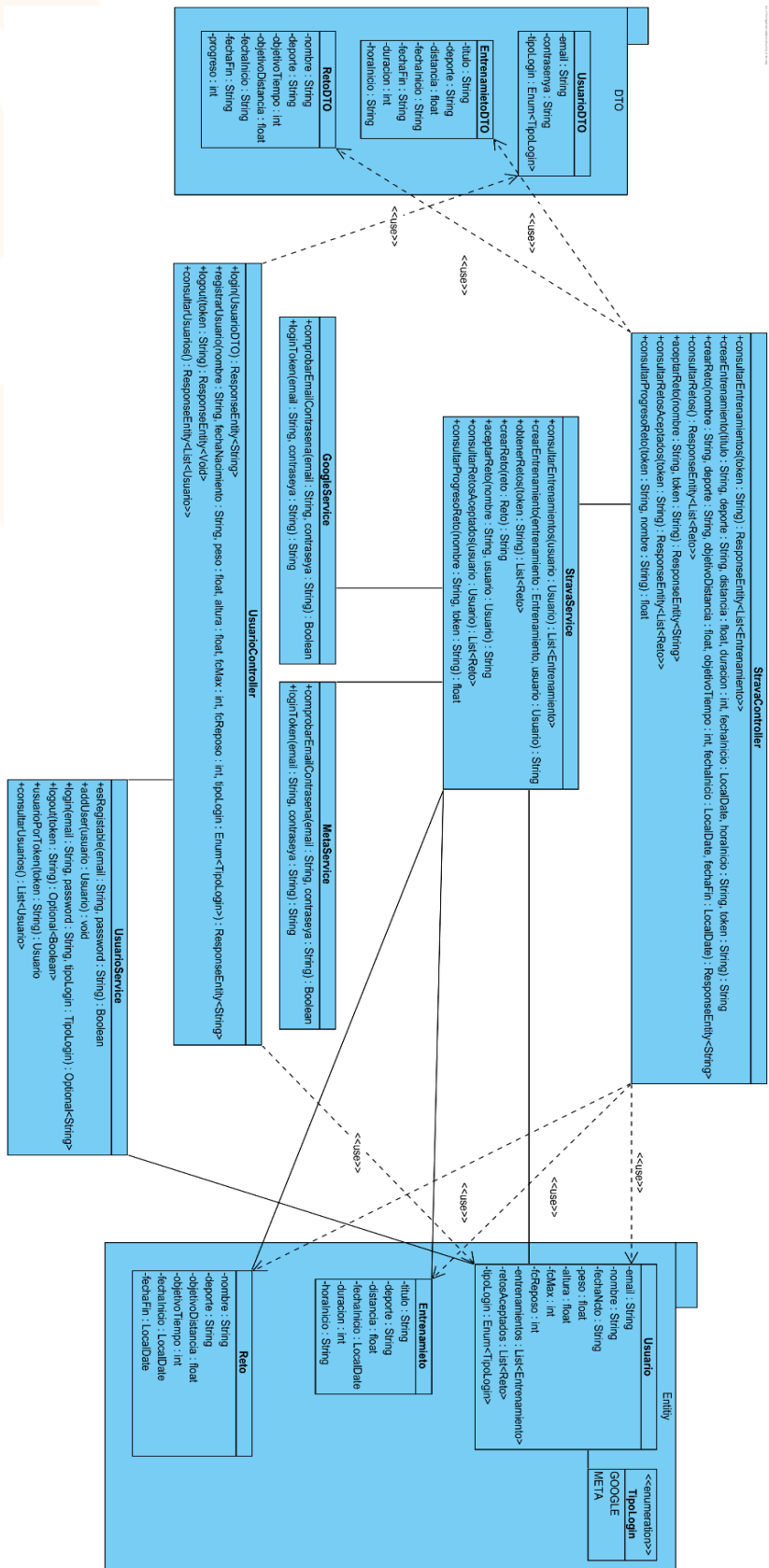
Parámetros de Entrada:

- **Path Parameters:**
 - **fechaInicio** (string, obligatorio): Fecha de inicio del rango (formato DD/MM/AAAA).
 - **fechaFin** (string, obligatorio): Fecha de fin del rango (formato DD/MM/AAAA).
- **Query Parameters:**
 - **token** (string, obligatorio): Token de autorización.

Respuestas:

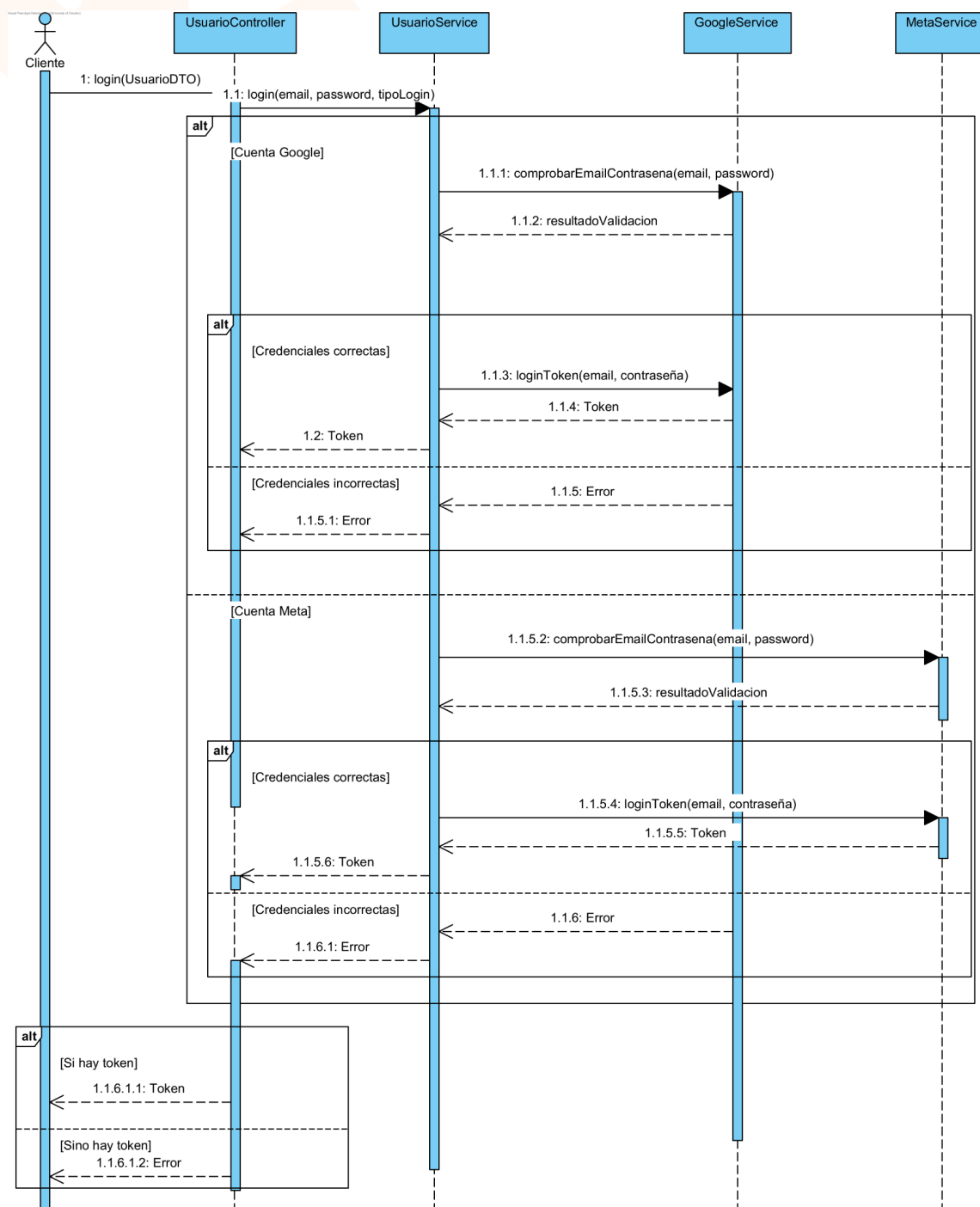
- **200:** Lista de entrenamientos obtenida exitosamente.
- **401:** Usuario no autorizado.
- **409:** Usuario no existe.
- **500:** Error interno del servidor.

