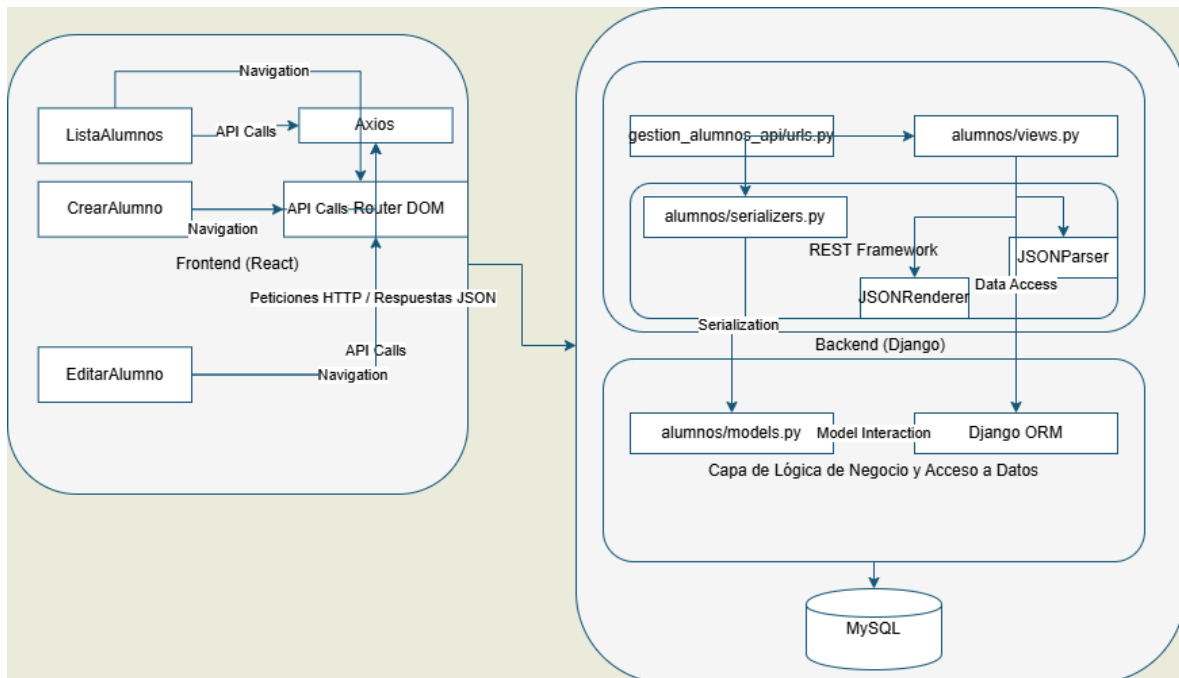


## Tutorial de CRUD con api rest full con Django:





### Diccionario De términos:

Método	Acción	Idempotente*	Seguro**
GET	Leer	Sí	Sí
POST	Crear	No	No
PUT	Reemplazar completo	Sí	No
PATCH	Modificar parcialmente	No	No
DELETE	Eliminar	Sí	No
HEAD	Obtener cabeceras	Sí	Sí
OPTIONS	Consultar opciones	Sí	Sí

### Backend (Django):

- **Django:** Un framework web de alto nivel basado en Python que fomenta el desarrollo rápido y un diseño limpio y pragmático. Lo usamos para construir la API REST.
- **Django REST Framework (DRF):** Un toolkit potente y flexible para construir Web APIs en Django. Lo utilizamos para:

- o **Serializers:** Componentes que convierten objetos de modelo de Django en formatos de datos como JSON (para la API) y viceversa.  
(`alumnos/serializers.py`)
- o **Views (ViewSet):** Clases que definen la lógica para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en nuestros recursos (como los alumnos).  
(`alumnos/views.py`)
- o **Router:** Componentes que generan automáticamente las URLs de la API basadas en nuestros ViewSets. (`gestion_alumnos_api/urls.py`)
- o **Renderers:** Componentes que determinan el formato en el que se presenta la respuesta de la API (por ejemplo, `JSONRenderer`).  
(`gestion_alumnos_api/settings.py`)
- o **Parsers:** Componentes que determinan cómo se procesan los datos de las peticiones entrantes (por ejemplo, `JSONParser`).  
(`gestion_alumnos_api/settings.py`)
- **Models:** Definiciones de la estructura de datos en la base de datos (nuestra tabla `alumnos`). (`alumnos/models.py`)
- **Migrations:** El sistema de Django para propagar los cambios en tus modelos de datos (creación, modificación de campos, etc.) a tu esquema de base de datos.  
(`alumnos/migrations/`)
- **manage.py:** Un script de utilidad para administrar proyectos Django (ejecutar el servidor de desarrollo, crear migraciones, aplicarlas, etc.).
- **settings.py:** El archivo de configuración principal del proyecto Django, donde se definen ajustes como la conexión a la base de datos, las aplicaciones instaladas, el middleware, etc. (`gestion_alumnos_api/settings.py`)
- **urls.py:** Archivos que definen las rutas de la API y las asocian a las vistas correspondientes. (`gestion_alumnos_api/urls.py`, `alumnos/urls.py` - aunque en nuestro caso usamos el router en el `urls.py` principal).
- **MySQL:** El sistema de gestión de bases de datos relacional que utilizamos para almacenar los datos de los alumnos.
- **mysqlclient:** El conector de Python necesario para interactuar con bases de datos MySQL desde Django.
- **django-cors-headers:** Una librería de Django para manejar las cabeceras CORS (Cross-Origin Resource Sharing), permitiendo que nuestro frontend React en un dominio diferente (puerto) pueda hacer peticiones a la API de Django.
- **Middleware:** Componentes que se ejecutan durante el procesamiento de una petición/respuesta en Django. Usamos el middleware de CORS.

### Frontend (React):

- **React:** Una biblioteca de JavaScript para construir interfaces de usuario dinámicas e interactivas. La utilizamos para crear el frontend de nuestra aplicación de gestión de alumnos.
- **JSX (JavaScript XML):** Una extensión de sintaxis para JavaScript que permite escribir código que se asemeja a HTML y que luego se transforma en llamadas a funciones de React.

