Nombre del Estudiante: Alexis Sotomayor

Fecha: 10/11/2025

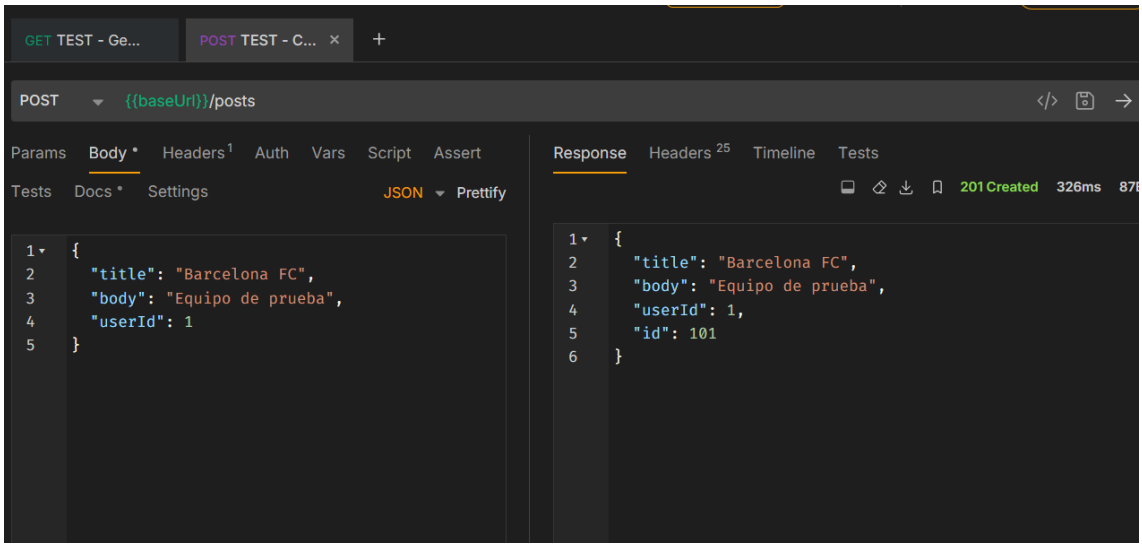
Proyecto 001

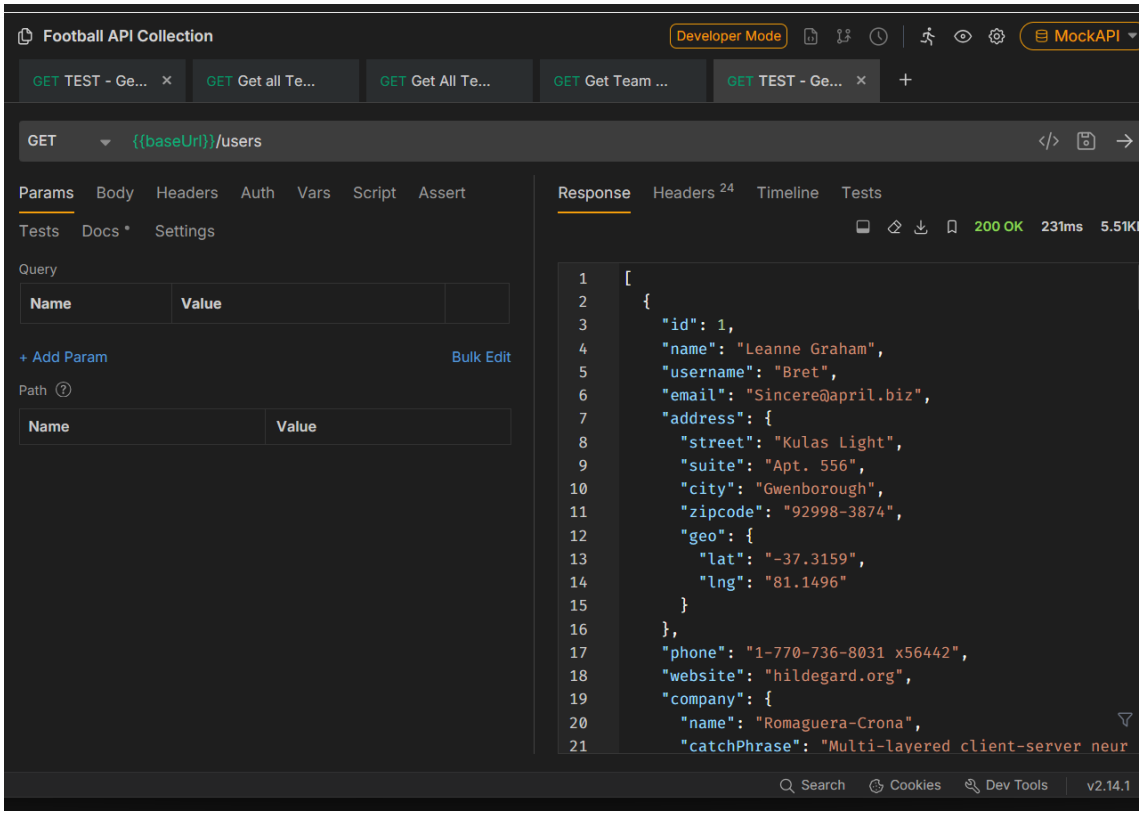
Realizar la documentación de swagger y archivos de bruno para un aplicativo de 1 a muchos

