

backend-evaluation-grupo-almo

API RESTful creada como parte de la evaluación técnica para el puesto de Desarrollador Backend en Grupo Almo. Incluye autenticación JWT, gestión de usuarios y roles, operaciones CRUD, y validaciones. Tecnologías utilizadas: Python, MySQL.

Documentación de la API RESTful para la Gestión de Usuarios y Roles

Tabla de Contenidos

- 1. [Introducción](#)
- 2. [Tecnologías Utilizadas](#)
- 3. [Instalación](#)
- 4. [Configuración de Variables de Entorno](#)
- 5. [Estructura del Proyecto](#)
- 6. [Endpoints](#)
 - [Registro de Usuario](#)
 - [Inicio de Sesión](#)
 - [Obtener Usuarios](#)
 - [Registrar Departamento](#)
 - [Listar Departamentos](#)
 - [Registrar Proyecto](#)
 - [Calcular Salario Promedio](#)
 - [Contar Empleados por Departamento](#)
 - [Obtener Cumpleaños del Mes](#)
- 7. [Consideraciones Finales](#)

Introducción

Esta API RESTful permite la gestión de usuarios y roles, así como la administración de departamentos y proyectos. Se utiliza para aplicaciones donde es necesario manejar información sobre empleados y sus roles dentro de una organización.

Tecnologías Utilizadas

- **Flask:** Microframework para Python utilizado para desarrollar la API.
- **Flask-JWT-Extended:** Para manejar la autenticación mediante JSON Web Tokens (JWT).
- **MySQL:** Sistema de gestión de bases de datos utilizado para almacenar información.
- **python-dotenv:** Para manejar las variables de entorno.

Instalación

1. Clona el repositorio:

```
git clone <url_del_repositorio>  
cd backend-evaluation-grupo-almo
```

2. Crea un entorno virtual:

```
python -m venv venv
```

3. Activa el entorno virtual:

```
venv\Scripts\activate
```

4. Instala las dependencias:

```
pip install -r requirements.txt
```

5. Iniciar la api

```
python app.py
```

Configuración de Variables de Entorno

Crea un archivo .env en la raíz del proyecto con el siguiente contenido:

```
DB_HOST=localhost  
DB_USER=tu_usuario  
DB_PASSWORD=tu_contraseña  
DB_NAME=tu_basedatos
```

Estructura del Proyecto

```
backend-evaluation-grupo-almo/  
├── src/  
│   ├── app.py  
│   ├── auth.py  
│   ├── db.py  
│   ├── models.py  
│   ├── routes.py  
│   └── validations.py
```