Modelado del primer diagrama de clases



Modelado de un diagrama de secuencia a nivel de componentes

Login Usuario

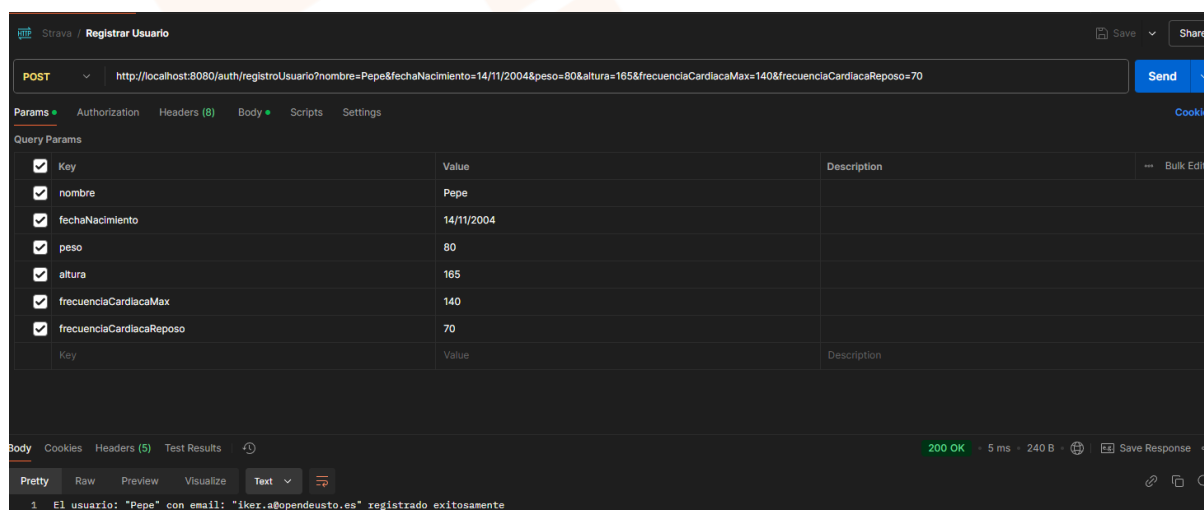


Implementación del primer prototipo del Servidor Central

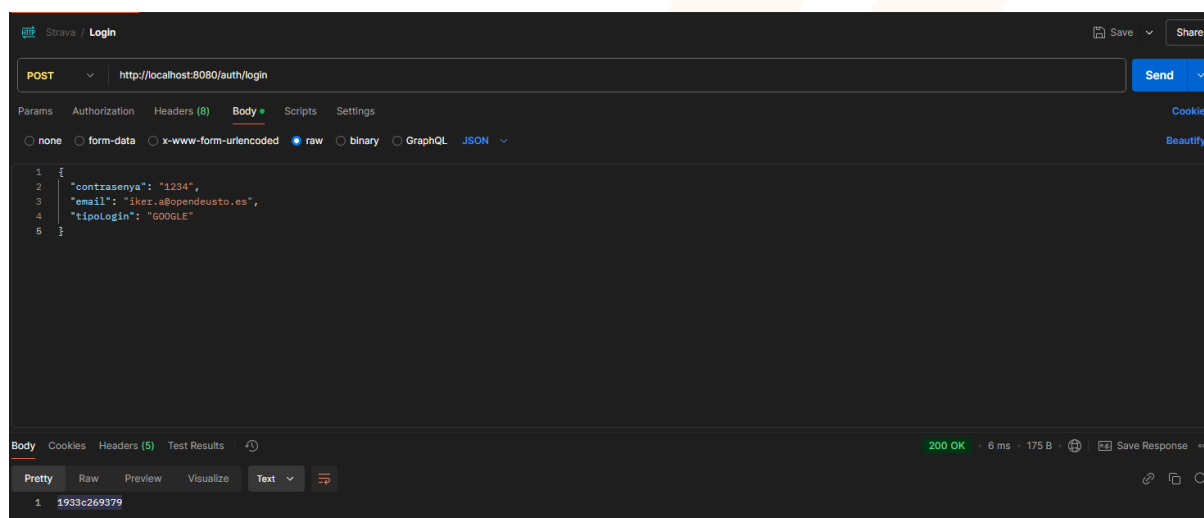
1. Descarga el proyecto desde GitHub clonando el repositorio o descargando el archivo ZIP.
2. Abre la carpeta del proyecto en tu equipo.
3. inicia el servidor y accede a <http://localhost:8080/swagger-ui/index.html#/>