- **Componentes:** Bloques de construcción reutilizables de la interfaz de usuario de React (ejemplos: `ListaAlumnos`, `CrearAlumno`, `EditarAlumno`). Pueden ser funcionales (con hooks) o de clase.
- **Estado (`useState`):** Un hook de React que permite a los componentes funcionales tener variables de estado local y controlar cómo se renderiza la interfaz de usuario con el tiempo.
- **Efectos Secundarios (`useEffect`):** Un hook de React que permite realizar efectos secundarios en componentes funcionales (por ejemplo, llamadas a la API).
- **axios:** Una librería de cliente HTTP basada en promesas para realizar peticiones al backend de Django.
- **react-router-dom:** Una librería para la navegación en aplicaciones React, permitiendo crear rutas para diferentes partes de la aplicación (lista, crear, editar).
  - **BrowserRouter (o Router):** Un componente que habilita el enrutamiento en la aplicación.
  - **Routes:** Un componente que renderiza el primer `Route` que coincide con la ubicación actual.
  - **Route:** Un componente que define una ruta específica y el componente que se debe renderizar cuando esa ruta coincide.
  - **Link:** Un componente para crear enlaces navegables dentro de la aplicación sin recargar la página.
  - **useNavigate:** Un hook para navegar programáticamente a diferentes rutas.
  - **useParams:** Un hook para acceder a los parámetros dinámicos de una ruta (como el `id` en la ruta de edición).
- **react-bootstrap:** Una biblioteca de componentes de interfaz de usuario de React que implementa el diseño de Bootstrap, proporcionando componentes estilizados y reutilizables como `Table`, `Button`, `Alert`, `Form`, `Form.Group`, `Form.Label`, `Form.Control`, `Container`, `Nav`, `Navbar`.
- **react-icons:** Una librería que proporciona iconos populares como Font Awesome, etc., como los iconos de edición (`FaEdit`) y eliminación (`FaTrash`).
- **npm o yarn:** Gestores de paquetes utilizados para instalar y administrar las dependencias del proyecto React (como `axios`, `react-router-dom`, `react-bootstrap`, `react-icons`).

Este diccionario proporciona una visión general de los principales componentes y tecnologías que hemos estado utilizando para construir nuestra aplicación de gestión de alumnos con un backend Django API y un frontend React. ¡Espero que te sea útil!

Proyecto ApiREst django con react js.

Crear la base de datos:

-- Creación de la base de datos gestion\_alumnos

CREATE DATABASE IF NOT EXISTS gestion\_alumnos;

-- Seleccionar la base de datos para trabajar con ella

USE gestion\_alumnos;

-- Creación de la tabla alumnos

```
CREATE TABLE IF NOT EXISTS alumnos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    numero_documento VARCHAR(20) UNIQUE NOT NULL,  
    nota1 DECIMAL(5, 2) NOT NULL,  
    nota2 DECIMAL(5, 2) NOT NULL,  
    nota3 DECIMAL(5, 2) NOT NULL  
);
```

-- (Opcional) Insertar algunos datos de prueba

```
INSERT INTO alumnos (nombre, numero_documento, nota1, nota2, nota3)  
VALUES  
  
('Juan Pérez', '123456789', 4.5, 3.8, 4.2),  
  
('María Gómez', '987654321', 3.9, 4.7, 4.0),  
  
('Carlos López', '112233445', 3.0, 3.5, 3.2);
```

-- (Opcional) Crear un usuario específico para la aplicación Django (por seguridad)

-- Reemplaza 'tu\_usuario\_django' y 'tu\_contraseña' con valores seguros

```
-- CREATE USER 'tu_usuario_django'@'localhost' IDENTIFIED BY  
'tu_contraseña';
```

```
-- GRANT ALL PRIVILEGES ON gestion_alumnos.* TO  
'tu_usuario_django'@'localhost';
```

-- FLUSH PRIVILEGES;

## Distribución de carpetas

gestion\_alumnos\_api/ (Carpeta raíz del proyecto Django)

├── gestion\_alumnos\_api/ (Carpeta del proyecto Django - contiene archivos de configuración)

| ├── \_\_init\_\_.py

| ├── asgi.py

| ├── settings.py (Configuración de Django, incluyendo la base de datos)

| ├── urls.py (Configuración de las URLs del proyecto)

| ├── wsgi.py

| └── \_\_pycache\_\_/

├── alumnos/ (Aplicación Django para la gestión de alumnos)

| ├── \_\_init\_\_.py

| ├── admin.py

| ├── apps.py

| ├── migrations/ (Contiene los archivos de migración de la base de datos)

| | └── \_\_init\_\_.py

| ├── models.py (Definición del modelo de datos Alumno)

| ├── serializers.py (Definición de los serializadores para los modelos)

| ├── tests.py

| ├── views.py (Definición de las vistas de la API - ViewSets)

| └── \_\_pycache\_\_/

├── venv/ (Entorno virtual de Python - contiene las dependencias)

├── manage.py (Script para ejecutar comandos de Django)

└── gestion\_alumnos\_frontend/ (Carpeta raíz del proyecto React)

```
├── node_modules/    (Contiene las dependencias de Node.js)
├── public/          (Archivos estáticos como index.html)
│   ├── index.html
│   └── ...
├── src/             (Código fuente de la aplicación React)
│   ├── App.js       (Componente principal de la aplicación)
│   ├── index.js     (Punto de entrada de la aplicación)
│   ├── components/  (Carpeta para los componentes React)
│   │   ├── ListaAlumnos.js
│   │   ├── CrearAlumno.js
│   │   ├── EditarAlumno.js
│   │   └── ...
│   ├── App.css      (Estilos para el componente App)
│   ├── index.css    (Estilos globales)
│   ├── assets/      (Carpeta para imágenes, etc. - opcional)
│   └── ...
├── .gitignore
├── package.json     (Información del proyecto y dependencias de npm)
├── package-lock.json (Información detallada de las dependencias de npm)
└── README.md
```

## Backend (Django REST Framework - DRF)

### Paso 1: Configuración del Entorno Django

1. **Instala Python y pip:** Asegúrate de tener Python y pip instalados en tu sistema.
2. **Crea un entorno virtual:**

```
python -m venv venv
```

activa maquina

venv\Scripts\activate # En Windows

```
PS C:\Users\LILLYU\Documents\GitHub\ReactJs\crudApirestDjango> python -m venv venv
PS C:\Users\LILLYU\Documents\GitHub\ReactJs\crudApirestDjango> venv\Scripts\activate
(venv) PS C:\Users\LILLYU\Documents\GitHub\ReactJs\crudApirestDjango>
```

Instala Django y Django REST Framework:

pip install django django-rest-framework mysqlclient

pip install django-cors-headers

**Crea un nuevo proyecto Django:**

django-admin startproject gestion\_alumnos\_api

cd gestion\_alumnos\_api

**Crea una nueva aplicación Django:**

python manage.py startapp alumnos

**Paso 2: Configuración de la Base de Datos MySQL**

Configura la conexión a la base de datos en gestion\_alumnos\_api/settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'gestion_alumnos',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

1. Reemplaza 'tu\_usuario\_mysql' y 'tu\_contraseña\_mysql' con tus credenciales de MySQL.
2. Ojo también agrega la aplicación **alumnos**:

```
3. INSTALLED_APPS = [
4.     'django.contrib.admin',
5.     'django.contrib.auth',
6.     'django.contrib.contenttypes',
```



```
7.     'django.contrib.sessions',
8.     'django.contrib.messages',
9.     'django.contrib.staticfiles',
10.    'corsheaders',
11.    'rest_framework',
12.    'alumnos',
13.]
```

Asegúrate que este:

```
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Agrega también:

```
CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
]
```

También agrega en ese archivo:

```
REST_FRAMEWORK = {
    'DEFAULT_RENDERER_CLASSES': [
        'rest_framework.renderers.JSONRenderer',
    ],
    'DEFAULT_PARSER_CLASSES': [
        'rest_framework.parsers.JSONParser',
    ],
    # Otros ajustes de REST Framework si los tienes
}
```

### Paso 3: Definición del Modelo de Alumno en alumnos/models.py

