**LABORATORIO 7 - GTICS 2025-1**

**CRUD DE ESTUDIANTES CON SPRING BOOT Y SWAGGER**

**Nombre:** Edgar Diaz Zevillanos  
**Código:** 20211602  
**Curso:** Gestión de Tecnologías de Información y Comunicaciones  
**Ciclo:** 20251  
**Repositorio:** https://github.com/JaimitoSky/LAB7\_GTICS\_20251\_20211602

**1. Introducción**

Este documento describe la implementación de un servicio web RESTful desarrollado con Spring Boot, orientado a la gestión académica de estudiantes. Esta API puede formar parte de un proyecto más amplio sobre integraciones con aplicaciones móviles y sistemas académicos.

**2. Entidad Estudiante**

La entidad Estudiante cuenta con los siguientes campos, con sus respectivas validaciones:

| **Campo** | **Tipo** | **Validación / Restricción** |
| --- | --- | --- |
| nombres | String | Obligatorio, solo letras |
| apellidos | String | Obligatorio, solo letras |
| dni | String | Obligatorio, 8 dígitos únicos |
| codigoPucp | String | Obligatorio, 8 dígitos únicos |
| fechaNacimiento | LocalDate | Obligatorio, mayor a 16 años |
| sexo | String | 'M' o 'F' |
| correoInstitucional | String | Formato email, dominio @pucp.edu.pe |
| correoPersonal | String | Formato email, distinto al institucional |
| telefono | String | 9 dígitos, empieza con '9' |
| direccion | String | Máximo 100 caracteres |
| departamento | String | Solo letras |
| provincia | String | Solo letras |
| carrera | String | Solo letras |
| fechaRegistro | LocalDateTime | Se genera automáticamente |
| ultimaActualizacion | LocalDateTime | Se actualiza automáticamente al editar |
| estado | Boolean | true: Activo, false: Inactivo |

**3. Endpoints Implementados**

**3.1. Obtener listado de estudiantes**

* **Método:** GET
* **Ruta:** /api/estudiantes
* **Descripción:** Devuelve todos los estudiantes registrados.
* **Notas:**
  + Excluye fechaRegistro y ultimaActualizacion
  + Traduce estado: True a "Activo", False a "Inactivo"
* **Captura:** (Insertar imagen de respuesta en Swagger con listado)

**3.2. Obtener estudiante por DNI**

* **Método:** GET
* **Ruta:** /api/estudiantes/{dni}
* **Descripción:** Retorna un estudiante según su DNI.
* **Errores comunes:**
  + 404 si el estudiante no existe
* **Captura:** (Insertar imagen de consulta con DNI válido y no válido)

**3.3. Registrar estudiante**

* **Método:** POST
* **Ruta:** /api/estudiantes
* **Descripción:** Crea un nuevo estudiante.
* **Validaciones:**
  + DNI y Código PUCP únicos
  + Campos como correos, teléfono y fecha correctamente validados
  + estado se asigna automáticamente como true
  + fechaRegistro se genera automáticamente
* **Ejemplo Request JSON:**

{

"id": 1,

"nombres": "Edgar",

"apellidos": "Diaz Zevillanos",

"dni": "72897779",

"codigoPucp": "20211602",

"fechaNacimiento": "2004-07-07",

"sexo": "M",

"correoInstitucional": "a20211602@pucp.edu.pe",

"correoPersonal": "edgar.diaz@gmail.com",

"telefono": "933969389",

"direccion": "Av. Universitaria 123",

"departamento": "Lima",

"provincia": "Lima",

"carrera": "Ingeniería de las Telecomunicaciones",}

* **Captura:** (Insertar imagen de POST exitoso y error por validación)

**3.4. Editar estudiante**

* **Método:** PUT
* **Ruta:** /api/estudiantes/{id}
* **Descripción:** Actualiza los campos enviados del estudiante.
* **Notas:**
  + ultimaActualizacion se actualiza automáticamente
  + No se permite modificar fechaRegistro
* **Captura:** (Insertar imagen con ejemplo de actualización)

**3.5. Eliminar estudiante (borrado lógico)**

* **Método:** DELETE
* **Ruta:** /api/estudiantes/{id}
* **Descripción:** Marca como inactivo (estado = false) al estudiante dado.
* **Captura:** (Insertar imagen del DELETE con respuesta exitosa)

**4. Manejo de Errores y Validaciones**

Se utiliza un GlobalExceptionHandler que gestiona:

* Errores de validación (@Valid)
* Excepciones personalizadas (estudiante no encontrado, duplicado, etc.)
* Respuestas en formato:

{

"mensaje": "Error de validación",

"errores": ["dni: debe tener 8 dígitos"]

}

* **Captura:** (Insertar imagen de error personalizado)

**5. Estado de Proyecto y Buenas Prácticas**

* Swagger UI disponible en /swagger-ui/index.html
* Validación exhaustiva con anotaciones y lógica personalizada
* Excepciones claras y controladas
* Mensajes con cuerpo estandarizado
* Uso correcto de códigos HTTP (200, 201, 400, 404, etc.)
* Repositorio con estructura clara y uso de ramas main y develop