Endpoints

The screenshot shows a VS Code editor with a REST client file named `users.http`. The request is a POST to `http://127.0.0.1:5000/login` with a JSON body containing `email` and `password`. The response is a 200 OK status with headers for `Server`, `Date`, `Content-Type`, and `Content-Length`, and a JSON body containing an `access_token`.

```
py 1, M requirements.txt M users.http U models.py U routes.py 1, M ... Response(23ms) X
```

```
users.http > ...
Authorization: Bearer <your_jwt_token> # Reemplaza con un token JWT válido s

#####
### INICIAR SESION ANTES ###
#####

### Iniciar sesión para obtener un token

Send Request
POST http://127.0.0.1:5000/login
Content-Type: application/json

{
  "email": "alice@example.com",
  "password": "Password123"
}
```

```
http://...:...
Server: Werkzeug/2.2.2 Python/3.11.5
Date: Thu, 10 Oct 2024 19:10:25 GMT
Content-Type: application/json
Content-Length: 310
Connection: close

1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.2 Python/3.11.5
3 Date: Thu, 10 Oct 2024 19:10:25 GMT
4 Content-Type: application/json
5 Content-Length: 310
6 Connection: close
7
8 {
9   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXc
cy00UzZyNyNwianRpIjoiaWY3ZmE5YzYtYzdkNmJ00jM2LWlX
Y2VzcyIsInN1YiI6ImFsaWkiLCJ0VG4yVWlwbGUuY291IiwibmJmIjo
V99.bxwVotzAY5fA8Y1RTroVoNx5t0NuZm9qEQ0p0FHqJuf8"
10 }
```

```
ts > curl -X POST http://127.0.0.1:5000/users

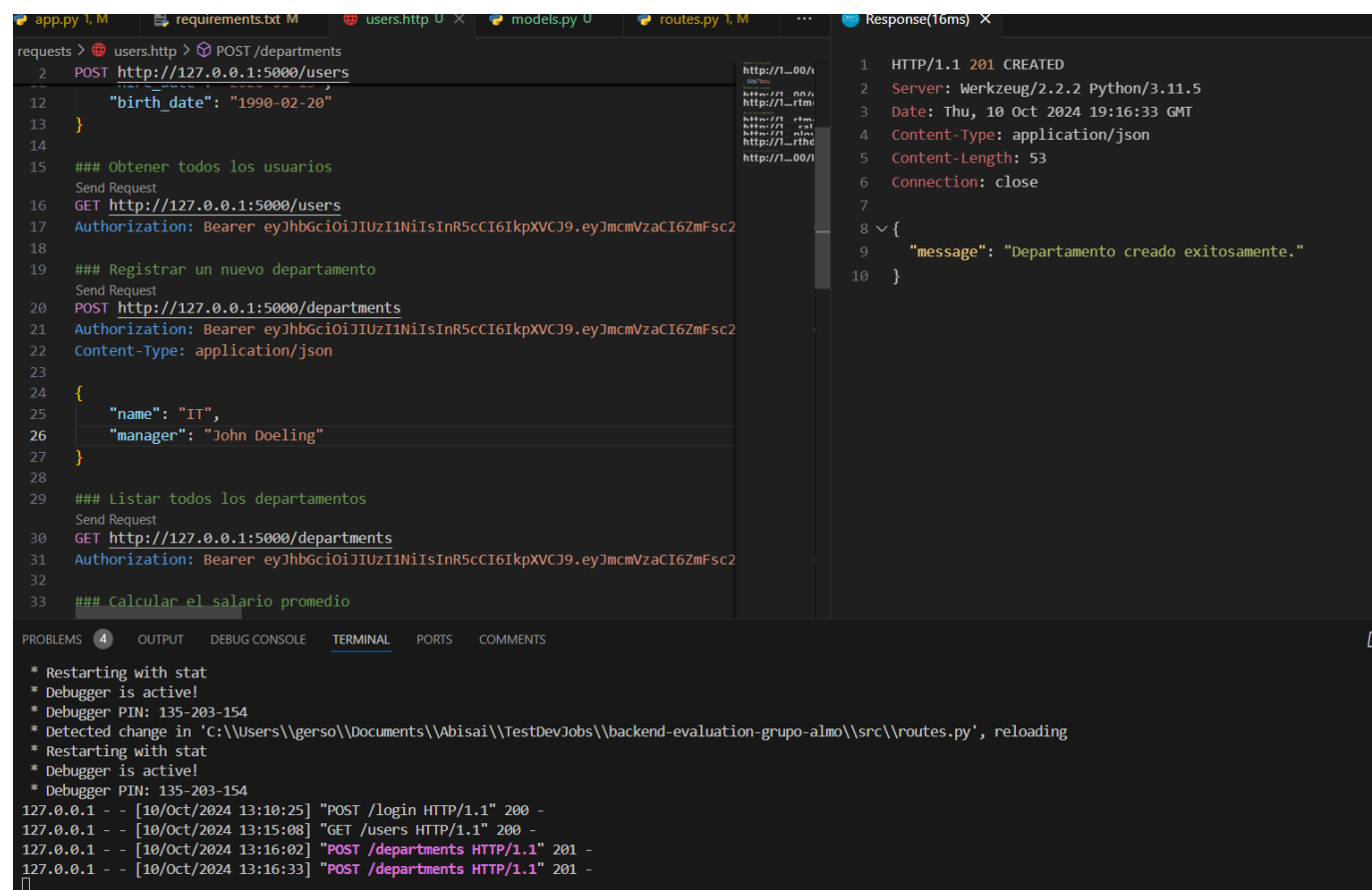
{
  "name": "Alice",
  "email": "alice@example.com",
  "password": "Password123",
  "role": "admin",
  "salary": 50000,
  "hire_date": "2020-01-15",
  "birth_date": "1990-02-20"
}

### Obtener todos los usuarios
Send Request
GET http://127.0.0.1:5000/users
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2Q

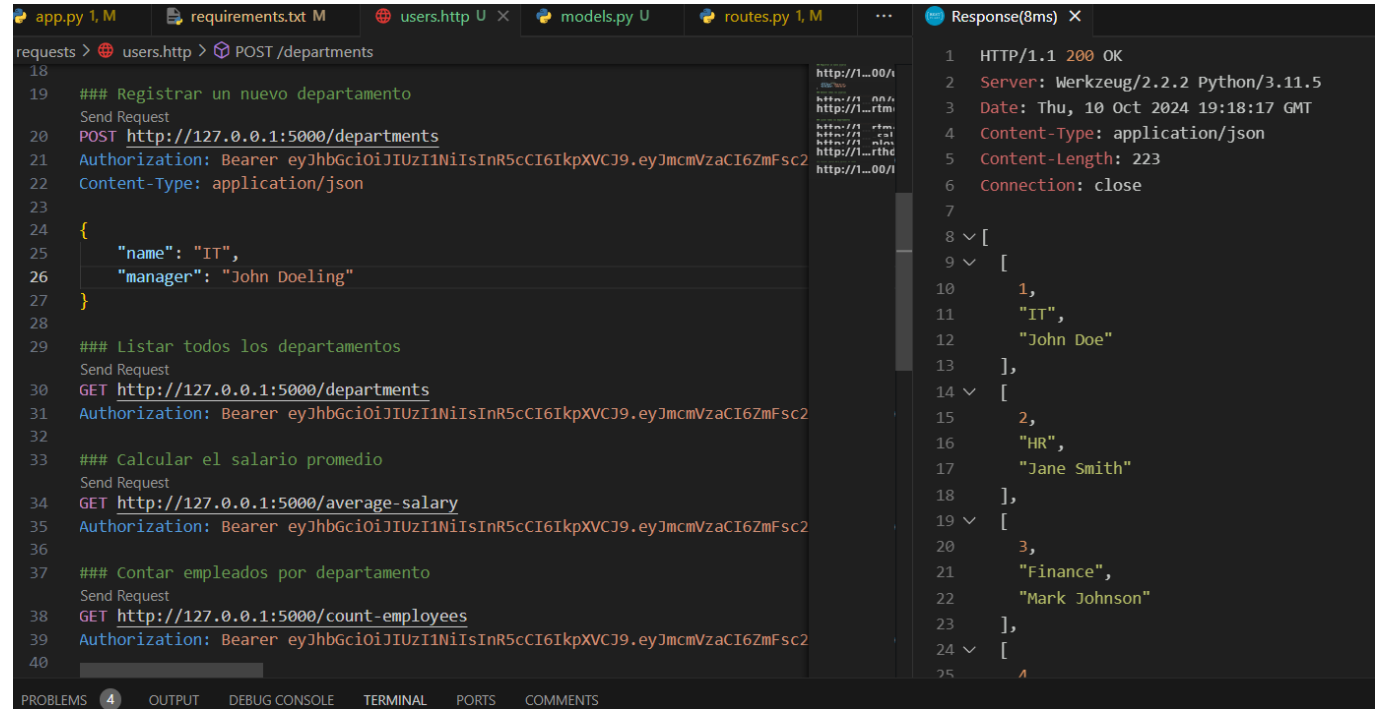
### Registrar un nuevo departamento
Send Request
POST http://127.0.0.1:5000/departments
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2Q
Content-Type: application/json

{
  "name": "IT",
  "manager": "John Doe"
}
```