```
from django.db import models
```

```
class Alumno(models.Model):
    nombre = models.CharField(max_length=100)
    numero_documento = models.CharField(max_length=20, unique=True)
    nota1 = models.DecimalField(max_digits=5, decimal_places=2)
    nota2 = models.DecimalField(max_digits=5, decimal_places=2)
    nota3 = models.DecimalField(max_digits=5, decimal_places=2)

    def promedio(self):
        return (self.nota1 + self.nota2 + self.nota3) / 3

    def __str__(self):
        return self.nombre
```

python manage.py makemigrations alumnos

```
(venv) PS C:\Users\LILLYU\Documents\Github\ReactJs\crudApirestDjango\gestion_alumnos_api> python manage.py makemigrations alumnos
C:\Users\LILLYU\Documents\Github\ReactJs\crudApirestDjango\venv\Lib\site-packages\django\core\management\commands\makemigrations.py:161: RuntimeWarning: Got an
error checking a consistent migration history performed for database connection 'default': (1045, "Access denied for user 'tu_usuario_mysql'@'localhost' (using
ssword: YES)")
  warnings.warn(
Migrations for 'alumnos':
  alumnos\migrations\0001_initial.py
```

python manage.py migrate

resultado migraciones:

The screenshot shows a database management interface with a sidebar on the left containing a tree view of the database structure. The main area displays a table with columns: Tabla, Acción, Filas, Tipo, Cotejamiento, Tamaño, and Residuo a depurar. The table lists 11 tables, including alumnos\_alumno, auth\_group, auth\_group\_permissions, auth\_permission, auth\_user, auth\_user\_groups, auth\_user\_user\_permissions, django\_admin\_log, django\_content\_type, django\_migrations, and django\_session. Each row has a set of icons for actions like Examinar, Estructura, Buscar, Insertar, Vaciar, and Eliminar. At the bottom, there is a summary row showing 11 tables, 54 rows, and a total size of 416.0 KB.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
alumnos_alumno	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	32.0 KB	-
auth_group	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	32.0 KB	-
auth_group_permissions	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	48.0 KB	-
auth_permission	Examinar Estructura Buscar Insertar Vaciar Eliminar	28	InnoDB	---	32.0 KB	-
auth_user	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	32.0 KB	-
auth_user_groups	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	48.0 KB	-
auth_user_user_permissions	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	48.0 KB	-
django_admin_log	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	48.0 KB	-
django_content_type	Examinar Estructura Buscar Insertar Vaciar Eliminar	7	InnoDB	---	48.0 KB	-
django_migrations	Examinar Estructura Buscar Insertar Vaciar Eliminar	19	InnoDB	---	16.0 KB	-
django_session	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	---	32.0 KB	-
<b>11 tablas</b>	<b>Número de filas</b>	<b>54</b>	<b>InnoDB</b>		<b>416.0 KB</b>	<b>0 B</b>

Paso 5: Creación de Serializadores en alumnos/serializers.py

```
from rest_framework import serializers
from .models import Alumno

class AlumnoSerializer(serializers.ModelSerializer):
    promedio = serializers.SerializerMethodField()
```

```

class Meta:
    model = Alumno
    fields = ['id', 'nombre', 'numero_documento', 'nota1', 'nota2',
'nota3', 'promedio']

    def get_promedio(self, obj):
        return obj.promedio()

```

Paso 6: Creación de Vistas (ViewSets) en alumnos/views.py

```

from rest_framework import viewsets
from .models import Alumno
from .serializers import AlumnoSerializer

class AlumnoViewSet(viewsets.ModelViewSet):
    queryset = Alumno.objects.all()
    serializer_class = AlumnoSerializer

```

Paso 7: Configuración de las Rutas de la API en gestion\_alumnos\_api/urls.py

```

from django.contrib import admin
from django.urls import path, include
from rest_framework import routers
from alumnos.views import AlumnoViewSet

router = routers.DefaultRouter()
router.register(r'alumnos', AlumnoViewSet)

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include(router.urls)),
]

```

## Paso 8: Ejecución del Servidor Django

python manage.py runserver

Tu API REST estará disponible en <http://127.0.0.1:8000/api/alumnos/>

Reinicie servidor

```
← → ↻ ⓘ 127.0.0.1:8000/api/alumnos/

Impresión con formato estilístico ☒

[
  {
    "id": 1,
    "nombre": "Juan Pérez",
    "numero_documento": "123456789",
    "nota1": "4.50",
    "nota2": "3.80",
    "nota3": "4.20",
    "promedio": 4.166666666666667
  },
  {
    "id": 2,
    "nombre": "María Gómez",
    "numero_documento": "987654321",
    "nota1": "3.90",
    "nota2": "4.70",
    "nota3": "4.00",
    "promedio": 4.2
  },
  {
    "id": 3,
    "nombre": "Carlos López",
    "numero_documento": "112233445",
    "nota1": "3.00",
    "nota2": "3.50",
    "nota3": "3.20",
    "promedio": 3.233333333333333
  }
]
```

Consultar un dato:

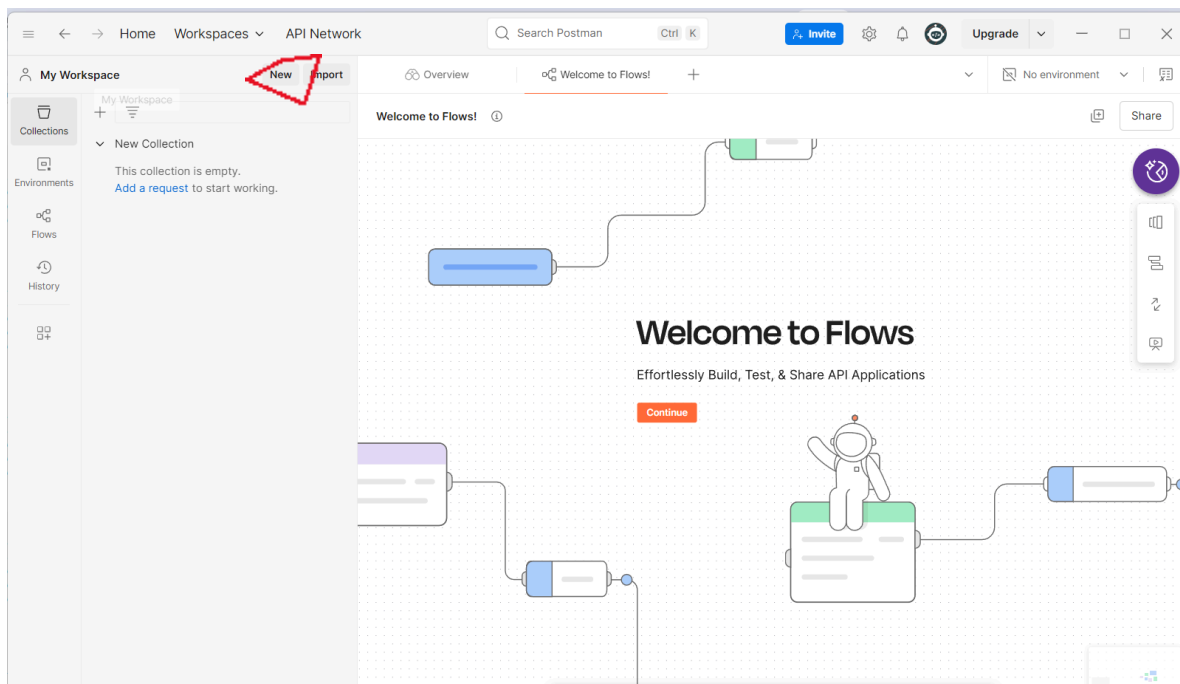
```
← → ↻ ⓘ 127.0.0.1:8000/api/alumnos/1/

