Logotipo, nombre de la empresa

Descripción generada automáticamente

**Documentación de API REST**

# BibliotecaApi

**Versión:** 1.0

**Fecha:** [21/09/2025]

**Autor:** [Dennis Brunaga]

**Estado:** [En desarrollo (listo para evaluación)]

# Tabla de Contenidos

1. [Información General](https://claude.ai/chat/46ab4705-92ba-4684-a20f-5abd2e114e15#informaci%C3%B3n-general)
2. [Configuración Inicial](https://claude.ai/chat/46ab4705-92ba-4684-a20f-5abd2e114e15#configuraci%C3%B3n-inicial)
3. [Autenticación](https://claude.ai/chat/46ab4705-92ba-4684-a20f-5abd2e114e15#autenticaci%C3%B3n)
4. [Endpoints](https://claude.ai/chat/46ab4705-92ba-4684-a20f-5abd2e114e15#endpoints)
5. [Modelos de Datos](https://claude.ai/chat/46ab4705-92ba-4684-a20f-5abd2e114e15#modelos-de-datos)
6. [Códigos de Error](https://claude.ai/chat/46ab4705-92ba-4684-a20f-5abd2e114e15#c%C3%B3digos-de-error)
7. Pruebas con Postman

# 1. Información General

## Descripción

BibliotecaApi es una API REST para gestionar una biblioteca universitaria. Permite **autenticación JWT** y CRUD de **Autores** y **Libros**, además de registrar **Préstamos**. Está pensada para aplicaciones académicas y de gestión de bibliotecas.

## URL Base

Base URL: https://localhost:7056

## Protocolo

* **HTTP/HTTPS:** HTTPS obligatorio
* **Formato de datos:** JSON
* **Codificación:** UTF-8

## Arquitectura REST

Esta API sigue los principios REST:

* Operaciones sin estado (stateless)
* Uso apropiado de métodos HTTP
* Recursos identificados por URIs
* Representaciones JSON

# 2. Configuración Inicial

## Requisitos Previos

* Clave API válida
* Acceso a internet
* Cliente HTTP (Postman, curl, etc.)

## Headers Obligatorios

http

Content-Type

:

application/json

Accept

:

application/json

Authorization

:

Bearer {token}

User-Agent

:

BibliotecaApi/1.0

## Entorno de Pruebas

https://localhost:7056/swagger/v1/swagger.json

## Entorno de Producción

https://api.biblioteca.com/v1

# 3. Autenticación

## Tipo de Autenticación JWT Bearer Token Obtener Token

**Endpoint:** POST /auth/login **Request:**

json

{

"email"

:

"admin@uni.edu"

,

"password"

:

"Admin123!"

}

**Response:**

json

{

"success"

:

true

,

"data"

:

{

"token"

:

"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

,

"expires\_in"

:

3600

,

"token\_type"

:

"Bearer"

}

}

## Uso del Token

http

Authorization

:

Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

## Renovar Token

**Endpoint:** POST /auth/refresh

# 4. Endpoints

## 4.1 Gestión de Libros

**4.1.1 Obtener todos los libros**

**Método:** GET **URL:** /libros

**Descripción:** Devuelve una lista paginada de libros.

**Parámetros de consulta:**

**Request:**

http

GET /libros?page=1&limit=20&categoria=novela

**Response exitoso (200):**

{

"success": true,

"data": {

"libros": [

{

"id": 1,

"titulo": "Cien años de soledad",

"autorId": 3,

"categoria": "Novela",

"disponible": true

}

],

"pagination": {

"current\_page": 1,

"per\_page": 20,

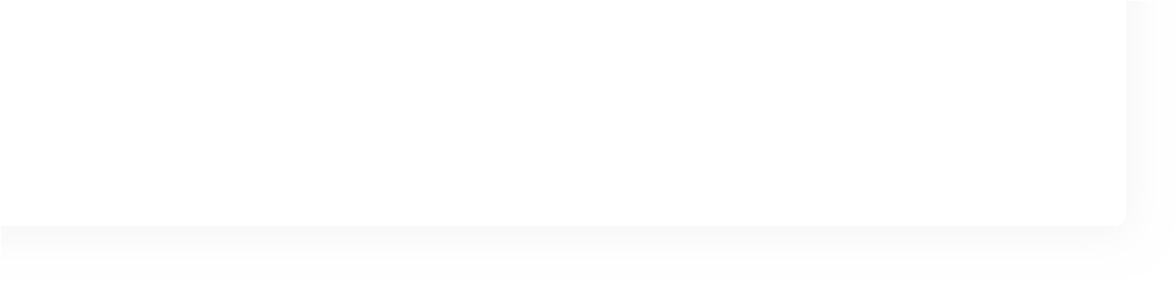
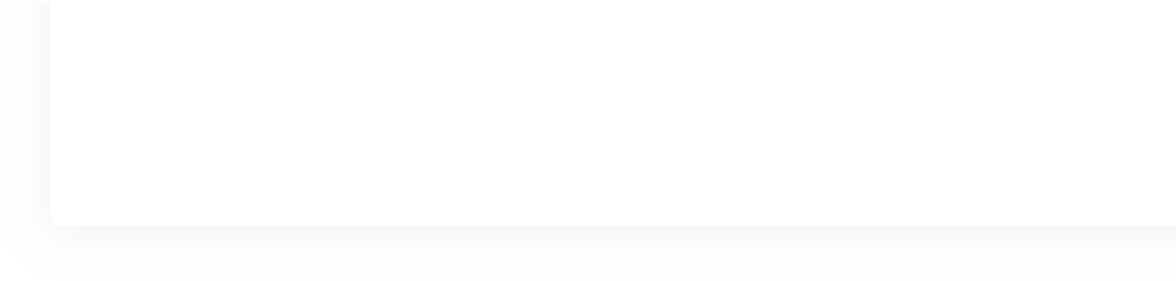
"total": 120,

"total\_pages": 6

}

}

}



**Parámetro**

**Tipo**

**Obligatorio**

**Descripción**

page

integer

No

Número de página (default: 1)

limit

integer

No

Elementos por página (default: 10, max: 100)

categoria

string

No

Filtrar por categoría

activo

boolean

No

Filtrar por disponibilidad

**4.1.2 Obtener producto por ID**

{

"success": true,

"data": {

"id": 1,

"titulo": "Cien años de soledad",

"autor": {

"id": 3,

"nombre": "Gabriel García Márquez"

},

"categoria": "Novela",

"disponible": true

}

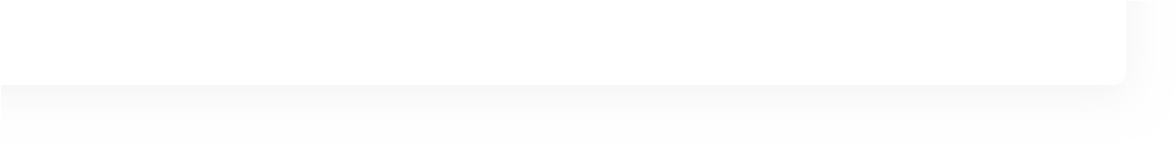
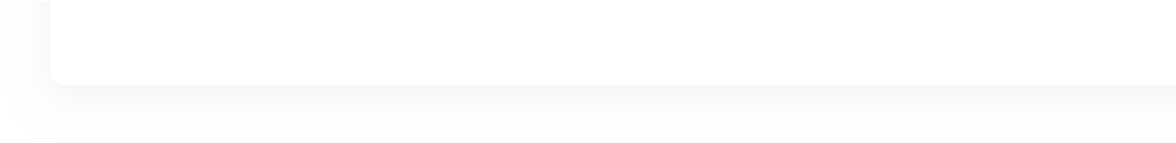
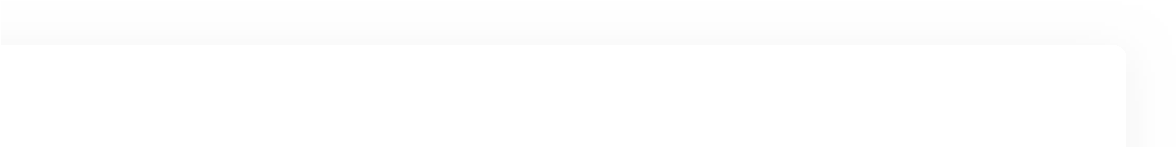
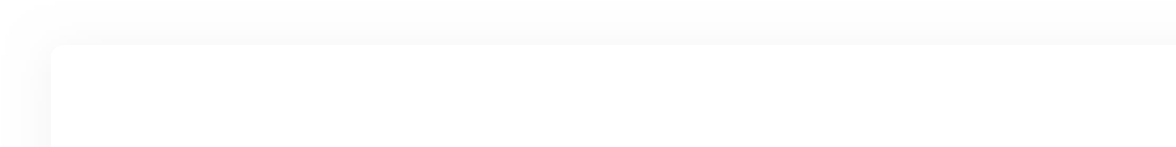
}