**6. Script de Base de Datos**

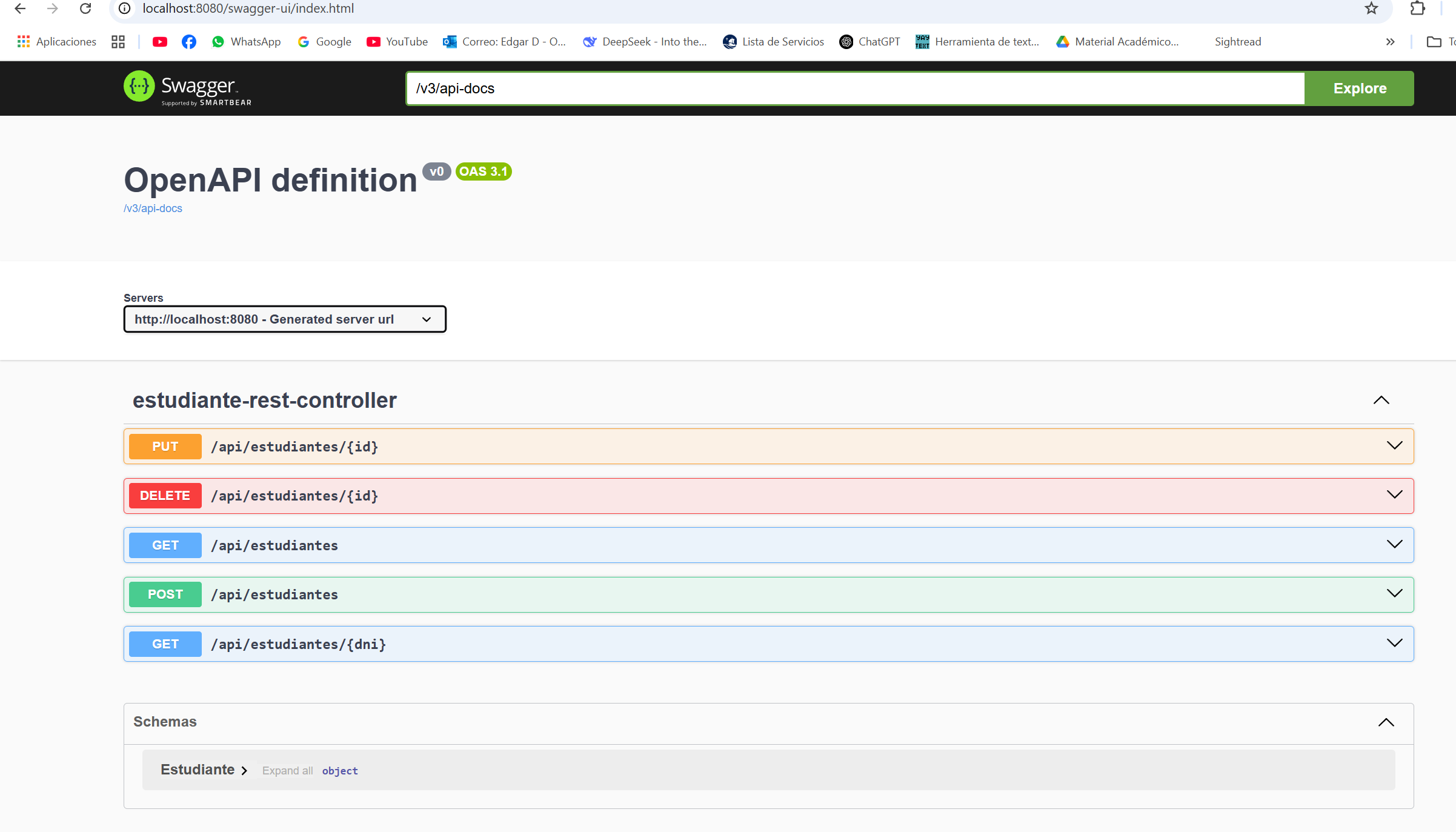
Este script está incluido en el repositorio GitHub con el nombre “lab7\_db.sql”.