Impresión con formato estilístico ☐

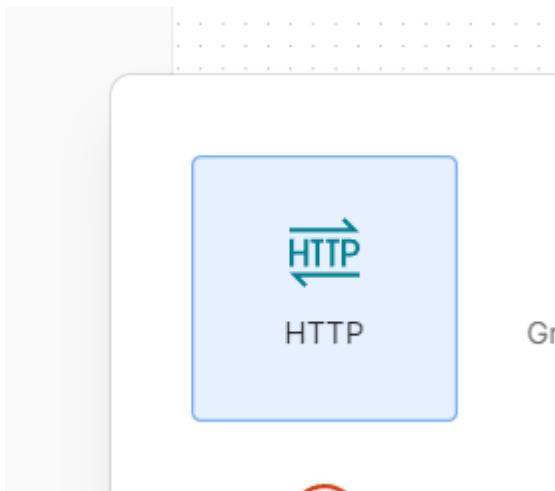
{"id":1,"nombre":"Juan Pérez","numero_documento":"123456789","nota1":"5.00","nota2":"3.80","nota3":"4.20","promedio":4.333333333333333}
```

Debes reiniciar el servidor

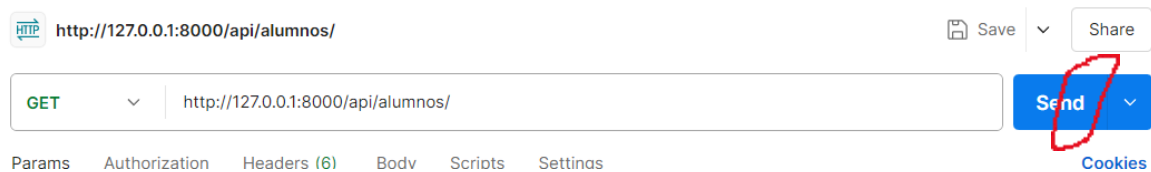
Probar en postman



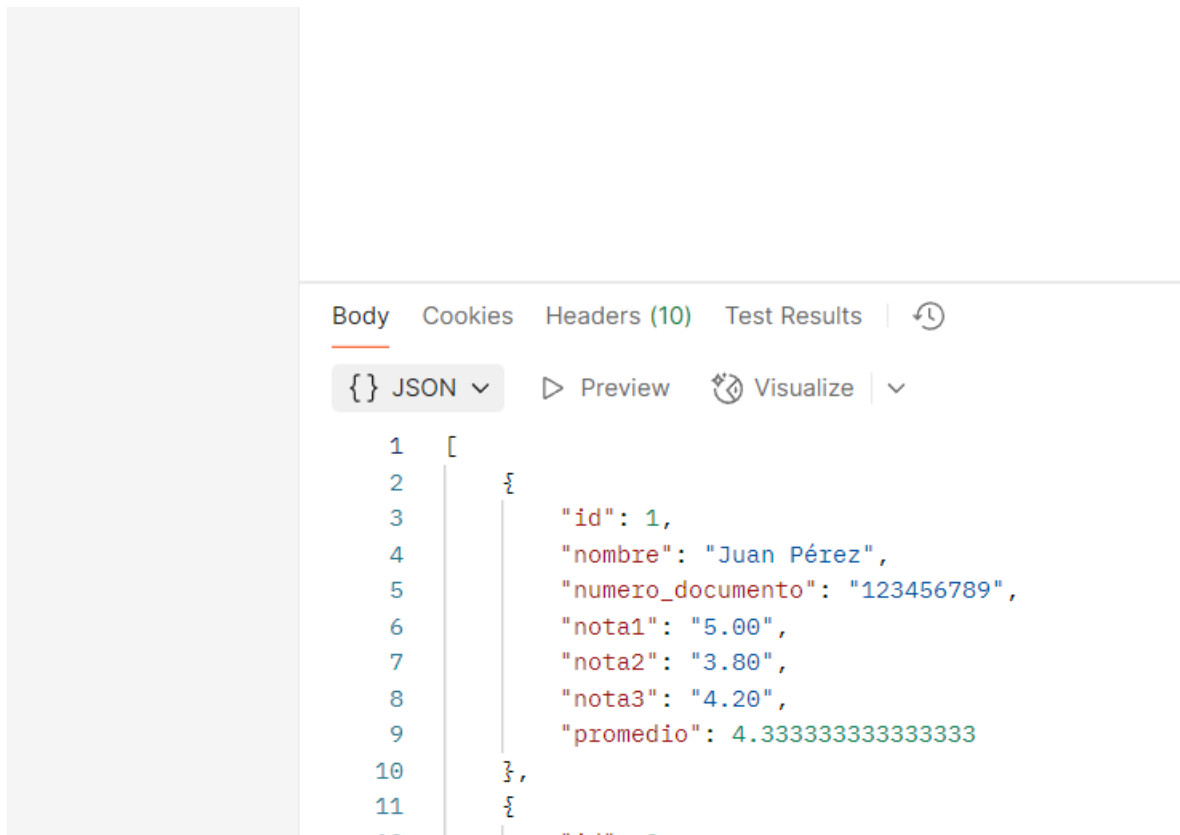
Seleccionar new y



copiar la url



Seleccionar enviar



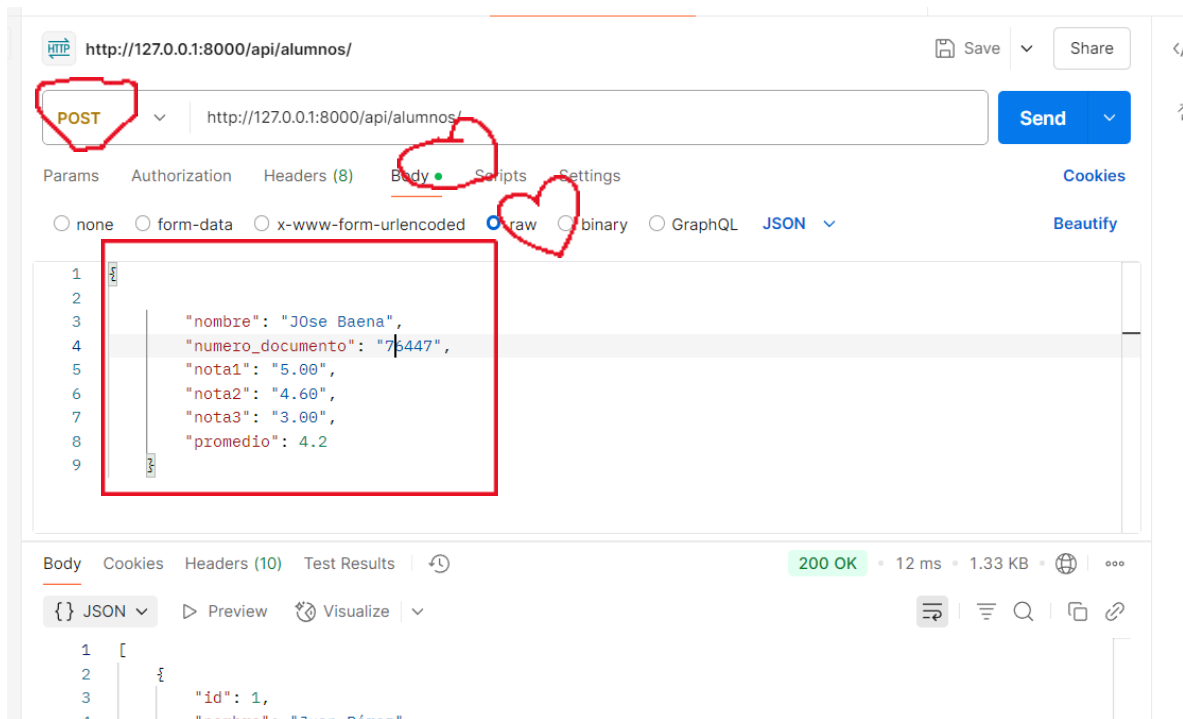
Para crear: debe tener configurado el área de trabajo tal como lo muestra la imagen:

Nota: permite masivos , por la codificación django