**Método:** GET

**URL:** /libros/{id}

**Descripción:** Devuelve la información de un libro específico.

**Parámetros de ruta:**



**Parámetro**

**Tipo**

**Descripción**

id

integer

ID único del libro

**Request:**

http

GET /libros/1

**Response exitoso (200):**

{

"success": true,

"data": {

"id": 1,

"titulo": "Cien años de soledad",

"autor": {

"id": 3,

"nombre": "Gabriel García Márquez"

},

"categoria": "Novela",

"disponible": true

}

}

**4.1.3 Crear nuevo producto**

**Método:** POST **URL:** /libros

**Descripción:** Crea un nuevo libro.

**Request Body:**

{

"titulo": "El Principito",

"autorId": 2,

"categoria": "Fábula",

"disponible": true

}

**Response exitoso (201):**

{

"success": true,

"message": "Libro creado exitosamente",

"data": {

"id": 15,

"titulo": "El Principito",

"autorId": 2,

"categoria": "Fábula",

"disponible": true

}

}

**4.1.4 Actualizar libro**

**Método:** PUT

**URL:** /libros/{id}

**Descripción:** Actualiza completamente un libro existente.

**Request Body:**

{

"titulo": "El Principito (Edición Anotada)",

"autorId": 2,

"categoria": "Fábula",

"disponible": true

}

**Response exitoso (200):**

{

"success": true,

"message": "Libro actualizado exitosamente",

"data": {

"id": 15,

"titulo": "El Principito (Edición Anotada)",

"autorId": 2,

"categoria": "Fábula",

"disponible": true

}

}

**4.1.5 Eliminar libro**

**Método:** DELETE

**URL:** /libros/{id}

**Descripción:** Elimina un libro (soft delete).

**Response exitoso (200):**

{

"success": true,

"message": "Libro eliminado exitosamente"

}

# 5. Modelos de Datos

## 5.1 Usuario

{

"id": "integer - ID único del usuario",

"nombre": "string - Nombre completo del usuario",

"email": "string - Correo electrónico del usuario",

"password": "string - Contraseña encriptada",

"rolId": "integer - Rol asignado (ej. Admin, Lector)"

}

## 5.2 Autor

{

"id": "integer - ID único del autor",

"nombre": "string - Nombre del autor",

"nacionalidad": "string - País de origen del autor"

}

**5.2 Autor**

{

"id": "integer - ID único del autor",

"nombre": "string - Nombre del autor",

"nacionalidad": "string - País de origen del autor"

}

**5.3 Libro**

{

"id": "integer - ID único del libro",

"titulo": "string - Título del libro",

"autorId": "integer - ID del autor",

"categoria": "string - Categoría o género literario",

"disponible": "boolean - Indica si el libro está disponible"