Validación del Prototipo utilizando Postman

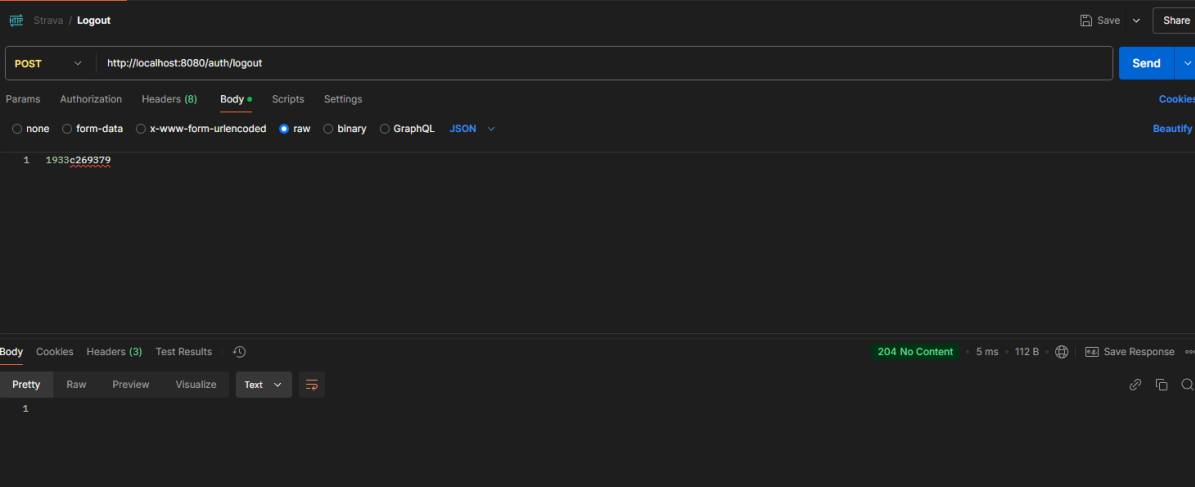
Registrar Usuario



Login



Logout



Strava / Logout

POST http://localhost:8080/auth/logout

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 1933c269379

Body Cookies Headers (3) Test Results

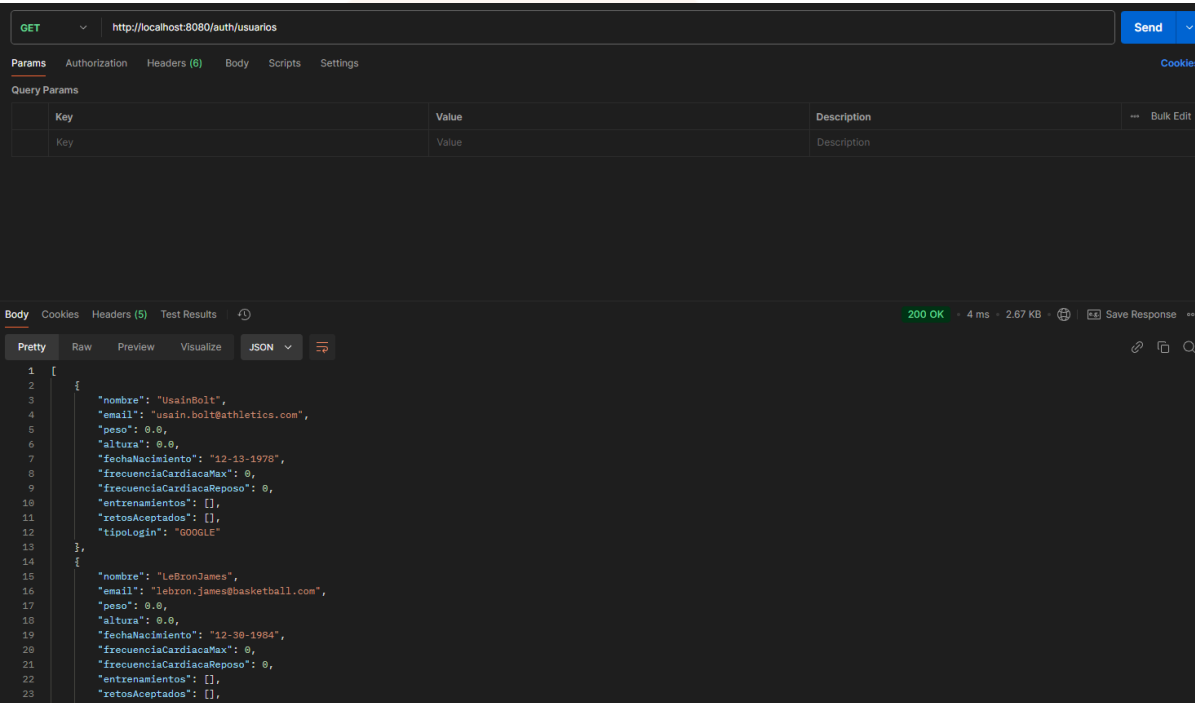
204 No Content · 5 ms · 112 B

Save Response

Pretty Raw Preview Visualize Text

1

Consultar usuarios



GET http://localhost:8080/auth/usuarios

Send

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (5) Test Results

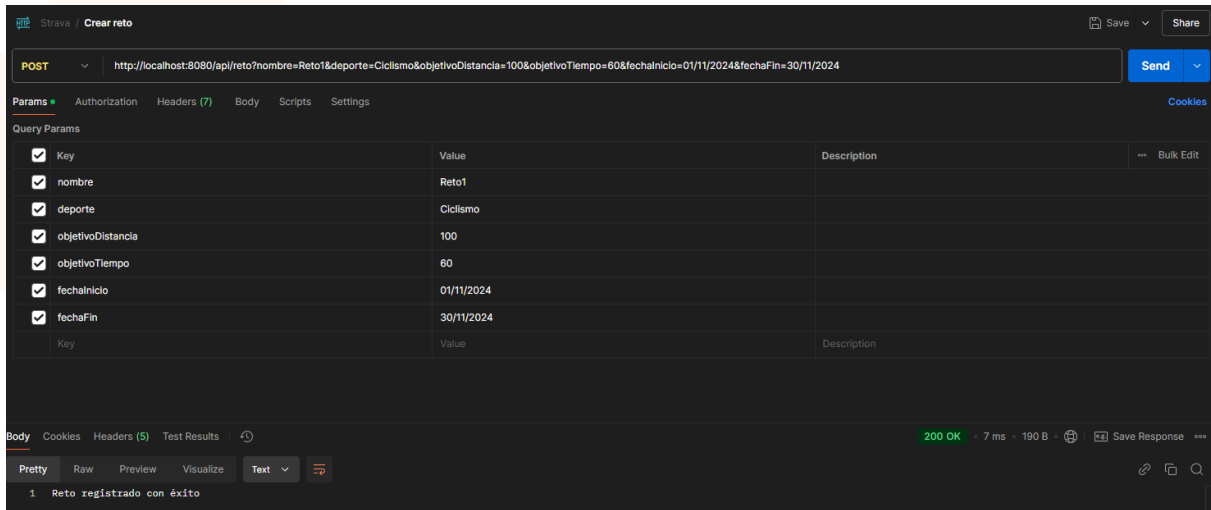
200 OK · 4 ms · 2.67 KB

Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "nombre": "UsainBolt",
4     "email": "usain.bolt@athletics.com",
5     "peso": 0.0,
6     "altura": 0.0,
7     "fechaNacimiento": "12-13-1978",
8     "frecuenciaCardiacaMax": 0,
9     "frecuenciaCardiacaReposo": 0,
10    "entrenamientos": [],
11    "retosAceptados": [],
12    "tipoLogin": "GOOGLE"
13  },
14  {
15    "nombre": "LeBronJames",
16    "email": "lebron.james@basketball.com",
17    "peso": 0.0,
18    "altura": 0.0,
