

PRUEBA TÉCNICA

Dairo Alexander Llanos Perdomo

INSTALACIÓN:

Usando el código en GitHub:

- Clonar el repositorio con el comando: git clone <https://github.com/Dairollanos/prueba-tecnica-quick.git>
- Crear un entorno virtual (en mi caso con el comando: python3 -m venv prueba-técnica)
- Activar el entorno virtual. (en mi caso desde linux suelo hacerlo con el comando: source bin/activate)
- Moverse a la carpeta raíz del proyecto (con el comando: cd prueba-técnica-quick)
- Instalar las librerías necesarias, las cuales están escritas en el archivo "requirements.txt". para instalarlas todas de una vez usar el siguiente comando: pip install -r requirements.txt
- Correr el comando para las migraciones de base de datos (python manage.py migrate), correr el servidor (python manage.py runserver) y con esto el proyecto debería estar listo para funcionar

ACCEDER LOS SERVICIOS REST

- **VISTA PROTEGIDAS:**

Las vistas protegidas lo están mediante un sistema de autenticación por token, cada usuario registrado tiene asignado un token único para el. Este token debe incluirse dentro de las cabeceras(headers) de cada petición a un endpoint protegido de la siguiente manera

```
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Si no se especifica el token al momento de enviar una petición a un endpoint protegido el servidor responderá con un código **"401 Unauthorized"**

- **Login usuario:**

para acceder al login se utiliza el siguiente endpoint: '/api/v1/users/login/ '

```
! 127.0.0.1:8000/api/v1/users/login/
```

Para acceder al sistema se hace una petición **POST** al endpoint con los datos (mobile_phone, password) como en el ejemplo siguiente:

```
{  
    "mobile_phone": "3024576840",  
    "password": "lalluviacae"  
}
```

Si la petición es correcta, el endpoint responderá con un código '**200 OK**' y retornara el token que deberá usarse para la autenticación de ese usuario en posteriores peticiones.

```
{
  "Token": "db99047725b48702895f46ad5655614c8ec0284e"
}
```

Si la petición es incorrecta, el endpoint responderá con un código '**404 Not Found**'.

- **crear usuario (protegido):**

para poder crear un usuario se utiliza el siguiente endpoint ' /api/v1/users/ ' .



```
! 127.0.0.1:8000/api/v1/users/
```

Para crear el usuario se hace una petición **POST** al endpoint con los siguientes datos (first_name, last_name, date_birth, address, password, mobile_phone, email), a continuación un ejemplo :

```
{
  "first_name": "jose",
  "last_name": "antonio",
  "date_birth": "1987-09-01",
  "address": "calle 15",
  "password": "lalluviacae",
  "mobile_phone": "3024576844",
  "email": "joseantonio@gmail.com"
}
```

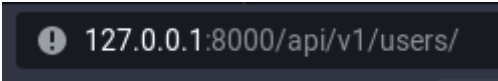
Si la petición es correcta, el endpoint responderá con un código '**200 OK**'.

Si la petición es incorrecta, el endpoint responderá con un código '**400 Bad Request**'. y retorna un mensaje con la explicación del error, ejemplo a continuación:

```
{
  "mobile_phone": ["This field is required."],
  "email": ["This field is required."]
}
```

- **obtener lista de todos usuario (protegido):**

para poder obtener la lista de todos los usuarios se utiliza el siguiente endpoint ‘
/api/v1/users/ ‘



127.0.0.1:8000/api/v1/users/

Para obtener la lista se hace una petición **GET** al endpoint.

Si la petición es correcta, el endpoint responderá con un código ‘**200 OK**’ y retornará la lista de todos los usuarios registrados en el sistema.

```
[
  {
    "id": 3,
    "first_name": "dairo",
    "last_name": "alexander",
    "date_birth": "2021-09-01",
    "address": "calle 39",
    "password": "lalluviacae",
    "mobile_phone": "3024576840",
    "email": "dairoollanos@gmail.com"
  },
  {
    "id": 8,
    "first_name": "jose",
    "last_name": "antonio",
    "date_birth": "1987-07-18",
    "address": "calle 15",
    "password": "unacontraseña",
    "mobile_phone": "3026234561",
    "email": "joseantonio@gmail.com"
  }
]
```

- **obtener usuario por id (protegido):**

para poder obtener un usuario en específico por su se utiliza el siguiente endpoint ‘
/api/v1/users/8/ ‘



127.0.0.1:8000/api/v1/users/8/

En este caso “ 8 “ corresponde al id del usuario del que se desean obtener los datos, puede ser cualquier id de un usuario previamente registrado.

Si la petición es correcta, el endpoint responderá con un código ‘**200 OK**’ y retornará todos los datos del usuario deseado

```
{
  "id": 8,
  "first_name": "jose",
  "last_name": "antonio",
  "date_birth": "1987-07-18",
  "address": "calle 15",
  "password": "unacontraseña",
  "mobile_phone": "3026234561",
  "email": "joseantonio@gmail.com"
}
```

Si la petición es incorrecta, el endpoint responderá con un código '**404 Not Found**' y retornará todos los datos del usuario deseado y retorna un mensaje con la explicación del error

```
{
  "detail": "Not found."
}
```

- **Actualizar datos del usuario(protegido):**

para poder actualizar los datos de un usuario se utiliza el siguiente endpoint
'/api/v1/users/8/ '



```
! 127.0.0.1:8000/api/v1/users/8/
```

En este caso " 8 " corresponde al id del usuario del que se desean actualizar los datos, puede ser cualquier id de un usuario previamente registrado.

Para actualizar los datos del usuario se hace una petición **PUT** al endpoint con todos los datos actualizados del usuario.

```
{
  "first_name": "jose",
  "last_name": "antonio",
  "date_birth": "1987-07-18",
  "address": "calle 15",
  "password": "unacontraseña",
  "mobile_phone": "3026234561",
  "email": "CORREOactualizado@gmail.com"
}
```

Si la petición es correcta, el endpoint responderá con un código **'200 OK'** y retornará el usuario con los datos actualizados.

```
{
  "id": 8,
  "first_name": "jose",
  "last_name": "antonio",
  "date_birth": "1987-07-18",
  "address": "calle 15",
  "password": "unacontraseña",
  "mobile_phone": "3026234561",
  "email": "CORREOactualizado@gmail.com"
}
```

Si la petición es incorrecta, el endpoint responderá con un código **'400 Bad Request'** y retorna un mensaje con la explicación del error, ejemplo a continuación:

```
{
  "email": ["This field is required."]
}
```

- **Eliminar usuario(protegido):**
para poder eliminar un usuario se utiliza el siguiente endpoint `/api/v1/users/8/`



```
127.0.0.1:8000/api/v1/users/8/
```

En este caso "8" corresponde al id del usuario que se desea eliminar, puede ser cualquier id de un usuario previamente registrado.

Si la petición es correcta, el endpoint responderá con un código '**200 OK**' y un mensaje de confirmación que el usuario fue eliminado

```
{  
  "OK": "User deleted correctly"  
}
```

Si la petición es incorrecta (id no existe), el endpoint responderá con un código '**404 Not Found**' y retorna un mensaje con la explicación del error

```
{  
  "detail": "Not found."  
}
```

TEST UNITARIOS:

dentro del archivo "test.py" se encuentran escritas pruebas para el proyecto.