}

**5.4 Préstamo**

{

"id": "integer - ID único del préstamo",

"usuarioId": "integer - ID del usuario que realizó el préstamo",

"libroId": "integer - ID del libro prestado",

"fechaPrestamo": "datetime - Fecha en que se prestó el libro",

"fechaDevolucion": "datetime - Fecha de devolución pactada"

}

## 5.5 Error

{

"success": "boolean - Siempre false en caso de error",

"error": {

"code": "string - Código del error",

"message": "string - Descripción del error",

"details": "object - Detalles adicionales"

}

}

# 6. Códigos de Error HTTP Status Codes

#### 200 – OK

{

"success": true,

"data": { ... }

}

#### 201 – Created

{

"success": true,

"message": "Recurso creado exitosamente",

"data": { ... }

}

#### 400 – Bad Request (Error de validación)

{

"success": false,

"error": {

"code": "VALIDATION\_ERROR",

"message": "Los datos proporcionados no son válidos",

"details": {

"titulo": ["El campo título es obligatorio"],

"autorId": ["El autorId debe existir en la base de datos"]

}

}

}

#### 401 – Unauthorized (Sin autenticación)

{

"success": false,

"error": {

"code": "UNAUTHORIZED",

"message": "Se requiere autenticación para acceder a este recurso"

}

}

#### 404 – Not Found

{

"success": false,

"error": {

"code": "NOT\_FOUND",

"message": "El recurso solicitado no existe"