19    "fechaNacimiento": "12-30-1984",
20    "frecuenciaCardiacaMax": 0,
21    "frecuenciaCardiacaReposo": 0,
22    "entrenamientos": [],
23    "retosAceptados": [],
24    "tipoLogin": "GOOGLE"
25  }
26 ]
```

Crear un reto



The screenshot shows a REST client interface for a Strava API. The request is a POST to `http://localhost:8080/api/retos?nombre=Reto1&deporte=Ciclismo&objetivoDistancia=100&objetivoTiempo=60&fechalnicio=01/11/2024&fechaFin=30/11/2024`. The response is a 200 OK status with a body containing the text "Reto registrado con éxito".

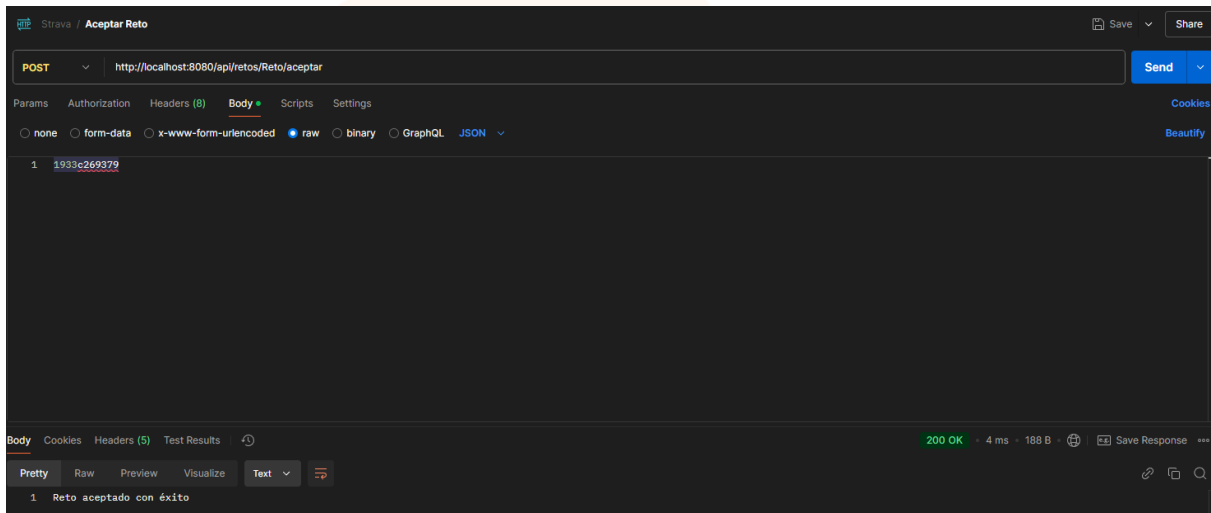
Query Params

Key	Value	Description
<input checked="" type="checkbox"/> nombre	Reto1	
<input checked="" type="checkbox"/> deporte	Ciclismo	
<input checked="" type="checkbox"/> objetivoDistancia	100	
<input checked="" type="checkbox"/> objetivoTiempo	60	
<input checked="" type="checkbox"/> fechalnicio	01/11/2024	
<input checked="" type="checkbox"/> fechaFin	30/11/2024	

Body

1 Reto registrado con éxito

Aceptar reto



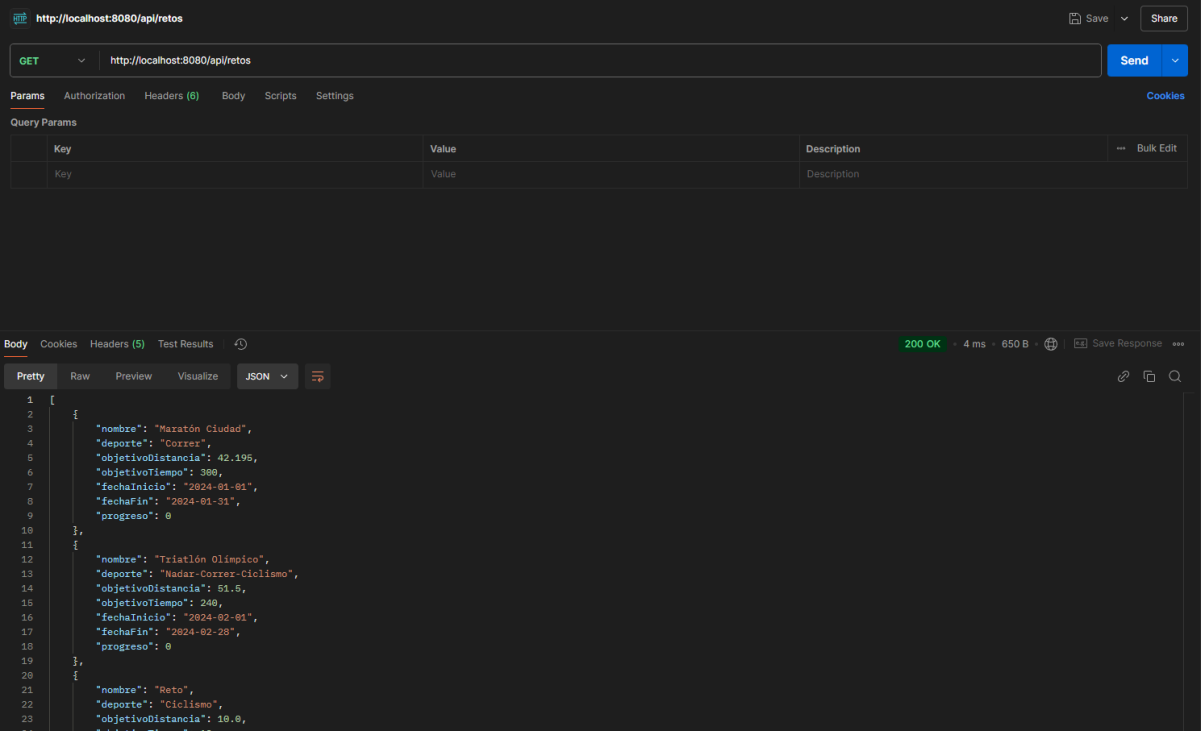