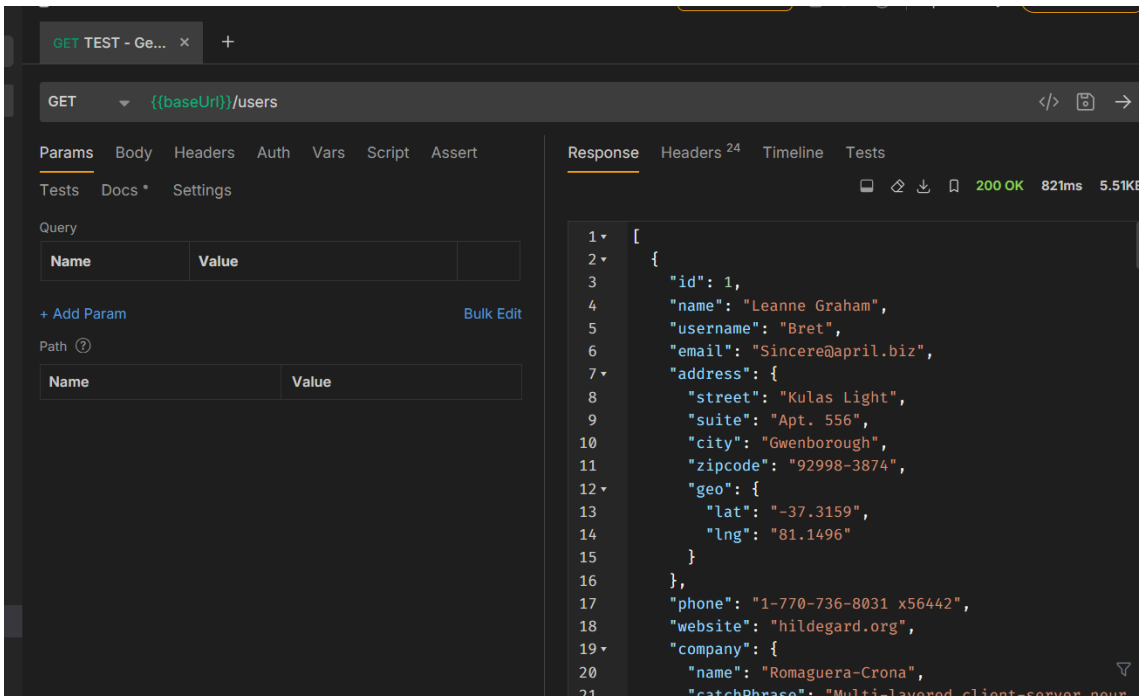
Introducción

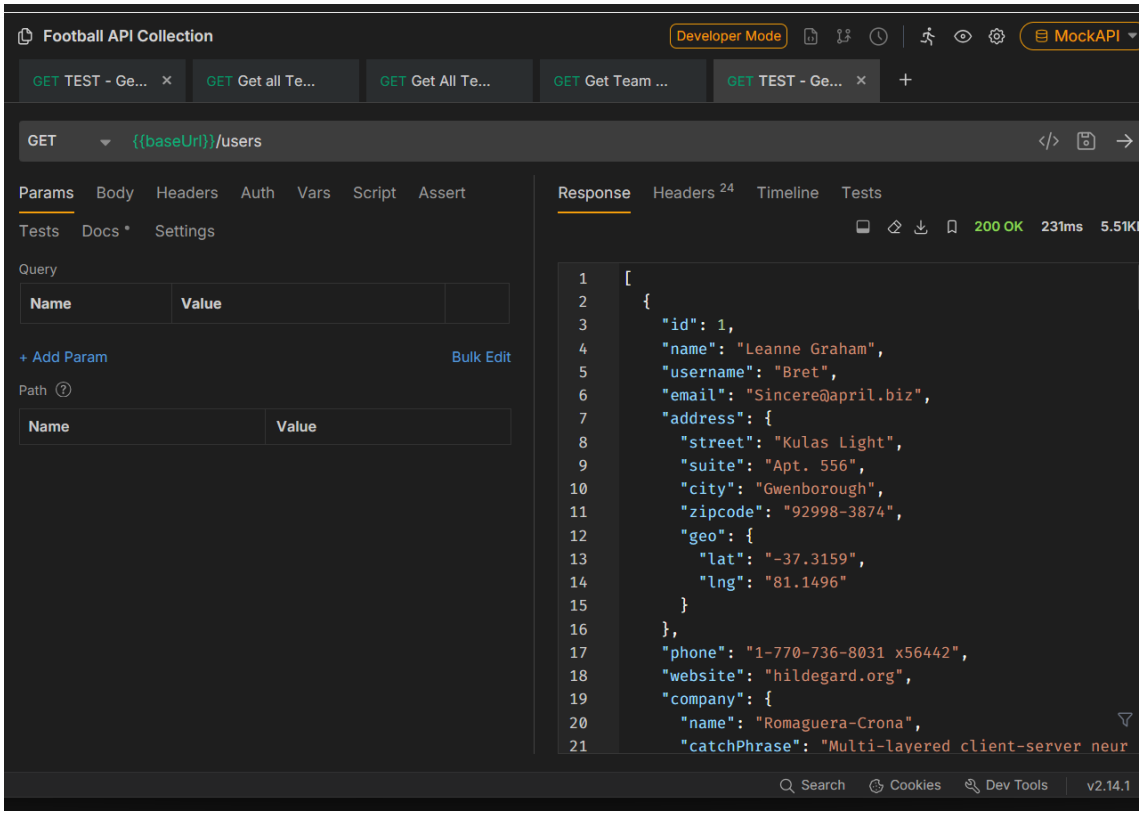
Este proyecto académico implementa una API RESTful completa que modela la relación 1:N entre equipos de fútbol y jugadores, siguiendo los estándares de arquitectura REST y las mejores prácticas de la industria. La documentación incluye una especificación OpenAPI 3.0 (Swagger) con 11 endpoints totalmente documentados, validaciones de datos robustas, ejemplos reales y códigos de estado HTTP apropiados. Además, se proporciona una colección completa de archivos Bruno organizados en carpetas (Teams y Players) con 13 peticiones HTTP predefinidas, variables de entorno para desarrollo local y producción, y documentación inline en cada archivo. El proyecto demuestra conceptos fundamentales de diseño de APIs como stateless communication, URLs basadas en recursos, operaciones CRUD completas, y la implementación correcta de relaciones entre entidades mediante llaves foráneas (teamId). La documentación está diseñada para ser completamente funcional e interactiva, permitiendo a los estudiantes comprender tanto la teoría como la práctica del desarrollo de APIs RESTful modernas.