**7. Conclusión**

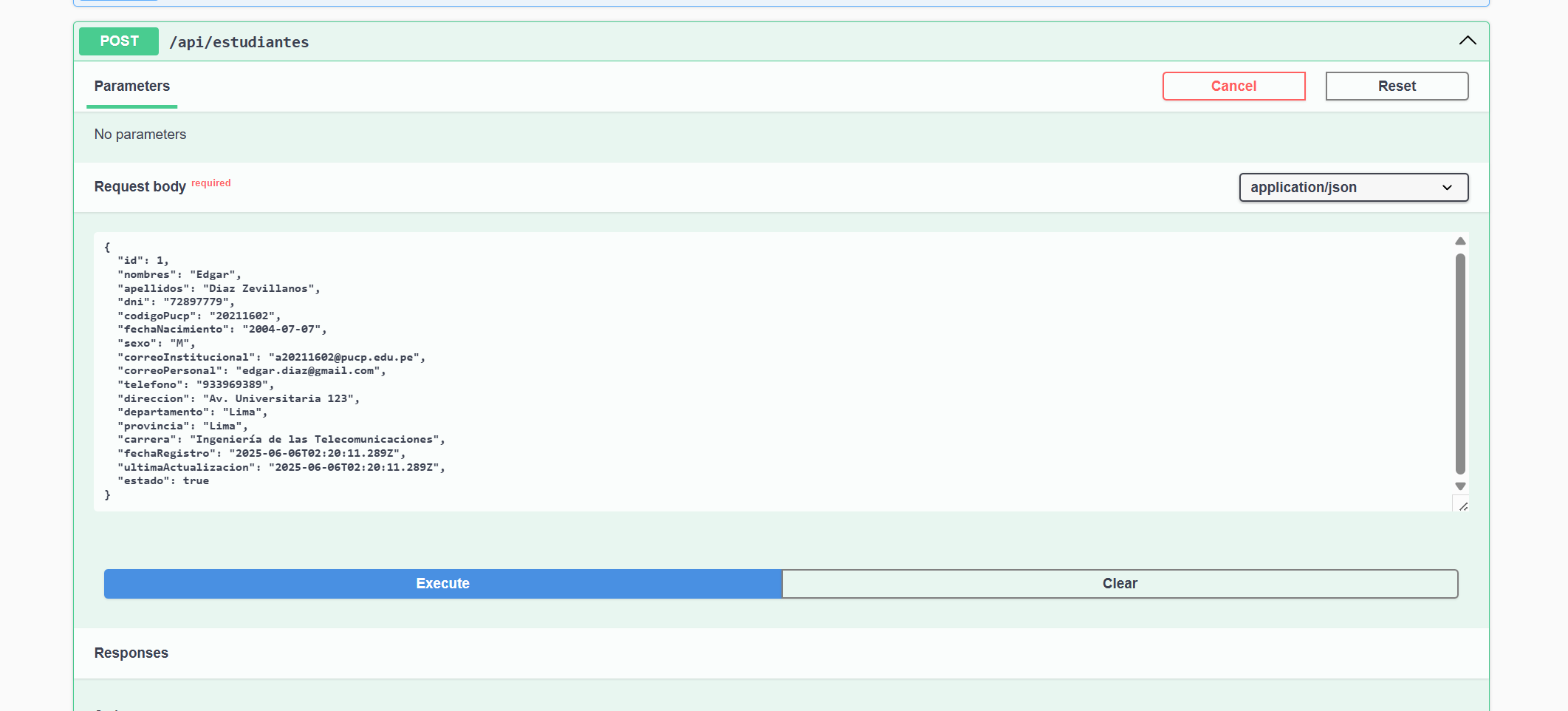
Se ha implementado correctamente un servicio web RESTful con todas las funcionalidades requeridas, asegurando calidad en validación, control de errores y documentación. El proyecto cumple con las exigencias del laboratorio 7 del curso GTICS.