The screenshot shows a REST client interface for a Strava API. The request is a POST to `http://localhost:8080/api/retos/Reto/acceptar`. The response is a 200 OK status with a body containing the text "Reto aceptado con éxito".

Body

1 1933c269379

1 Reto aceptado con éxito

Consultar retos



http://localhost:8080/api/retos

GET http://localhost:8080/api/retos

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

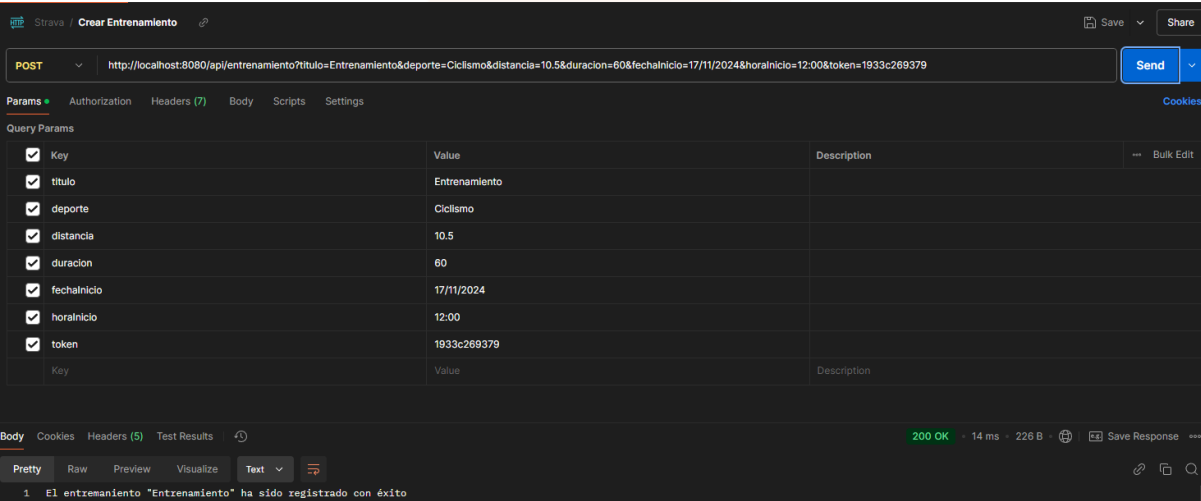
Body Cookies Headers (5) Test Results

200 OK · 4 ms · 650 B

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "nombre": "Maratón Ciudad",
4     "deporte": "Correr",
5     "objetivoDistancia": 42.195,
6     "objetivoTiempo": 900,
7     "fechaInicio": "2024-01-01",
8     "fechaFin": "2024-01-31",
9     "progreso": 0
10  },
11  {
12    "nombre": "Triatlón Olímpico",
13    "deporte": "Nadar-Correr-Ciclismo",
14    "objetivoDistancia": 51.5,
15    "objetivoTiempo": 240,
16    "fechaInicio": "2024-02-01",
17    "fechaFin": "2024-02-28",
18    "progreso": 0
19  },
20  {
21    "nombre": "Reto",
22    "deporte": "Ciclismo",
23    "objetivoDistancia": 19.0,
24    "objetivoTiempo": 10
```