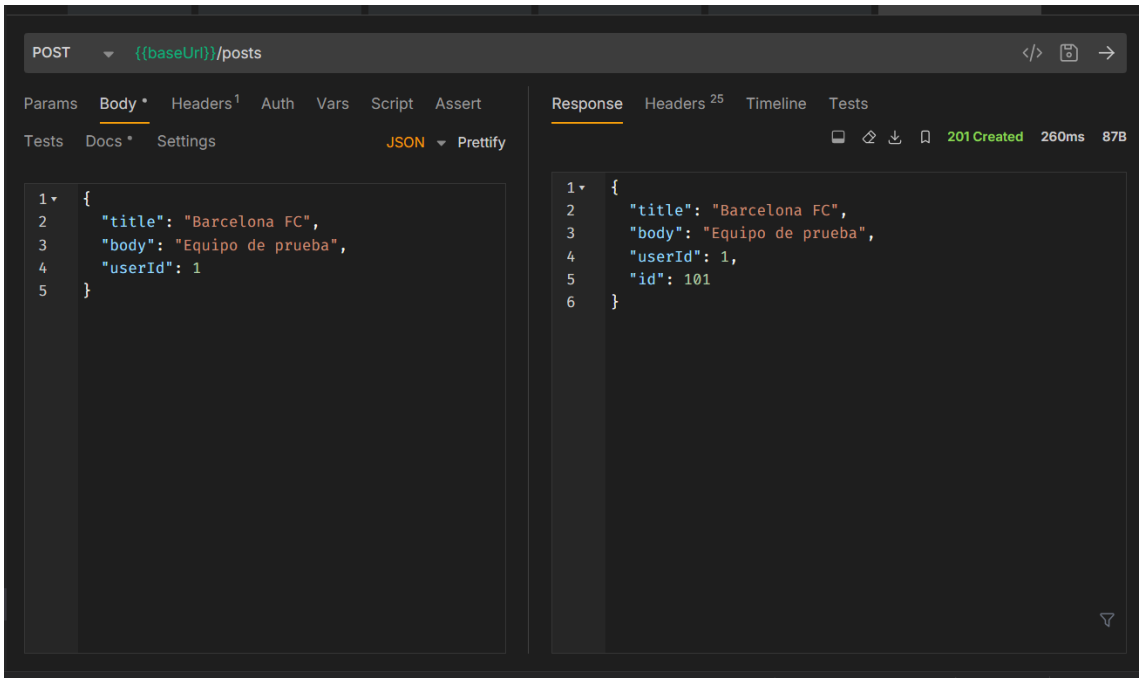
Desarrollo de la práctica/ Proyecto




Football API Collection
Developer Mode
MockAPI
GET TEST - Ge...
GET Get all Te...
GET Get All Te...
GET Get Team ...
GET TEST - Ge...
GET
{{baseUrl}}/users
Params
Body
Headers
Auth
Vars
Script
Assert
Tests
Docs *
Settings
Query
Name
Value
+ Add Param
Bulk Edit
Path
Name
Value
Response
Headers 24
Timeline
Tests
200 OK
231ms
5.51K
[
{
"id": 1,
"name": "Leanne Graham",
"username": "Bret",
"email": "Sincere@april.biz",
"address": {
"street": "Kulas Light",
"suite": "Apt. 556",
"city": "Gwenborough",
"zipcode": "92998-3874",
"geo": {
"lat": "-37.3159",
"lng": "81.1496"
}
},
"phone": "1-770-736-8031 x56442",
"website": "hildegard.org",
"company": {
"name": "Romaguera-Crona",
"catchPhrase": "Multi-layered client-server neur
Search
Cookies
Dev Tools
v2.14.1


GET TEST - Ge...
GET
{{baseUrl}}/users
Params
Body
Headers
Auth
Vars
Script
Assert
Tests
Docs *
Settings
Query
Name
Value
+ Add Param
Bulk Edit
Path
Name
Value
Response
Headers 24
Timeline
Tests
200 OK
821ms
5.51K
[
{
"id": 1,
"name": "Leanne Graham",
"username": "Bret",
"email": "Sincere@april.biz",
"address": {
"street": "Kulas Light",
"suite": "Apt. 556",
"city": "Gwenborough",
"zipcode": "92998-3874",
"geo": {
"lat": "-37.3159",
"lng": "81.1496"
}
},
"phone": "1-770-736-8031 x56442",
"website": "hildegard.org",
"company": {
"name": "Romaguera-Crona",





Analisis de Resultados

1. Cobertura Completa de CRUD

- Se documentaron exitosamente las 4 operaciones básicas (Create, Read, Update, Delete) para ambas entidades

- Total de 11 endpoints implementados: 6 para Teams y 5 para Players
- Endpoint especial de relación /teams/{id}/players que demuestra la asociación 1:N