**8. Anexos**

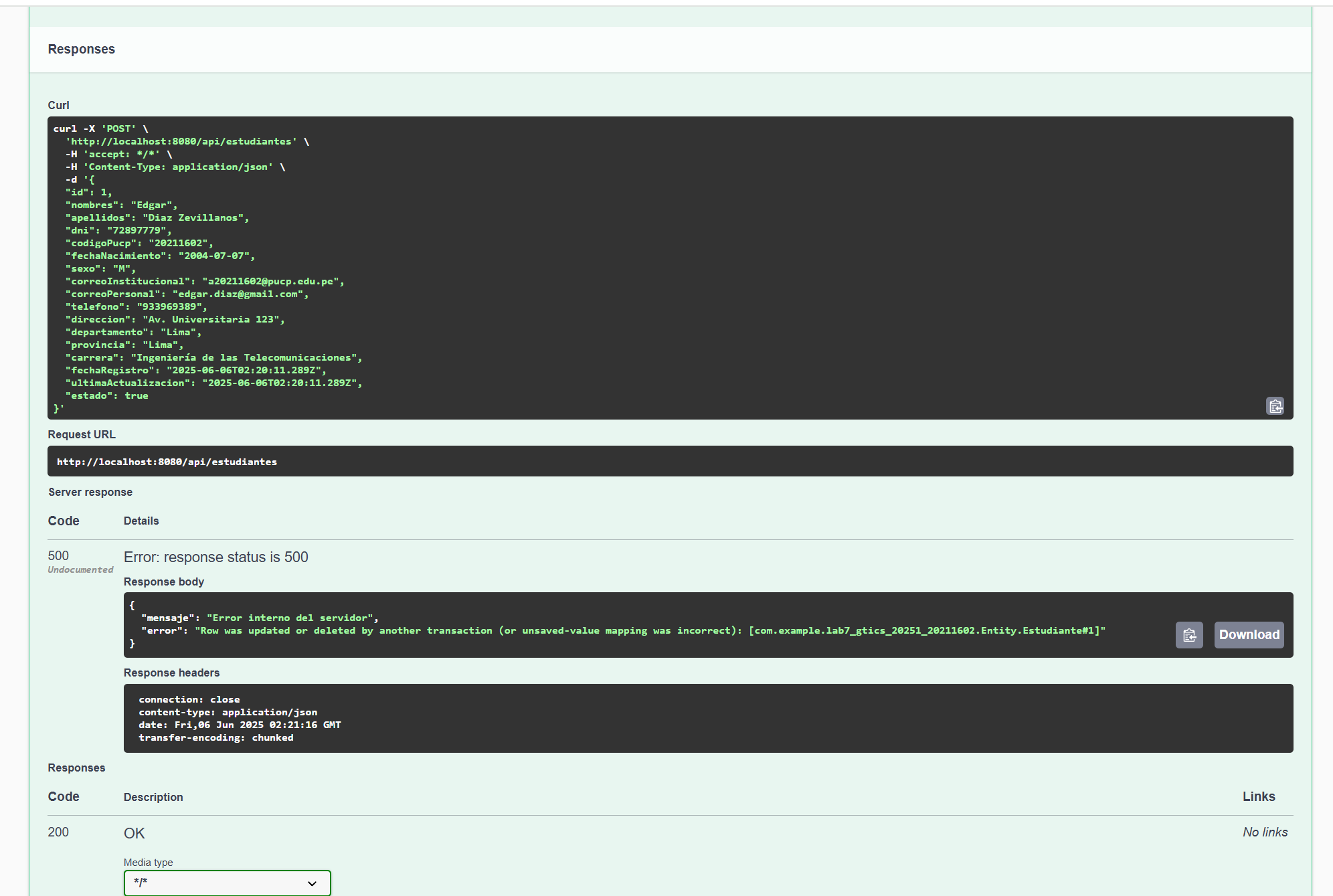
* Capturas completas de Swagger



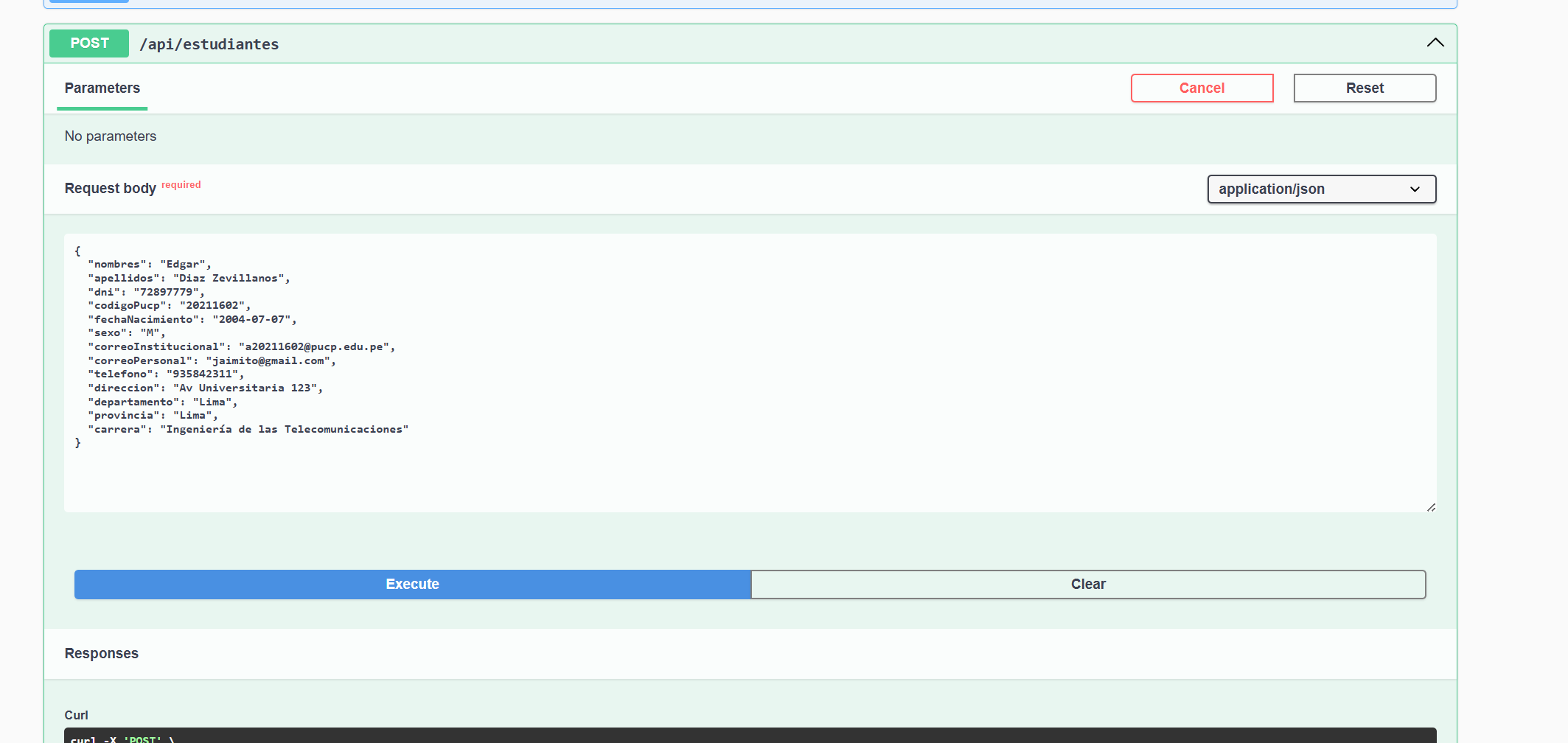