Crear un entrenamiento



Strava / Crear Entrenamiento

POST http://localhost:8080/api/entrenamiento?titulo=Entrenamiento&deporte=Ciclismo&distancia=10.5&duracion=60&fechaInicio=17/11/2024&horaInicio=12:00&token=1933c269379

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value	Description
<input checked="" type="checkbox"/> titulo	Entrenamiento	
<input checked="" type="checkbox"/> deporte	Ciclismo	
<input checked="" type="checkbox"/> distancia	10.5	
<input checked="" type="checkbox"/> duracion	60	
<input checked="" type="checkbox"/> fechaInicio	17/11/2024	
<input checked="" type="checkbox"/> horaInicio	12:00	
<input checked="" type="checkbox"/> token	1933c269379	
Key	Value	Description

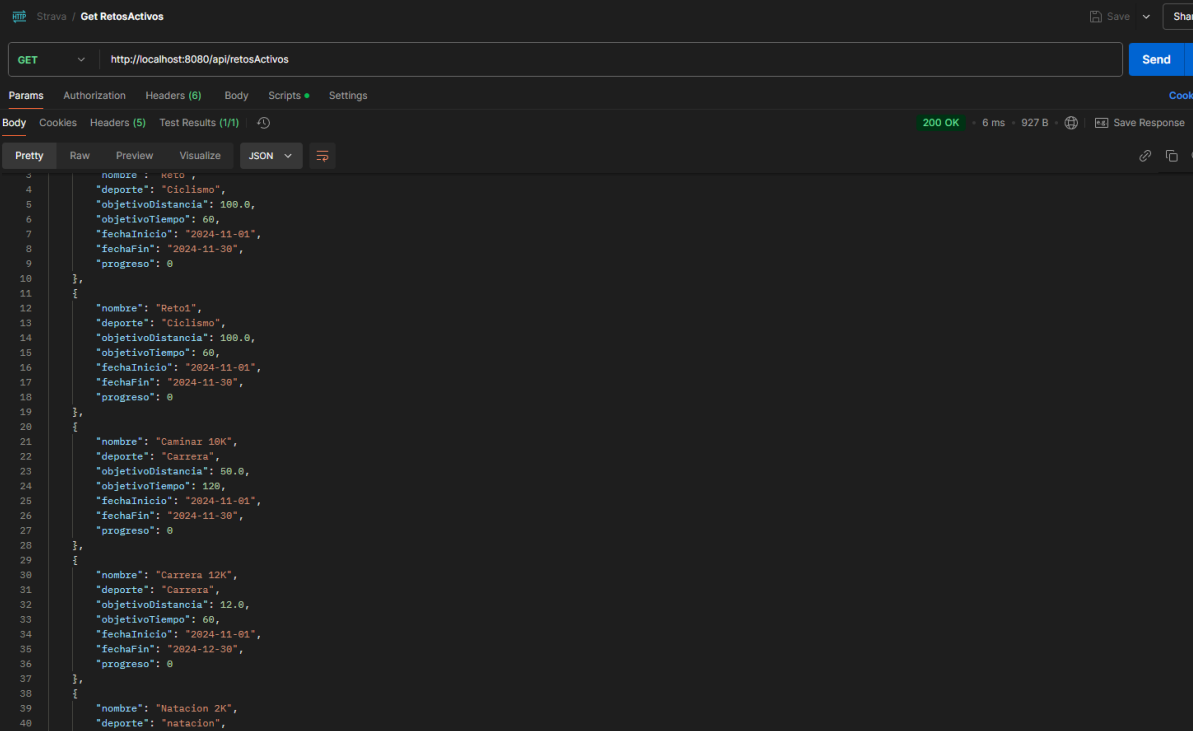
Body Cookies Headers (5) Test Results

200 OK · 14 ms · 226 B

Pretty Raw Preview Visualize Text

```
1 El entrenamiento "Entrenamiento" ha sido registrado con éxito
```

Consultar retos activos (sin parámetros)



Strava / Get RetosActivos

GET <http://localhost:8080/api/retosActivos> Send

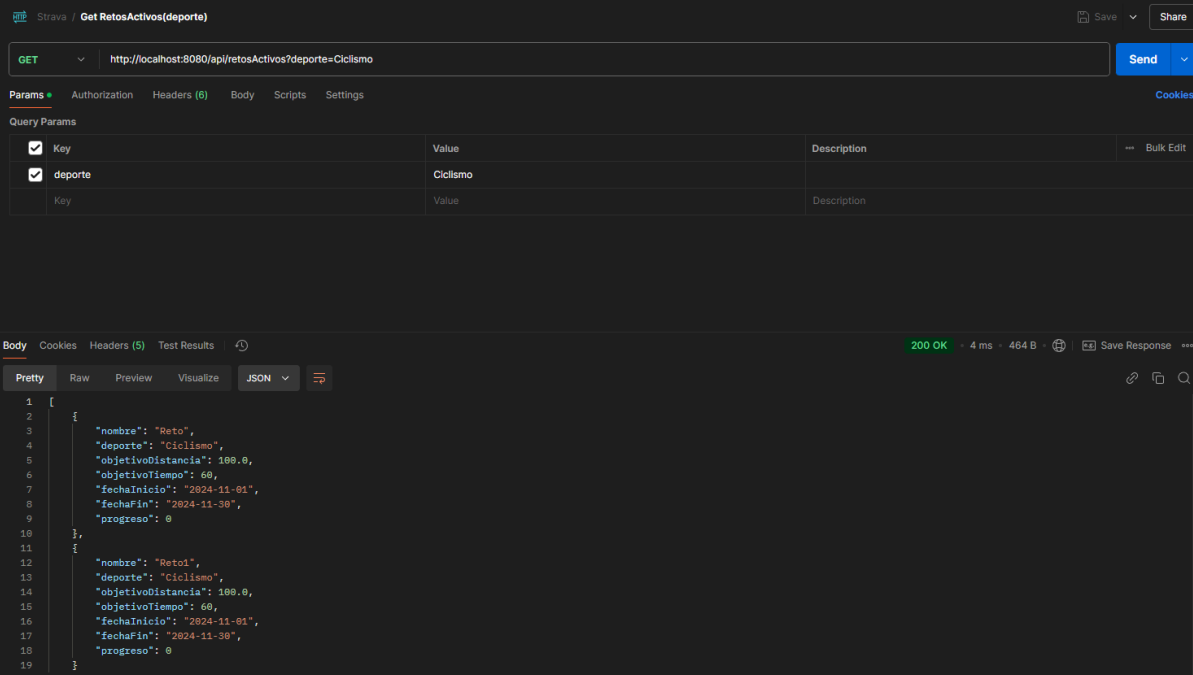
Params Authorization Headers (6) Body Scripts Settings

Body Cookies Headers (5) Test Results (1/1) 200 OK · 6 ms · 927 B Save Response

Pretty Raw Preview Visualize JSON

```
3  {
4    "nombre": "Reto",
5    "deporte": "Ciclismo",
6    "objetivoDistancia": 100.0,
7    "objetivoTiempo": 60,
8    "fechaInicio": "2024-11-01",
9    "fechaFin": "2024-11-30",
10   "progreso": 0
11 },
12 {
13   "nombre": "Retos",
14   "deporte": "Ciclismo",
15   "objetivoDistancia": 100.0,
16   "objetivoTiempo": 60,
17   "fechaInicio": "2024-11-01",
18   "fechaFin": "2024-11-30",
19   "progreso": 0
20 },
21 {
22   "nombre": "Caminar 10K",
23   "deporte": "Carrera",
24   "objetivoDistancia": 50.0,
25   "objetivoTiempo": 120,
26   "fechaInicio": "2024-11-01",
27   "fechaFin": "2024-11-30",
28   "progreso": 0
29 },
30 {
31   "nombre": "Carrera 12K",
32   "deporte": "Carrera",
33   "objetivoDistancia": 12.0,
34   "objetivoTiempo": 60,
35   "fechaInicio": "2024-11-01",
36   "fechaFin": "2024-12-30",
37   "progreso": 0
38 },
39 {
40   "nombre": "Natacion 2K",
41   "deporte": "Natacion",
42   "objetivoDistancia": 2.0,
```

Consultar retos activos (con parámetros)



Strava / Get RetosActivos(deporte)