```
[
  {
    "nombre": "Margatita Escobar",
    "numero_documento": "446447",
    "nota1": 5.00,
    "nota2": 4.60,
    "nota3": 3.00
  },
  {
```

```
"nombre": "Yolana Mosquera",  
  
"numero_documento": "10076447",  
  
"nota1": 5.00,  
  
"nota2": 4.60,  
  
"nota3": 3.00  
  
}
```

]

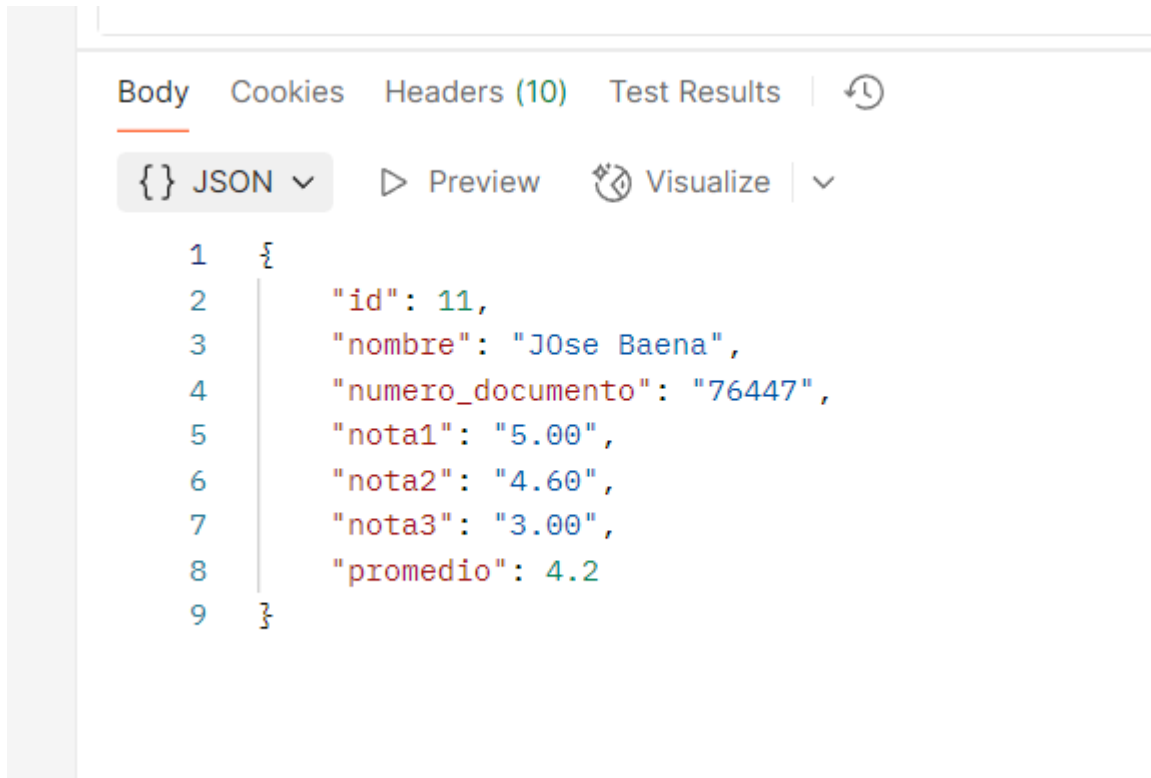


{

```
"nombre": "JOse Baena",  
  
"numero_documento": "76447",  
  
"nota1": "5.00",
```

```
"nota2": "4.60",  
"nota3": "3.00",  
"promedio": 4.2  
}
```

### Respuesta exitosa



**Eliminar tal como aparece en la imagen:**



HTTP <http://127.0.0.1:8000/api/alumnos/11/>

**DELETE** ▼ <http://127.0.0.1:8000/api/alumnos/11/>

Params Authorization Headers (8) **Body ●** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

```
1 {
2
3     "nombre": "Jose Baena",
4     "numero_documento": "76447",
5     "nota1": "5.00",
6     "nota2": "4.60",
7     "nota3": "3.00",
8     "promedio": 4.2
9 }
```

Body Cookies Headers (9) Test Results ↺

📄 Raw ▼ ▶ Preview 🔄 Visualize ▼

1

**Confirmación:**



http://127.0.0.1:8000/api/alumnos/11/

GET



http://127.0.0.1:8000/api/alumnos/11/

Params

Authorization

Headers (8)

Body ●

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