Vista general de Swagger UI

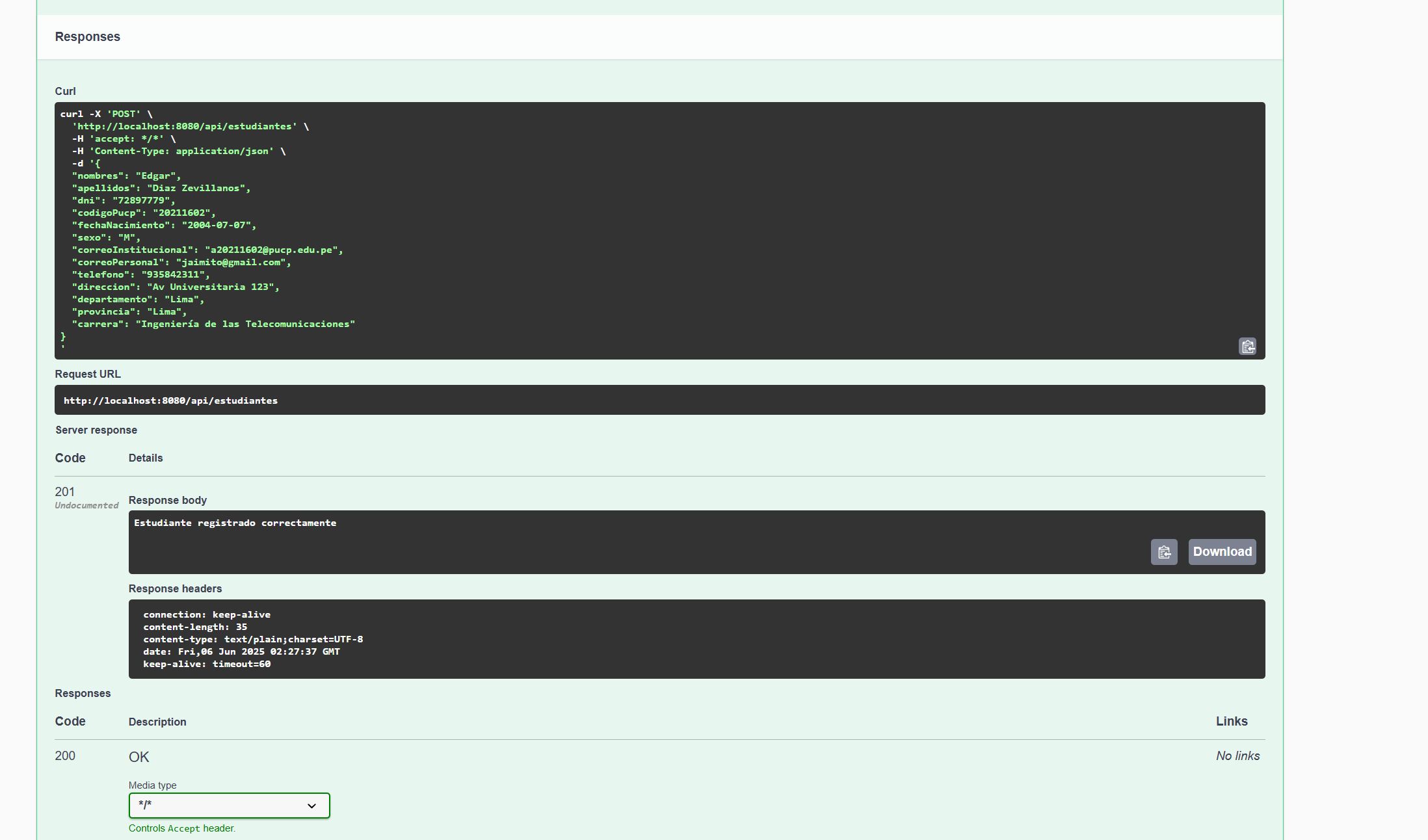


Intento de creación de estudiante (POST)