}

}

#### 500 – Internal Server Error

{

"success": false,

"error": {

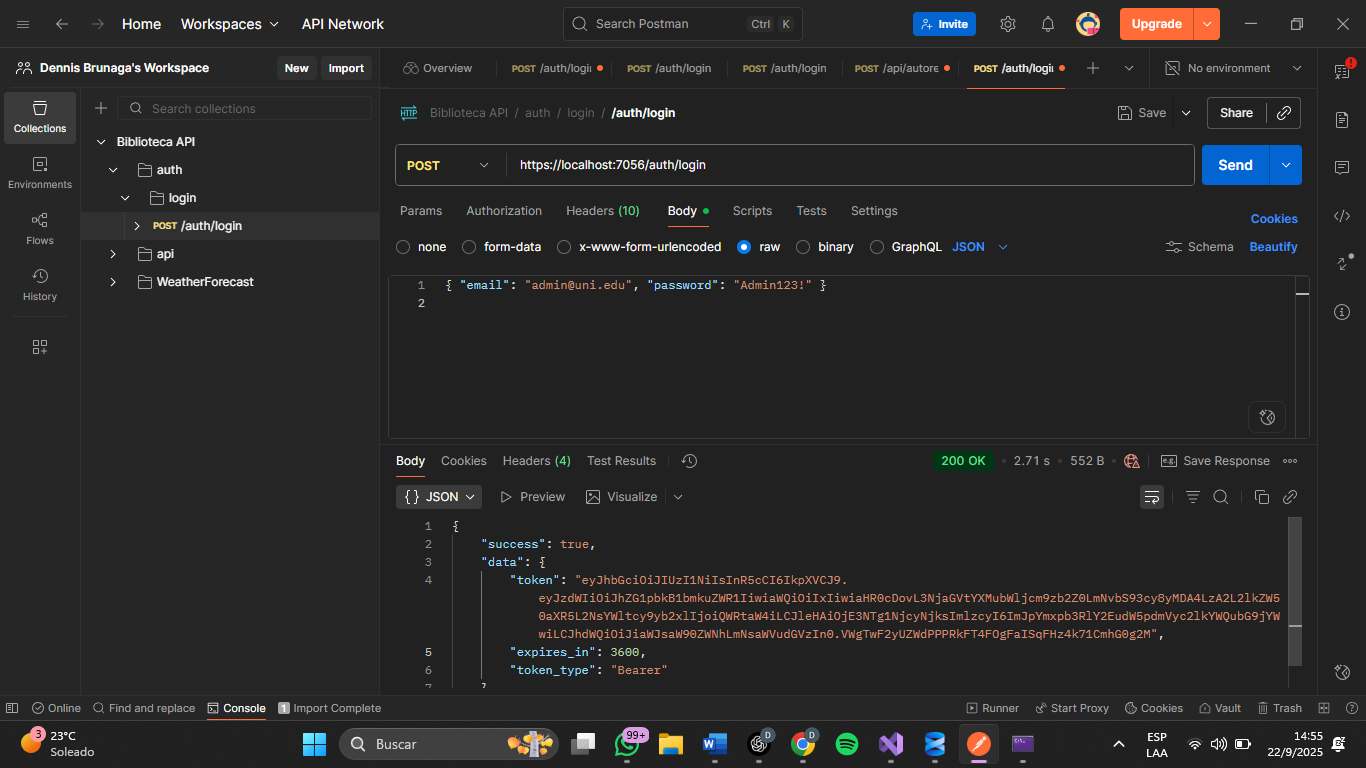
"code": "SERVER\_ERROR",

"message": "Ha ocurrido un error inesperado en el servidor"