```
1 {
2
3     "nombre": "Jose Baena",
4     "numero_documento": "76447",
5     "nota1": "5.00",
6     "nota2": "4.60",
7     "nota3": "3.00",
8     "promedio": 4.2
9 }
```

Body

Cookies

Headers (10)

Test Results



{ } JSON ▼

▶ Preview

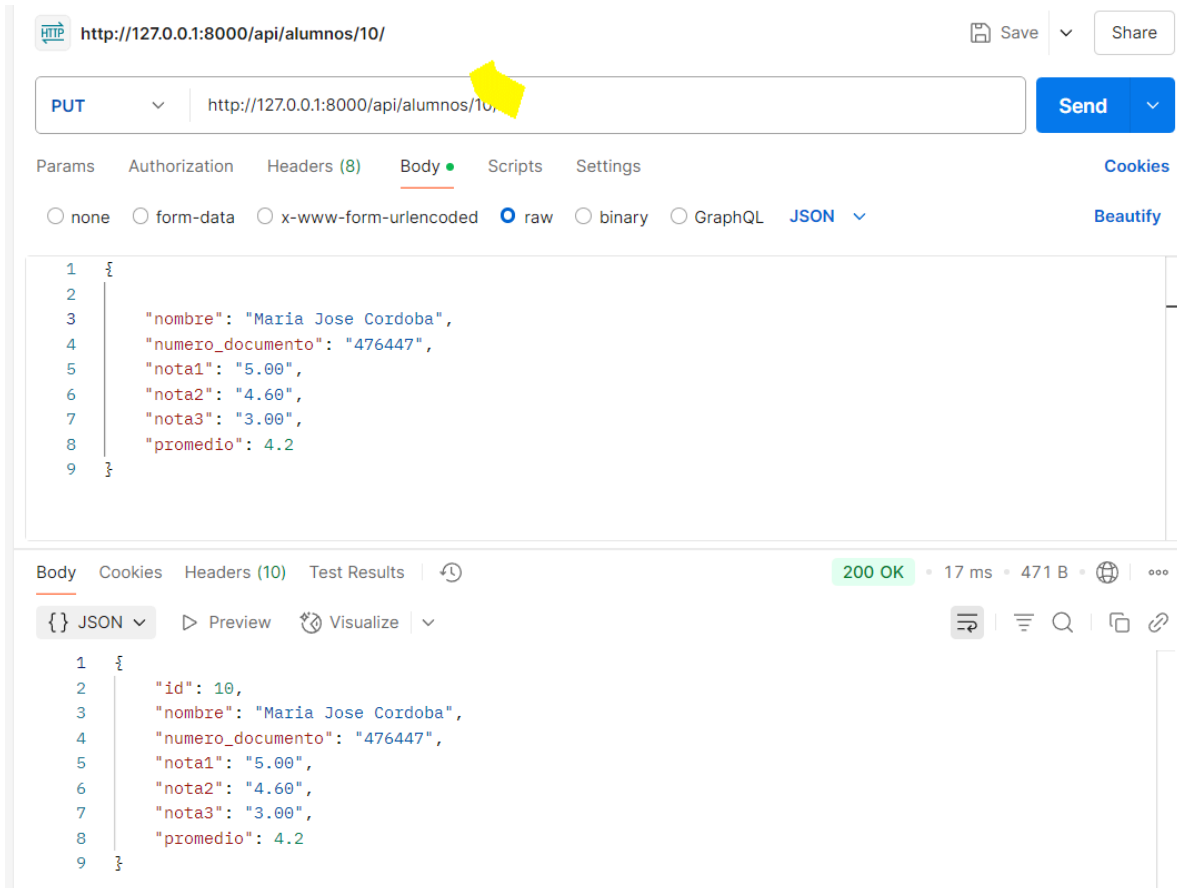


Visualize ▼

```
1 {
2   "detail": "No Alumno matches the given query."
3 }
```

**Actualizar:**

**Debe fijar cual código modificar**



## Frontend (React.js)

### Paso 1: Configuración del Proyecto React

1. Asegúrate de tener Node.js y npm (o yarn) instalados.
2. Crea un nuevo proyecto React:

```
npx create-react-app gestion_alumnos_frontend
```

```
cd gestion_alumnos_frontend
```

### Paso 2: Instalación de Dependencias

```
npm install axios react-router-dom bootstrap react-bootstrap
```

```
npm install react-icons
```

### Paso 3: Creación de Componentes React

Crea componentes para listar, crear, actualizar y eliminar alumnos. Aquí te doy una idea de la estructura:

- **src/components/ListaAlumnos.js:** Muestra la lista de alumnos y botones para crear, editar y eliminar.
- **src/components/CrearAlumno.js:** Formulario para crear un nuevo alumno.
- **src/components/EditarAlumno.js:** Formulario para editar la información de un alumno existente.

Ejemplo básico de `src/components/ListaAlumnos.js`:

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import { Table, Button, Alert } from 'react-bootstrap';
import { FaEdit, FaTrash } from 'react-icons/fa'; // Importa los iconos para
editar y eliminar

// Componente ListaAlumnos: muestra una lista de alumnos con opciones para
editar y eliminar
const ListaAlumnos = () => {
  // Estado para almacenar la lista de alumnos
  const [alumnos, setAlumnos] = useState([]);

  // Estado para manejar la carga de datos
  const [loading, setLoading] = useState(true);

  // Estado para manejar errores al cargar los datos
  const [error, setError] = useState(null);

  // Estado para mostrar alertas al eliminar un alumno
  const [deleteAlert, setDeleteAlert] = useState(null);

  // Tamaño de los iconos para las acciones
  const iconSize = 20;

  // Hook useEffect: carga la lista de alumnos al montar el componente
  useEffect(() => {
    fetchAlumnos(); // Llama a la función para obtener los alumnos
  }, []);

  // Función para obtener la lista de alumnos desde la API
  const fetchAlumnos = async () => {
    setLoading(true); // Indica que los datos están cargando
    setError(null); // Reinicia el estado de error
    try {
      // Realiza una solicitud GET para obtener los alumnos
    }
  }
}
```

```

        const response = await
axios.get('http://127.0.0.1:8000/api/alumnos/');
        setAlumnos(response.data); // Actualiza el estado con los datos
obtenidos
    } catch (err) {
        // Maneja errores al cargar los datos
        setError(err.message || 'Error al obtener alumnos.');
```

} finally {

setLoading(false); // Finaliza el estado de carga

}

};

// Función para eliminar un alumno

const handleDelete = async (id) => {

// Confirma si el usuario desea eliminar el alumno

if (window.confirm("¿Estás seguro de eliminar este alumno?")) {

try {

// Realiza una solicitud DELETE para eliminar el alumno

await

axios.delete(`http://127.0.0.1:8000/api/alumnos/\${id}/`);

// Muestra una alerta de éxito

setDeleteAlert({ variant: 'success', message: 'Alumno

eliminado exitosamente!' });

fetchAlumnos(); // Recarga la lista de alumnos después de

eliminar

setTimeout(() => setDeleteAlert(null), 3000); // Oculta la

alerta después de 3 segundos

} catch (err) {

// Maneja errores al eliminar el alumno

setDeleteAlert({ variant: 'danger', message: err.message ||

'Error al eliminar alumno.' });

setTimeout(() => setDeleteAlert(null), 3000); // Oculta la

alerta después de 3 segundos

}

}

};

// Muestra un mensaje de carga mientras se obtienen los datos

if (loading) {

return <div>Cargando alumnos...</div>;