3 / 6



Listar Departamentos



Calcular Salario Promedio

```
app.py 1, M  requirements.txt M  users.http U X  models.py U  routes.py 1, M  ...  Response(20ms) X

requests > users.http > POST /departments
18
19  ### Registrar un nuevo departamento
20  Send Request
21  POST http://127.0.0.1:5000/departments
22  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2
23  Content-Type: application/json
24  {
25    "name": "IT",
26    "manager": "John Doeling"
27  }
28
29  ### Listar todos los departamentos
30  Send Request
31  GET http://127.0.0.1:5000/departments
32  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2
33
34  ### Calcular el salario promedio
35  Send Request
36  GET http://127.0.0.1:5000/average-salary
37  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2
38
39  ### Contar empleados por departamento
40  Send Request
41  GET http://127.0.0.1:5000/count-employees
42  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2
43
44  HTTP/1.1 200 OK
45  Server: Werkzeug/2.2.2 Python/3.11.5
46  Date: Thu, 10 Oct 2024 19:17:17 GMT
47  Content-Type: application/json
48  Content-Length: 39
49  Connection: close
50  {
51    "average_salary": "40000.000000"
52  }
```

Cumpleaños del mes

```
sts > users.http > POST /users
Send Request
GET http://127.0.0.1:5000/average-salary
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2

### Contar empleados por departamento
Send Request
GET http://127.0.0.1:5000/count-employees
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2

### Obtener cumpleaños del mes en curso
Send Request
GET http://127.0.0.1:5000/birthdays
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2

#####
### INICIAR SESION ANTES ###
#####

### Iniciar sesión para obtener un token
Send Request
POST http://127.0.0.1:5000/login
Content-Type: application/json
{
  "email": "alice@example.com",
  "password": "Password123"
}

HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.11.5
Date: Thu, 10 Oct 2024 19:21:55 GMT
Content-Type: application/json
Content-Length: 175
Connection: close
[
  6,
  "Almo",
  "almo@example.com",
  "Password123",
  "user",
  "50000.00",
  "Wed, 15 Jan 2020 00:00:00 GMT",
  "Sat, 20 Oct 1990 00:00:00 GMT"
]
```

Consideraciones Finales

Seguridad:

Uso de HTTPS: Para ambientes de producción, siempre debes implementar HTTPS para proteger los datos transmitidos entre el cliente y el servidor. Protección del JWT: Asegúrate de que los tokens JWT sean manejados de manera segura. Nunca los guardes en almacenamiento local del lado del cliente sin protección adecuada.

Gestión de errores:

Implementa un manejo robusto de errores para asegurarte de que los usuarios reciban mensajes significativos y no mensajes de error que revelen información interna del sistema.

Pruebas:

Implementa pruebas unitarias y de integración para asegurar que los endpoints de la API funcionen correctamente y se comporten como se espera en diferentes escenarios. Considera herramientas como pytest para pruebas unitarias en Python.