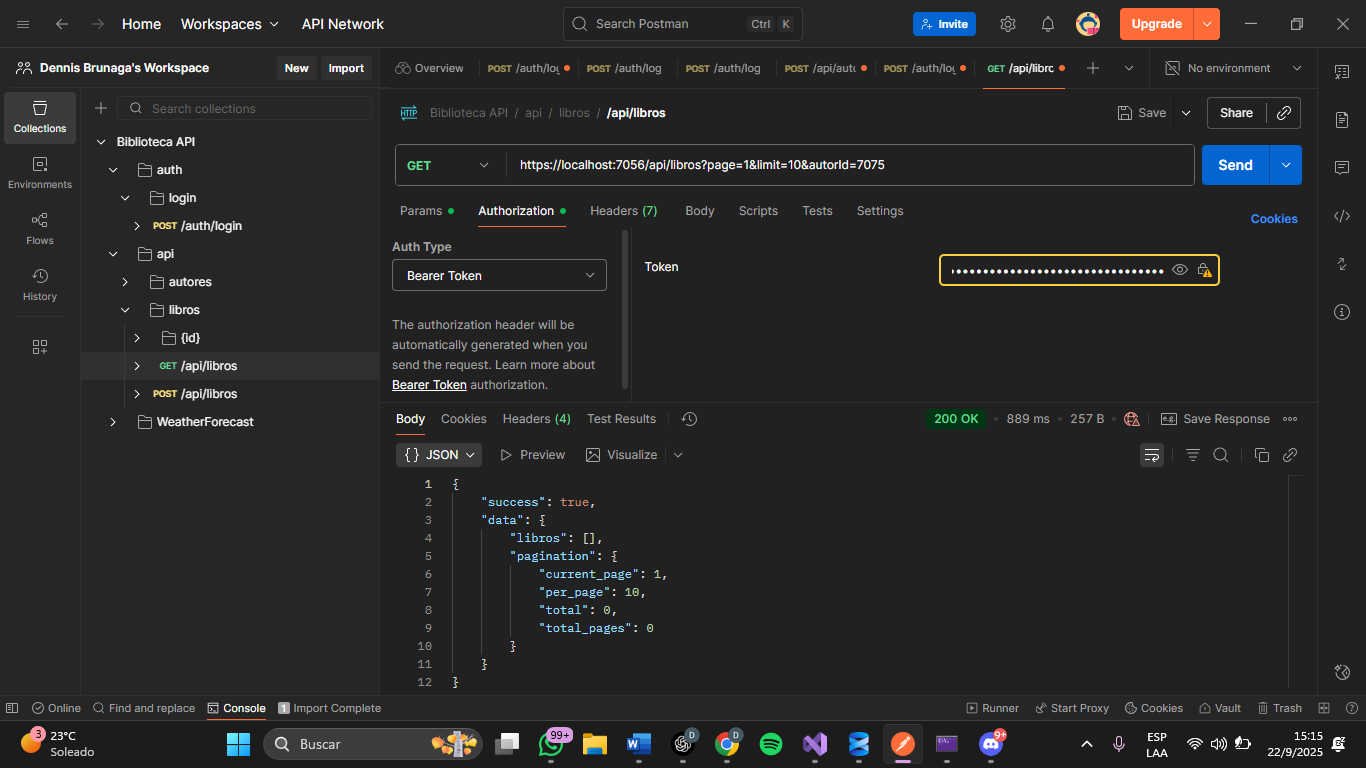
}

}

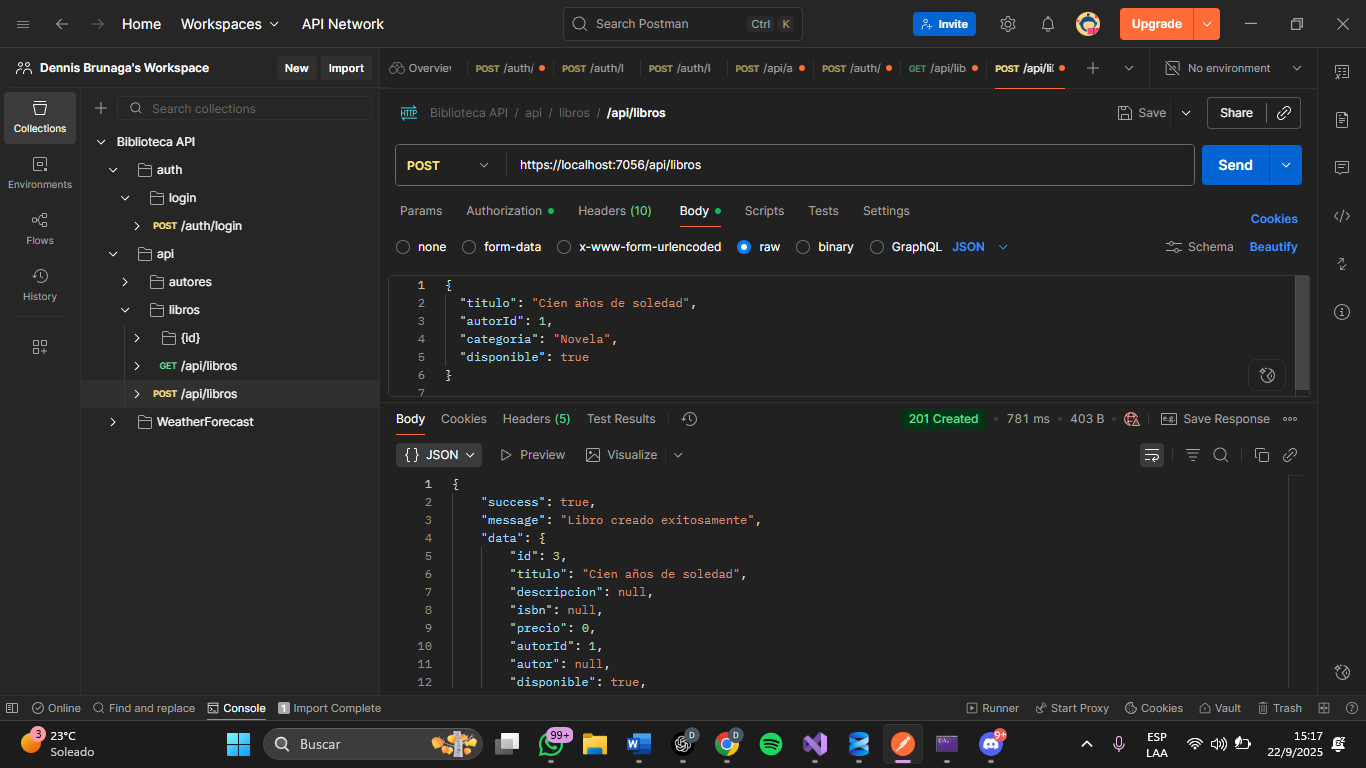
**PRUEBAS CON POSTMAN**



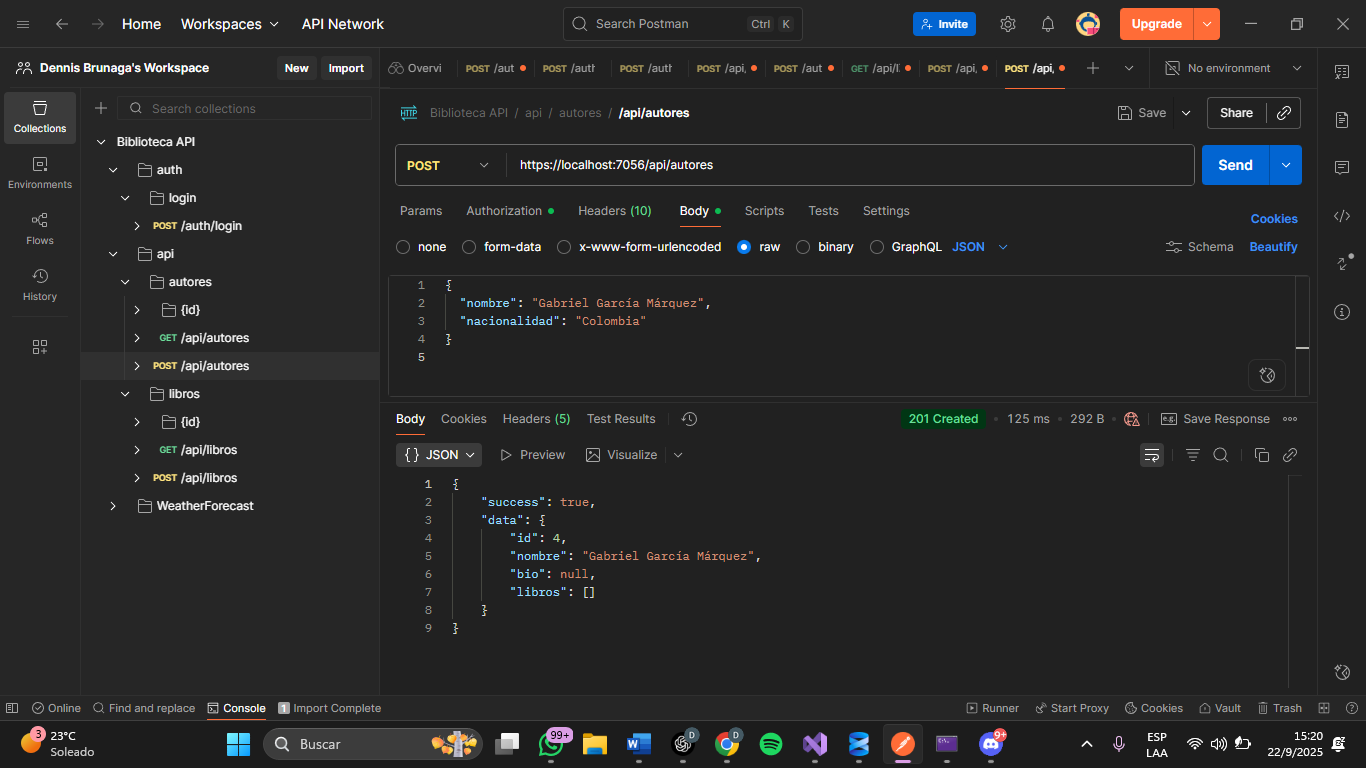
Se probó el endpoint **POST /auth/login** con credenciales válidas. La API respondió con un token JWT válido con duración de 1 hora.