}

// Muestra un mensaje de error si ocurre un problema al cargar los datos

if (error) {

return <Alert variant="danger">{error}</Alert>;

```

    }

    return (
      <div>
        { /* Título de la página */ }
        <h2>Lista de Alumnos</h2>

        { /* Botón para crear un nuevo alumno */ }
        <Link to="/crear" className="btn btn-primary mb-3">Crear Nuevo
Alumno</Link>

        { /* Alerta condicional: se muestra si hay un mensaje en
deleteAlert */ }
        {deleteAlert && <Alert
variant={deleteAlert.variant}>{deleteAlert.message}</Alert>}

        { /* Tabla para mostrar la lista de alumnos */ }
        <Table striped bordered hover>
          <thead>
            <tr>
              <th>ID</th>
              <th>Nombre</th>
              <th>Documento</th>
              <th>Nota 1</th>
              <th>Nota 2</th>
              <th>Nota 3</th>
              <th>Promedio</th>
              <th>Acciones</th>
            </tr>
          </thead>
          <tbody>
            { /* Mapea los alumnos y genera una fila para cada uno
*/ }

            {alumnos.map(alumno => (
              <tr key={alumno.id}>
                <td>{alumno.id}</td>
                <td>{alumno.nombre}</td>
                <td>{alumno.numero_documento}</td>
                <td>{alumno.nota1}</td>
                <td>{alumno.nota2}</td>
                <td>{alumno.nota3}</td>
                <td>{alumno.promedio ?
alumno.promedio.toFixed(2) : 'N/A'}</td>
                <td className="d-flex gap-2"> { /* Usa flex para
alinear los iconos */ }

```

```

                                {/* Botón para editar el alumno */}
                                <Link to={`/editar/${alumno.id}`}
className="btn btn-warning btn-sm">
                                <FaEdit size={iconSize} /> {/* Icono de
editar */}

                                </Link>
                                {/* Botón para eliminar el alumno */}
                                <Button variant="danger" btn-sm onClick={()
=> handleDelete(alumno.id)}>
                                <FaTrash size={iconSize} /> {/* Icono de
eliminar */}

                                </Button>
                                </td>
                                </tr>
                                }}}
                                </tbody>
                                </Table>
                                </div>
                                );
};

export default ListaAlumnos;

```

#### Paso 4: Configuración de Rutas en src/App.js

```

import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from
'react-router-dom';
import ListaAlumnos from './components/ListaAlumnos';
import CrearAlumno from './components/CrearAlumno';
import EditarAlumno from './components/EditarAlumno'; // Importa
EditarAlumno
import 'bootstrap/dist/css/bootstrap.min.css';
import { Container, Nav, Navbar } from 'react-bootstrap';

function App() {
  return (
    <Router>
      <Navbar bg="light" expand="lg">
        <Container>
          <Navbar.Brand as={Link} to="/">Gestión de
Alumnos</Navbar.Brand>

```

```

        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
            <Nav className="me-auto">
                <Nav.Link as={Link} to="/">Lista de
Alumnos</Nav.Link>
                <Nav.Link as={Link} to="/crear">Crear
Alumno</Nav.Link>
            </Nav>
        </Navbar.Collapse>
    </Container>
</Navbar>
<Container className="mt-3">
    <Routes>
        <Route path="/" element={<ListaAlumnos />} />
        <Route path="/crear" element={<CrearAlumno />} />
        <Route path="/editar/:id" element={<EditarAlumno />} />
    </Routes>
    { /* Nueva ruta para editar */ }
    </Container>
</Router>
);
}

export default App;

```

### Paso 5: Implementación de los Componentes CrearAlumno.js y EditarAlumno.js

Estos componentes contendrán formularios para crear y editar la información de los alumnos, utilizando axios para hacer las peticiones POST y PUT a la API de Django.

**Ejemplo básico de src/components/CrearAlumno.js:**

```

import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate, Link } from 'react-router-dom';
import { Form, Button, Alert } from 'react-bootstrap';

// Componente CrearAlumno: permite crear un nuevo alumno enviando los datos
a una API REST
const CrearAlumno = () => {
    // Estado inicial del formulario
    const [alumno, setAlumno] = useState({
        nombre: '', // Nombre del alumno
        numero_documento: '', // Número de documento del alumno
    });

```



```

        nota1: '', // Nota 1 del alumno
        nota2: '', // Nota 2 del alumno
        nota3: '', // Nota 3 del alumno
    });

    // Hook para redirigir al usuario a otra ruta
    const navigate = useNavigate();

    // Estado para mostrar alertas (éxito o error)
    const [showAlert, setShowAlert] = useState(null);

    // Maneja los cambios en los campos del formulario
    const handleChange = (e) => {
        // Actualiza el estado del formulario con el valor ingresado
        setAlumno({ ...alumno, [e.target.name]: e.target.value });
    };

    // Maneja el envío del formulario
    const handleSubmit = async (e) => {
        e.preventDefault(); // Previene el comportamiento por defecto del
formulario
        try {
            // Envía los datos del formulario a la API
            const response = await
axios.post('http://127.0.0.1:8000/api/alumnos/', alumno);
            // Muestra una alerta de éxito
            setShowAlert({ variant: 'success', message: 'Alumno creado
exitosamente!' });
            // Redirige al usuario a la página principal después de 1.5
segundos
            setTimeout(() => navigate('/'), 1500);
        } catch (error) {
            // Muestra una alerta de error si ocurre un problema
            console.error("Error al crear alumno:", error);
            setShowAlert({ variant: 'danger', message: error.message ||
'Error al crear el alumno.' });
        }
    };

    return (
        <div>
            {/* Título del formulario */}
            <h2>Crear Nuevo Alumno</h2>

```

```

        { /* Alerta condicional: se muestra si hay un mensaje en
showAlert */}
        {showAlert && (
            <Alert variant={showAlert.variant} onClose={() =>
setShowAlert(null)} dismissible>
                {showAlert.message}
            </Alert>
        )}

        { /* Formulario para crear un nuevo alumno */}
        <Form onSubmit={handleSubmit}>
            { /* Campo para el nombre del alumno */}
            <Form.Group className="mb-3">
                <Form.Label>Nombre</Form.Label>
                <Form.Control
                    type="text"
                    name="nombre"
                    value={alumno.nombre}
                    onChange={handleChange}
                    required
                />
            </Form.Group>

            { /* Campo para el número de documento */}
            <Form.Group className="mb-3">
                <Form.Label>Número de Documento</Form.Label>
                <Form.Control
                    type="text"
                    name="numero_documento"
                    value={alumno.numero_documento}
                    onChange={handleChange}
                    required
                />
            </Form.Group>

            { /* Campo para la Nota 1 */}
            <Form.Group className="mb-3">
                <Form.Label>Nota 1</Form.Label>
                <Form.Control
                    type="number"
                    step="0.01"
                    name="nota1"
                    value={alumno.nota1}
                    onChange={handleChange}
                    required
                />
            </Form.Group>
        </Form>
    )}

```

```

        />
      </Form.Group>

      { /* Campo para la Nota 2 */ }
      <Form.Group className="mb-3">
        <Form.Label>Nota 2</Form.Label>
        <Form.Control
          type="number"
          step="0.01"
          name="nota2"
          value={alumno.nota2}
          onChange={handleChange}
          required
        />
      </Form.Group>

      { /* Campo para la Nota 3 */ }
      <Form.Group className="mb-3">
        <Form.Label>Nota 3</Form.Label>
        <Form.Control
          type="number"
          step="0.01"
          name="nota3"
          value={alumno.nota3}
          onChange={handleChange}
          required
        />
      </Form.Group>

      { /* Botón para enviar el formulario */ }
      <Button variant="primary" type="submit">Guardar</Button>

      { /* Enlace para cancelar y volver a la página principal */ }
      <Link to="/" className="btn btn-secondary
ml-2">Cancelar</Link>
    </Form>
  </div>
);
};

export default CrearAlumno;