GET <http://localhost:8080/api/retosActivos?deporte=Ciclismo> Send

Params Authorization Headers (6) Body Scripts Settings

Query Params

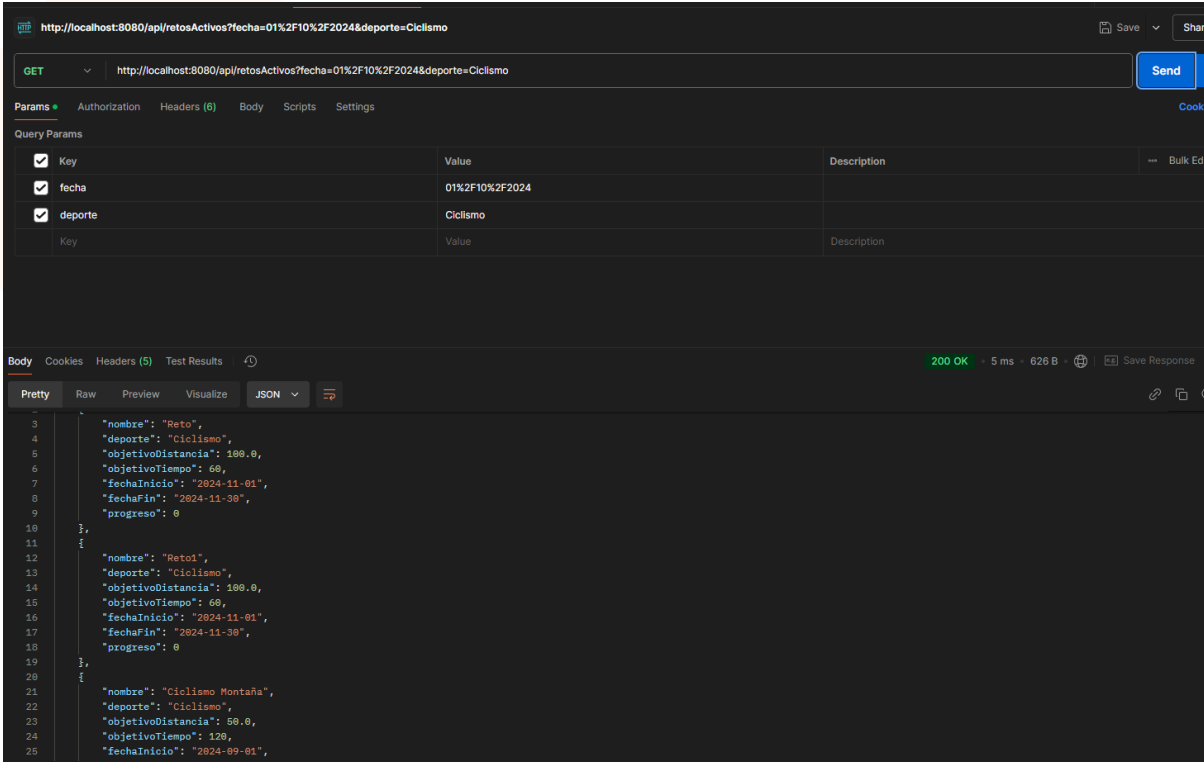
Key	Value	Description
<input checked="" type="checkbox"/> deporte	Ciclismo	

Body Cookies Headers (5) Test Results (1/1) 200 OK · 4 ms · 464 B Save Response

Pretty Raw Preview Visualize JSON

```
1  [
2    {
3      "nombre": "Reto",
4      "deporte": "Ciclismo",
5      "objetivoDistancia": 100.0,
6      "objetivoTiempo": 60,
7      "fechaInicio": "2024-11-01",
8      "fechaFin": "2024-11-30",
9      "progreso": 0
10   },
11   {
12     "nombre": "Retos",
13     "deporte": "Ciclismo",
14     "objetivoDistancia": 100.0,
15     "objetivoTiempo": 60,
16     "fechaInicio": "2024-11-01",
17     "fechaFin": "2024-11-30",
18     "progreso": 0
19   }
20 ]
```

Consultar retos activos (con parámetros)



URL: `http://localhost:8080/api/retosActivos?fecha=01%2F10%2F2024&deporte=Ciclismo`

Method: **GET**

Query Params:

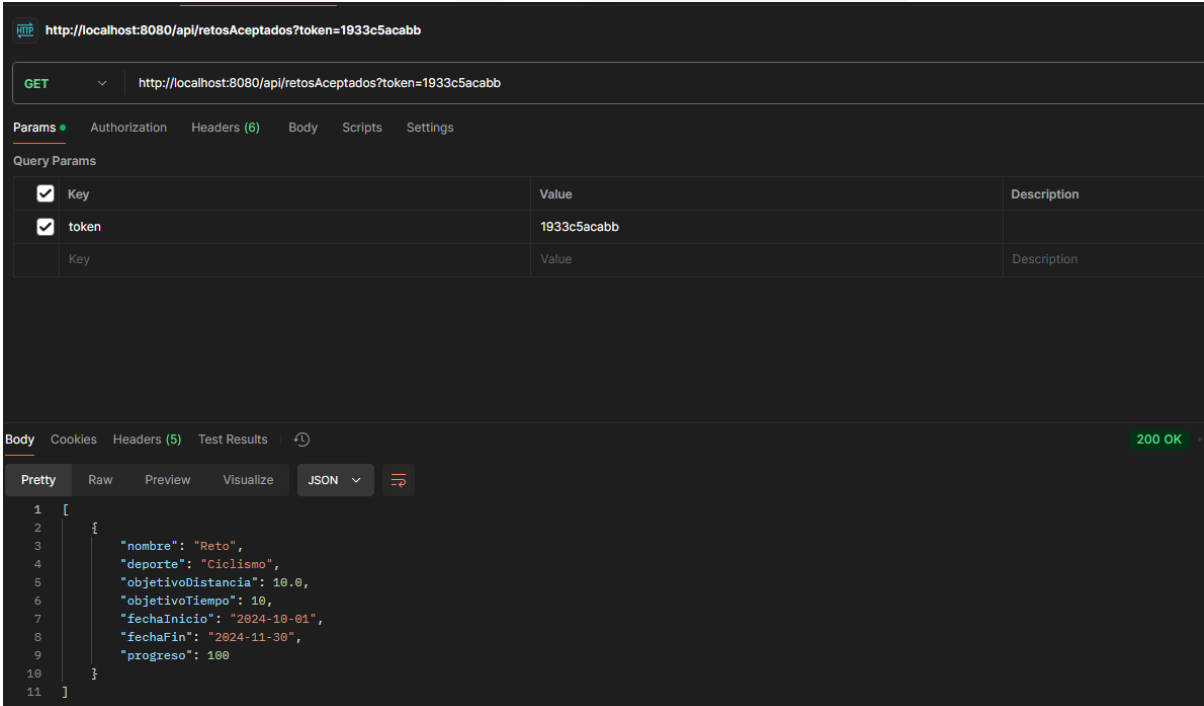
Key	Value	Description
fecha	01%2F10%2F2024	
deporte	Ciclismo	