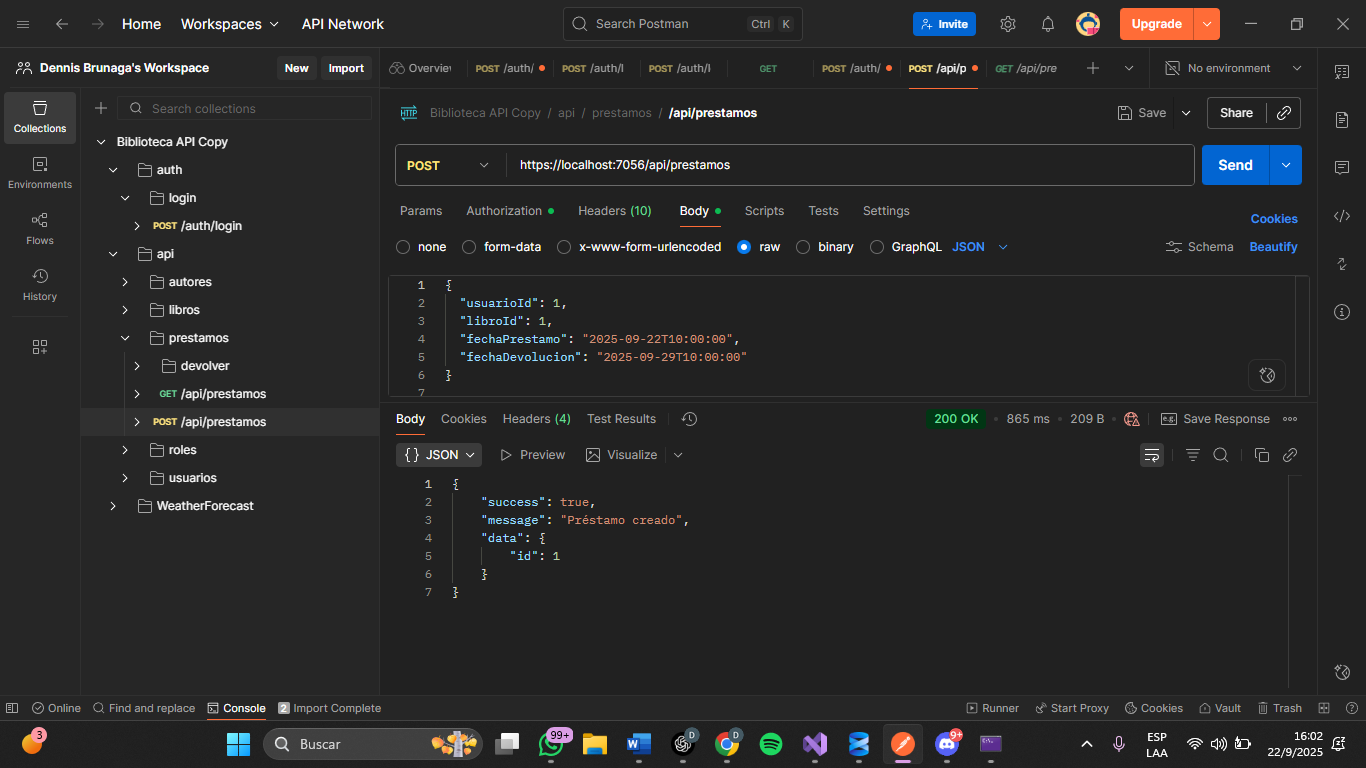
Se probó el endpoint **GET /libros** usando el token JWT obtenido previamente. La API devolvió la lista de libros registrados en la base de datos.



Se probó el endpoint **POST /libros** enviando título, autorId y categoría. La API respondió con un status 201 y devolvió el objeto del libro creado.



Se probó el endpoint **POST /autores** enviando nombre y nacionalidad. La API respondió con status 201 y devolvió el autor creado.



Se probó el endpoint **POST /prestamos** registrando un préstamo de un libro a un usuario. La API respondió con status 201 y devolvió el objeto del préstamo.

**© 2025 Brunaga Inteligence. Todos los derechos reservados.**