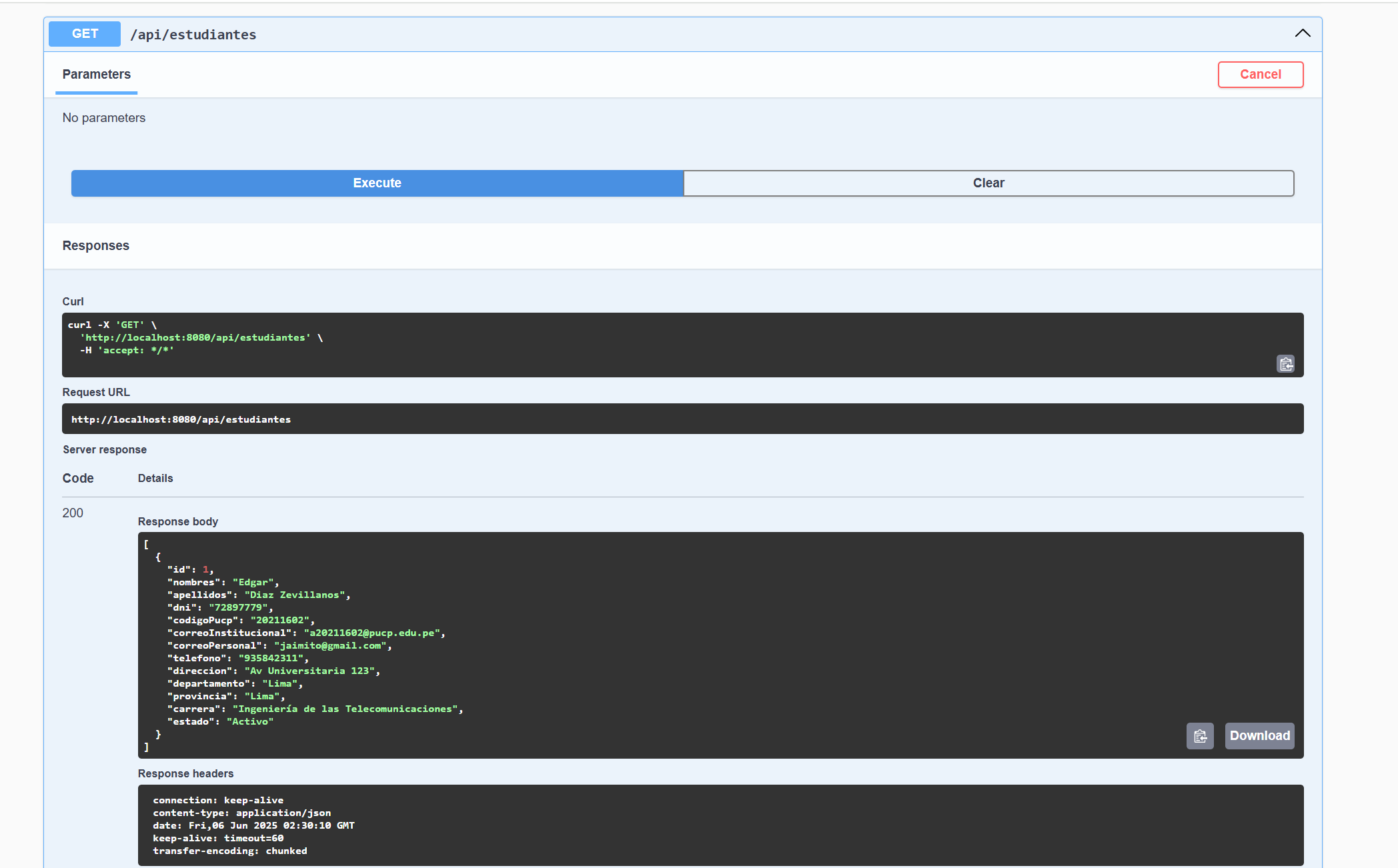


Error al crear estudiante (status 500), una de las varias excepciones

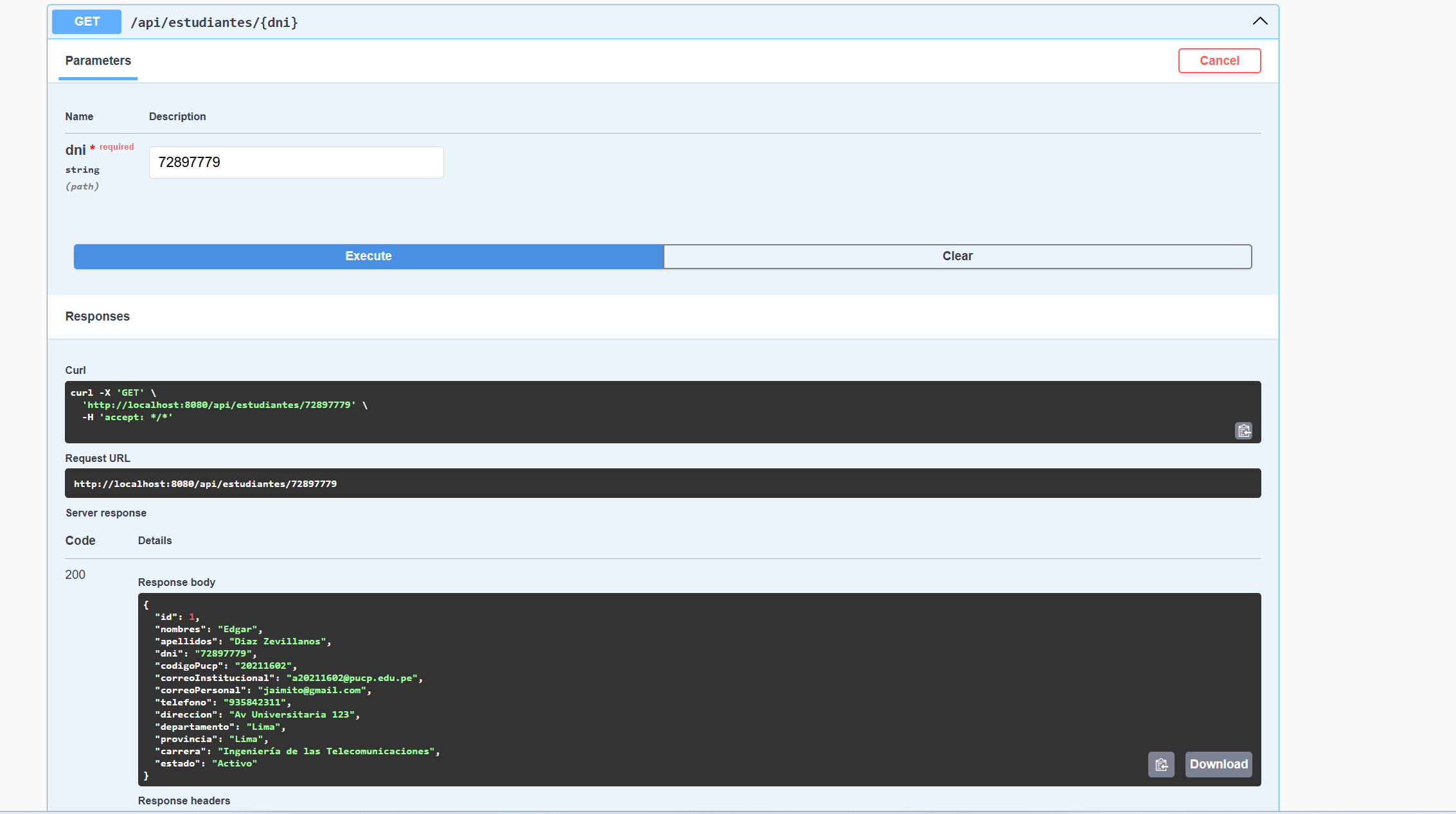




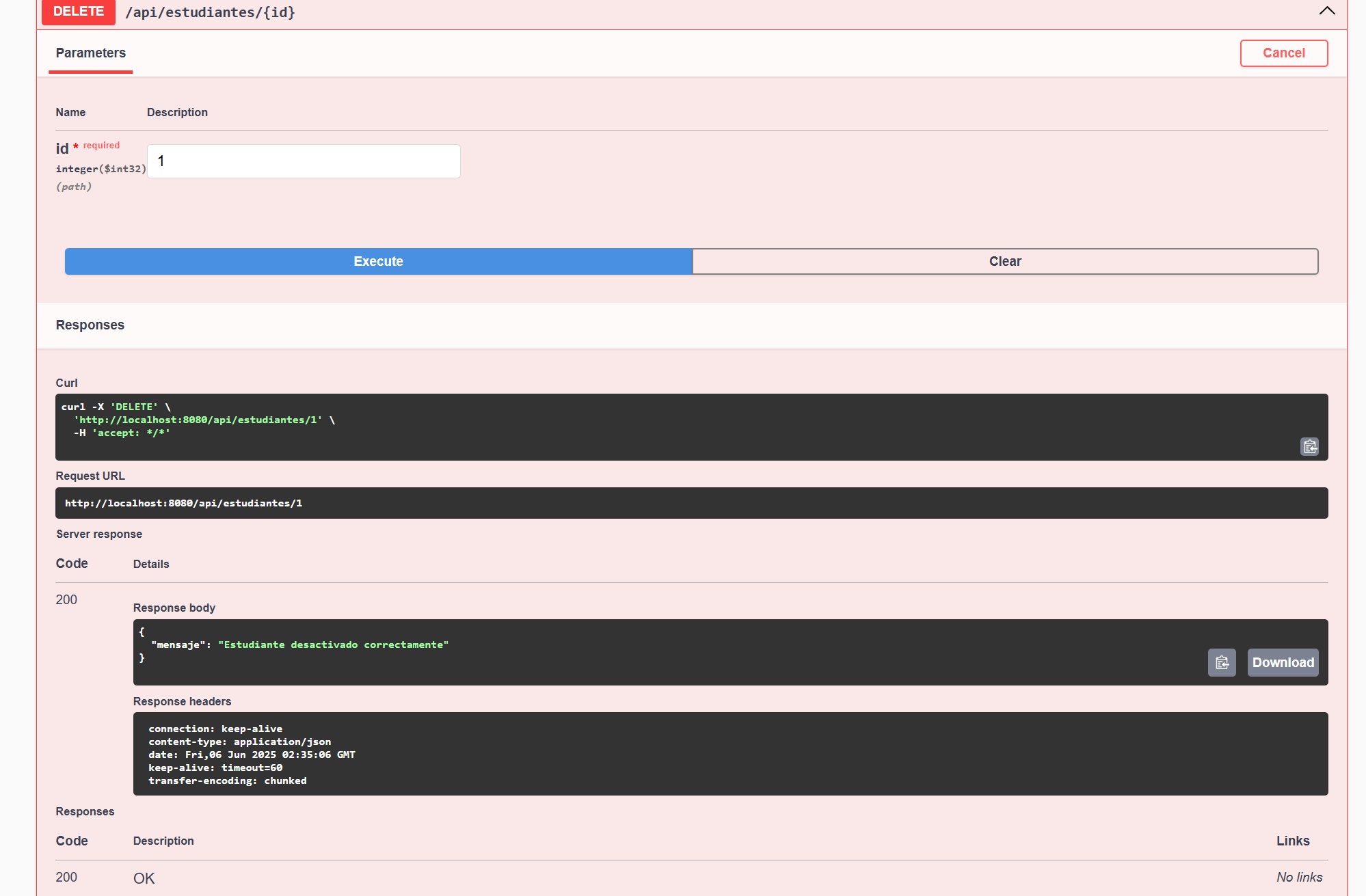