Body:

```
200 OK • 5 ms • 626 B
```

```
[{"nombre": "Reto", "deporte": "Ciclismo", "objetivoDistancia": 100.0, "objetivoTiempo": 60, "fechaInicio": "2024-11-01", "fechaFin": "2024-11-30", "progreso": 0}, {"nombre": "Retos", "deporte": "Ciclismo", "objetivoDistancia": 100.0, "objetivoTiempo": 60, "fechaInicio": "2024-11-01", "fechaFin": "2024-11-30", "progreso": 0}, {"nombre": "Ciclismo Montaña", "deporte": "Ciclismo", "objetivoDistancia": 50.0, "objetivoTiempo": 120, "fechaInicio": "2024-09-01", "fechaFin": "2024-09-01", "progreso": 0}]
```

Consultar retos aceptados (con funcionalidad del progreso)



URL: `http://localhost:8080/api/retosAceptados?token=1933c5acabb`

Method: **GET**

Query Params:

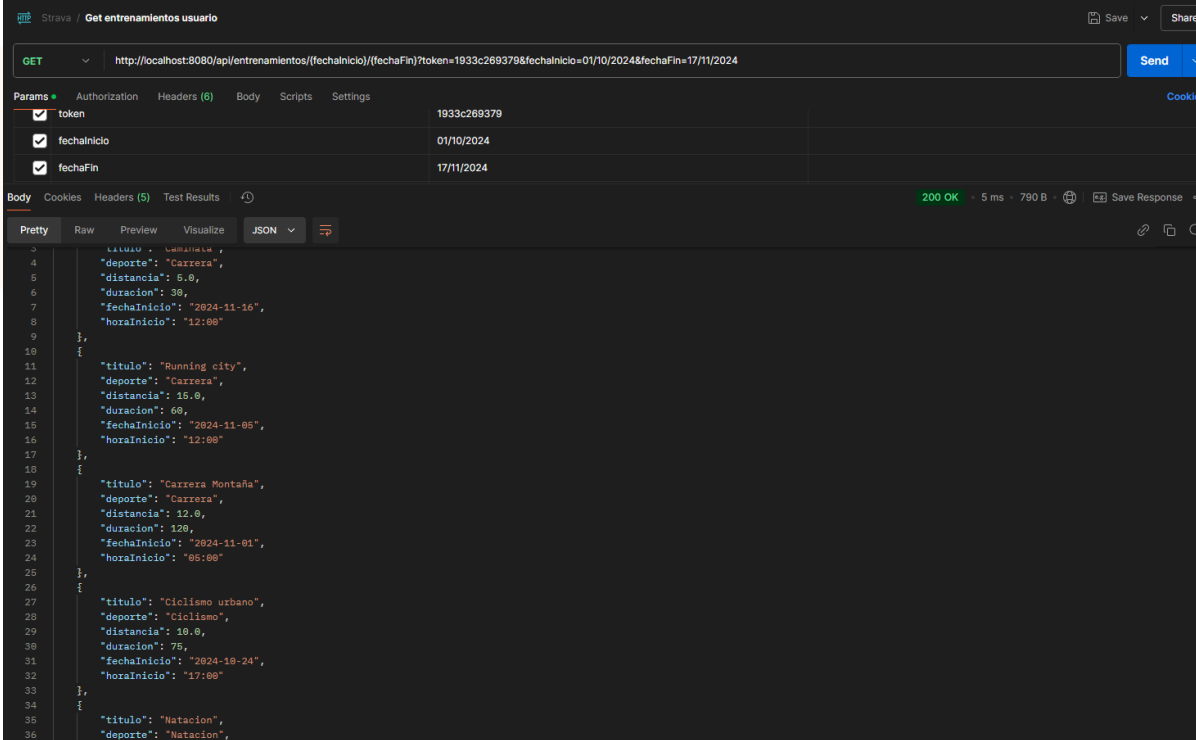
Key	Value	Description
token	1933c5acabb	

Body:

```
200 OK
```

```
[{"nombre": "Reto", "deporte": "Ciclismo", "objetivoDistancia": 10.0, "objetivoTiempo": 10, "fechaInicio": "2024-10-01", "fechaFin": "2024-11-30", "progreso": 100}]
```

Consultar entrenamientos del usuario



```
GET http://localhost:8080/api/entrenamientos/{fechaInicio}/{fechaFin}?token=1933c269379&fechaInicio=01/10/2024&fechaFin=17/11/2024

Params
token 1933c269379
fechaInicio 01/10/2024
fechaFin 17/11/2024

Body
200 OK - 5 ms - 790 B
Save Response

[{"deporte": "Carrera", "distancia": 6.0, "duracion": 30, "fechaInicio": "2024-11-16", "horaInicio": "12:00"}, {"titulo": "Running city", "deporte": "Carrera", "distancia": 15.0, "duracion": 60, "fechaInicio": "2024-11-05", "horaInicio": "12:00"}, {"titulo": "Carrera Montaña", "deporte": "Carrera", "distancia": 12.0, "duracion": 120, "fechaInicio": "2024-11-01", "horaInicio": "06:00"}, {"titulo": "Ciclismo urbano", "deporte": "Ciclismo", "distancia": 10.0, "duracion": 75, "fechaInicio": "2024-10-24", "horaInicio": "17:00"}, {"titulo": "Natacion", "deporte": "Natacion", "distancia": 1.0, "duracion": 15, "fechaInicio": "2024-10-20", "horaInicio": "08:00"}]
```

Uso de Inteligencia Artificial en el proyecto

En el desarrollo de este proyecto, se ha empleado el uso de herramientas basadas en inteligencia artificial para mejorar la funcionalidad, la solución de problemas de código o para la ayuda a la elaboración de diagramas. A continuación se describen las principales aplicaciones de IA utilizadas:

- **Generación de Documentación:** Hemos empleado herramientas de IA para ayudar en la generación automática de la documentación del API. Esto incluyó la creación de descripciones detalladas de los endpoints, parámetros y respuestas, lo que facilitó la implementación de Swagger. Además de la generación de los README en github y de su correcta estructura.
- **Optimización de Código:** Hemos utilizado la IA para analizar el código y proporcionar sugerencias para realizar mejoras en el rendimiento y la seguridad.
- **Análisis de Errores:** Para la depuración de errores, la IA fue útil en la identificación de problemas relacionados con el mapeo de los datos de los usuarios (tokens). Esto permitió una resolución más rápida de problemas como el conflicto de rutas en los métodos `consultarRetos()` y `consultarRetosActivos()`.
- **Creación de funciones :** Gracias a la IA se logró acelerar el proceso de desarrollo inicial, se utilizó como herramienta de apoyo para generar y prototipar funciones y sirvió como un punto de partida, pero las decisiones finales y ajustes necesarios fueron realizados por los desarrolladores. Además ha servido para ayudar con las diferencias entre el lenguaje de java y el de C++ usado el anterior curso.