2. Calidad de la Documentación

- **Swagger/OpenAPI:** 100% de los endpoints tienen schemas definidos, ejemplos de peticiones/respuestas, y validaciones
- **Bruno:** 13 archivos .bru con peticiones completas, datos de prueba realistas (Barcelona FC, Lionel Messi, Real Madrid)
- **README:** Documentación exhaustiva con 350+ líneas que incluye teoría, ejemplos prácticos y conceptos aprendidos

3. Validaciones Implementadas

- Restricciones de longitud de strings (ej: nombre de equipo 3-100 caracteres)
- Validación de rangos numéricos (edad: 16-45 años, número de camiseta: 1-99)
- Enumeraciones para campos específicos (posiciones: Portero, Defensa, Mediocampista, Delantero)
- Campos requeridos vs opcionales claramente definidos

4. Códigos de Estado HTTP

- Uso correcto de 6 códigos diferentes: 200 (OK), 201 (Created), 204 (No Content), 400 (Bad Request), 404 (Not Found), 500 (Internal Server Error)
- Respuestas de error estructuradas con esquema Error que incluye código, mensaje y detalles

Organización y Estructura

```
Proyecto/
├─── football-api.yaml (450+ líneas)
├─── bruno-collection/ (13 archivos)
│    ├─── Teams/ (6 peticiones)
│    ├─── Players/ (5 peticiones)
│    └─── environments/ (2 ambientes)
└─── README.md (documentación completa)
```

Conclusiones

El proyecto demuestra comprensión profunda de los principios RESTful, incluyendo el uso apropiado de métodos HTTP, URLs semánticas basadas en recursos, comunicación stateless, y representaciones JSON. La arquitectura implementada sigue fielmente los 6 constraints de REST propuestos por Roy Fielding.

Se comprobó que una documentación completa (Swagger + Bruno + README) es fundamental para el éxito de una API. La especificación OpenAPI permite generar automáticamente clientes, servidores y documentación interactiva, mientras que Bruno facilita el testing y la colaboración en equipo.

La implementación de la relación 1:N mediante llaves foráneas y endpoints anidados demuestra que REST puede manejar eficientemente relaciones complejas entre entidades. El endpoint /teams/{id}/players es un patrón estándar de la industria para navegar asociaciones.

Las restricciones implementadas (longitud, rango, enumeraciones) son esenciales para mantener la integridad de datos y proporcionar feedback inmediato a los clientes de la API. El uso de esquemas reutilizables en OpenAPI reduce la duplicación y facilita el mantenimiento.

La distinción entre DTOs de entrada (TeamInput, PlayerInput) y respuesta (Team, Player con ID autogenerado) es una práctica profesional que mejora la seguridad y claridad de la API.

Prompt Generador

CONTEXTO
Eres un arquitecto de software senior especializado en diseño de APIs RESTful y documentación técnica. Tu tarea es generar documentación profesional completa para una API que implementa una relación 1 a muchos (one-to-many).

REQUISITOS DEL PROYECTO

1. ALCANCE FUNCIONAL
- Implementar relación 1:N entre dos entidades relacionadas
- Documentar operaciones CRUD completas (Create, Read, Update, Delete)
- Incluir endpoint de navegación de relación (/parent/{id}/children)
- Mínimo 10 endpoints totalmente funcionales
- Ejemplos con datos realistas y del dominio seleccionado

2. TECNOLOGÍAS Y ESTÁNDARES
- **Especificación:** OpenAPI 3.0.0 o superior
- **Cliente HTTP:** Bruno (alternativa moderna a Postman)
- **Formato de datos:** JSON
- **Versionamiento:** /v1 en la URL base
- **Códigos HTTP:** Usar apropiadamente 200, 201, 204, 400, 404, 500

3. ESTRUCTURA DE ENTIDADES

Entidad Padre (1)
```yaml
ParentEntity:
  type: object
  properties:
    id:
      type: integer
      description: ID autogenerado
      example: 1
    [campo_principal]:
      type: string
      minLength: 3
      maxLength: 100
      description: [Descripción del campo]
      example: "[Ejemplo realista]"
    [campos_adicionales]:
      # Incluir al menos 3-5 campos relevantes
      # Tipos variados: string, integer, date, enum
  required:
    - [campo_principal]
    - [otros_requeridos]
```