Estudiante creado con éxito



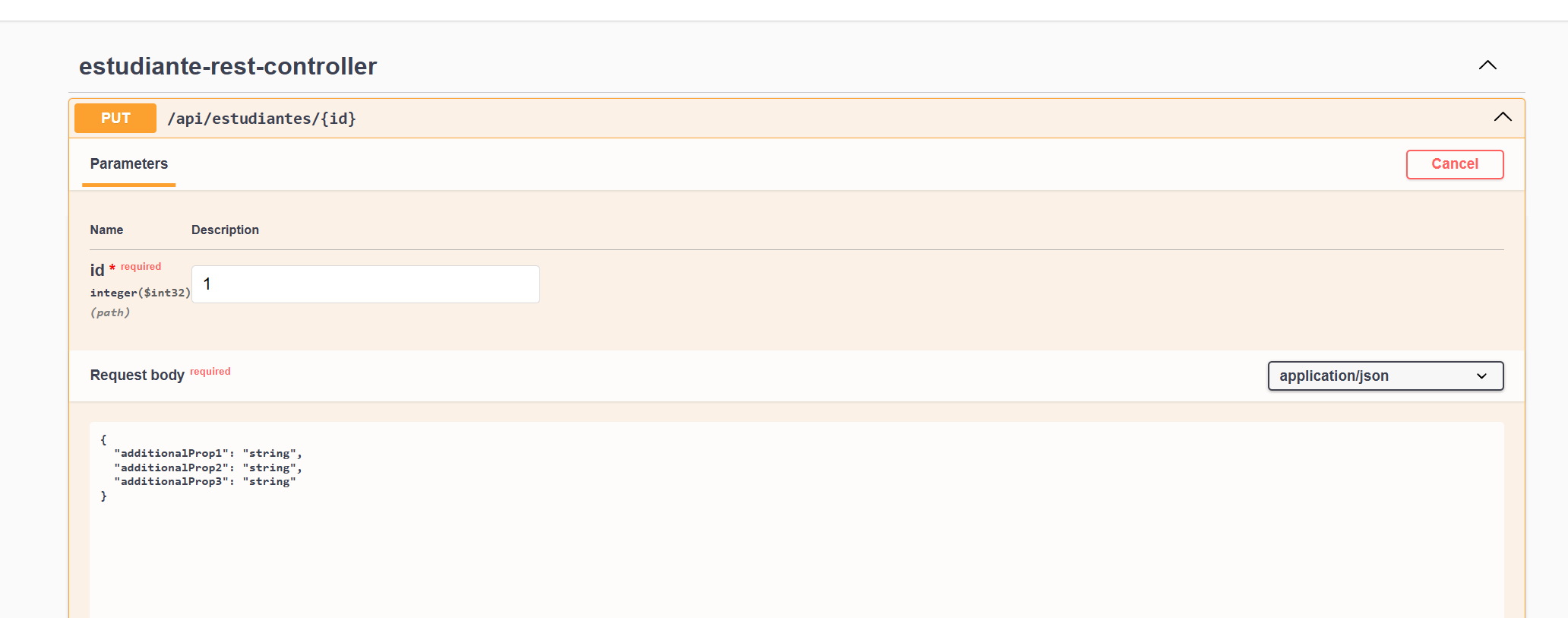
Llamado de la lista (GET) de estudiantes exitoso (solo aparece el estudiante recién creado)