```

### Ejemplo básico de src/components/EditarAlumno.js:

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useNavigate, Link, useParams } from 'react-router-dom';
import { Form, Button, Alert } from 'react-bootstrap';

// Componente EditarAlumno: permite editar los datos de un alumno existente
const EditarAlumno = () => {
  // Obtiene el parámetro `id` de la URL
  const { id } = useParams();

  // Hook para redirigir al usuario a otra ruta
  const navigate = useNavigate();

  // Estado inicial del formulario con los datos del alumno
  const [alumno, setAlumno] = useState({
    nombre: '', // Nombre del alumno
    numero_documento: '', // Número de documento del alumno
    nota1: '', // Nota 1 del alumno
    nota2: '', // Nota 2 del alumno
    nota3: '', // Nota 3 del alumno
  });

  // Estado para manejar la carga de datos
  const [loading, setLoading] = useState(true);

  // Estado para manejar errores al cargar o actualizar datos
  const [error, setError] = useState(null);

  // Estado para mostrar alertas (éxito o error)
  const [showAlert, setShowAlert] = useState(null);

  // Hook useEffect: carga los datos del alumno al montar el componente
  useEffect(() => {
    const fetchAlumno = async () => {
      setLoading(true); // Indica que los datos están cargando
      setError(null); // Reinicia el estado de error
      try {
        // Realiza una solicitud GET para obtener los datos del
        alumno
        const response = await
        axios.get(`http://127.0.0.1:8000/api/alumnos/${id}/`);
        setAlumno(response.data); // Actualiza el estado con los
        datos del alumno
      } catch (error) {
        setError(error);
      }
    };
    fetchAlumno();
  }, [id]);
};
```

```

        } catch (err) {
            // Maneja errores al cargar los datos
            setError(err.message || 'Error al cargar el alumno.');
```

} finally {

```
            setLoading(false); // Finaliza el estado de carga
        }
    };

    fetchAlumno();
}, [id]); // Se ejecuta cuando cambia el `id`

// Maneja los cambios en los campos del formulario
const handleChange = (e) => {
    // Actualiza el estado del formulario con el valor ingresado
    setAlumno({ ...alumno, [e.target.name]: e.target.value });
};

// Maneja el envío del formulario
const handleSubmit = async (e) => {
    e.preventDefault(); // Previene el comportamiento por defecto del
formulario
    try {
        // Realiza una solicitud PUT para actualizar los datos del
alumno
        await axios.put(`http://127.0.0.1:8000/api/alumnos/${id}/`,
alumno);
        // Muestra una alerta de éxito
        setShowAlert({ variant: 'success', message: 'Alumno actualizado
exitosamente!' });
        // Redirige al usuario a la página principal después de 1.5
segundos
        setTimeout(() => navigate('/'), 1500);
    } catch (err) {
        // Maneja errores al actualizar los datos
        console.error("Error al actualizar alumno:", err);
        setShowAlert({ variant: 'danger', message: err.message || 'Error
al actualizar el alumno.' });
    }
};

// Muestra un mensaje de carga mientras se obtienen los datos
if (loading) {
    return <div>Cargando información del alumno...</div>;
}

```

```

// Muestra un mensaje de error si ocurre un problema al cargar los datos
if (error) {
    return <Alert variant="danger">{error}</Alert>;
}

return (
    <div>
        {/* Título del formulario */}
        <h2>Editar Alumno</h2>

        {/* Alerta condicional: se muestra si hay un mensaje en
showAlert */}
        {showAlert && (
            <Alert variant={showAlert.variant} onClose={() =>
setShowAlert(null)} dismissible>
                {showAlert.message}
            </Alert>
        )}

        {/* Formulario para editar los datos del alumno */}
        <Form onSubmit={handleSubmit}>
            {/* Campo para el nombre del alumno */}
            <Form.Group className="mb-3">
                <Form.Label>Nombre</Form.Label>
                <Form.Control
                    type="text"
                    name="nombre"
                    value={alumno.nombre}
                    onChange={handleChange}
                    required
                />
            </Form.Group>

            {/* Campo para el número de documento */}
            <Form.Group className="mb-3">
                <Form.Label>Número de Documento</Form.Label>
                <Form.Control
                    type="text"
                    name="numero_documento"
                    value={alumno.numero_documento}
                    onChange={handleChange}
                    required
                />
            </Form.Group>

```

```

    { /* Campo para la Nota 1 */ }
    <Form.Group className="mb-3">
      <Form.Label>Nota 1</Form.Label>
      <Form.Control
        type="number"
        step="0.01"
        name="nota1"
        value={alumno.nota1}
        onChange={handleChange}
        required
      />
    </Form.Group>

    { /* Campo para la Nota 2 */ }
    <Form.Group className="mb-3">
      <Form.Label>Nota 2</Form.Label>
      <Form.Control
        type="number"
        step="0.01"
        name="nota2"
        value={alumno.nota2}
        onChange={handleChange}
        required
      />
    </Form.Group>

    { /* Campo para la Nota 3 */ }
    <Form.Group className="mb-3">
      <Form.Label>Nota 3</Form.Label>
      <Form.Control
        type="number"
        step="0.01"
        name="nota3"
        value={alumno.nota3}
        onChange={handleChange}
        required
      />
    </Form.Group>

    { /* Botón para enviar el formulario */ }
    <Button variant="primary" type="submit">Guardar
Cambios</Button>

    { /* Enlace para cancelar y volver a la página principal */ }

```

```

        <Link to="/" className="btn btn-secondary
ml-2">Cancelar</Link>
      </Form>
    </div>
  );
};

export default EditarAlumno;

```

El componente EditarAlumno.js sería similar a CrearAlumno.js, pero realizaría una petición GET para obtener los datos del alumno a editar y una petición PUT para guardar los cambios.

El componente EditarAlumno.js sería similar a CrearAlumno.js, pero realizaría una petición GET para obtener los datos del alumno a editar y una petición PUT para guardar los cambios.

## Paso 6: Ejecución del Frontend React









```
cd gestion_alumnos_frontend
```

```
npm start
```

Gestión de Alumnos   Lista de Alumnos   Crear Alumno

### Lista de Alumnos

Crear Nuevo Alumno

ID	Nombre	Documento	Nota 1	Nota 2	Nota 3	Promedio	Acciones
1	Juan Pérez	123456789	5.00	3.80	4.20	4.33	 
2	María Gómez	987654321	3.90	4.70	4.00	4.20	 
3	Carlos López	112233445	3.00	3.50	3.20	3.23	 
6	CARLOS VELEZ	6447	5.00	4.60	3.00	4.20	 



## Editar Alumno

Nombre

Juan Pérez

Número de Documento

123456789

Nota 1

5,00

Nota 2

3,80

Nota 3

4,20

Guardar Cambios

Cancelar