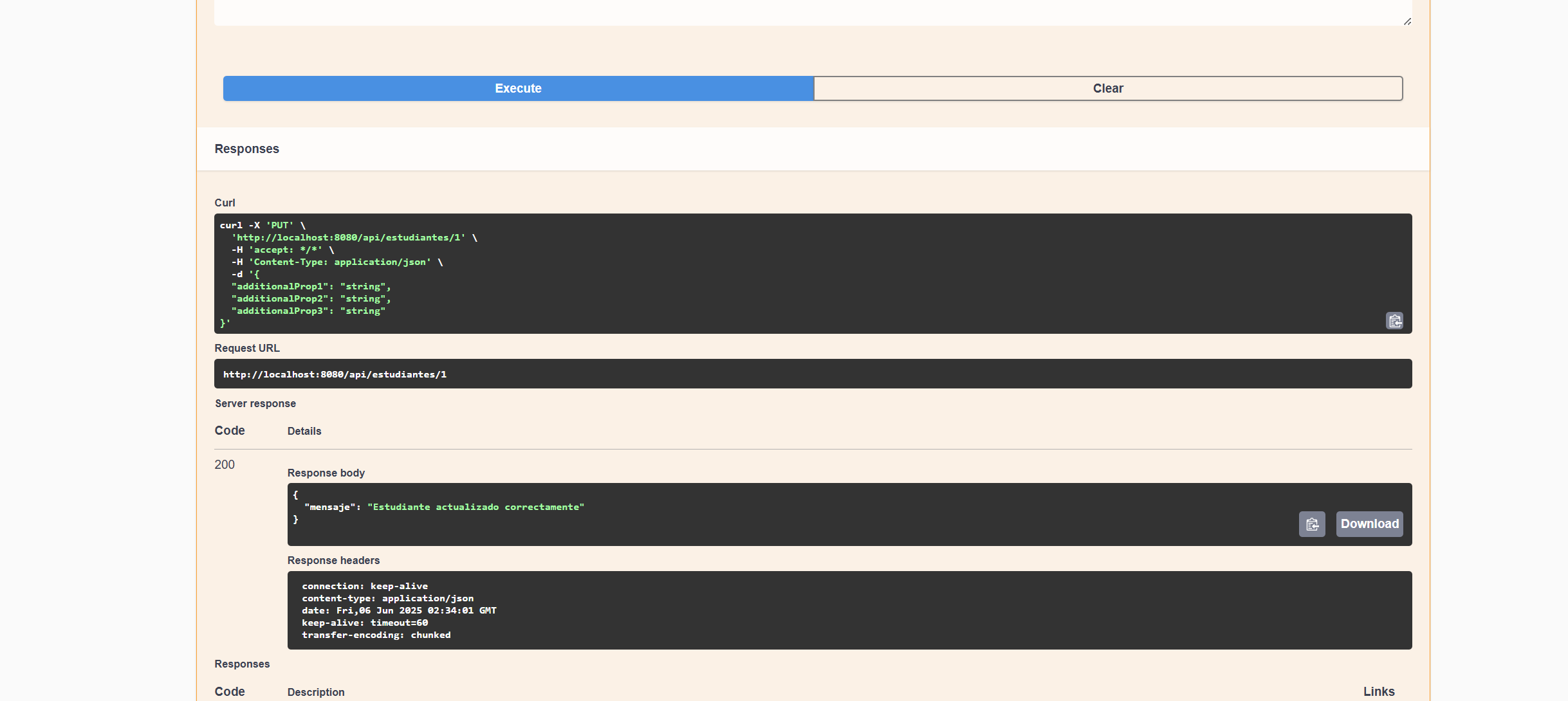


Llamado de estudiante recién creado exitoso con DNI (GET)



Desactivación de estudiante con ID (DELETE)





Activación de estudiante con ID (PUT)

* Pull request develop a main en GitHub

