

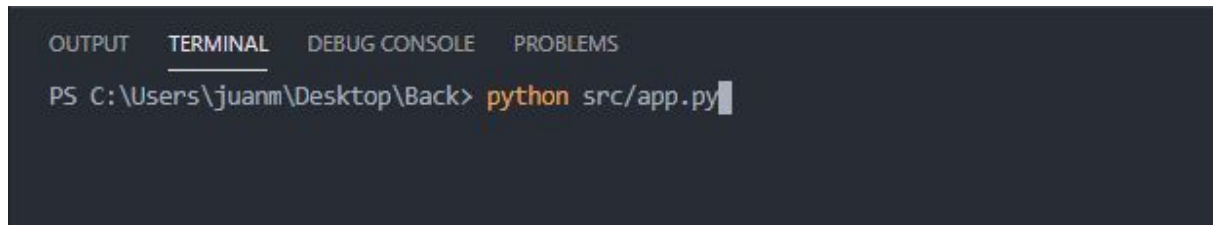
## Manual de Usuario:

el siguiente texto se realizó como una pequeña forma de mostrar la funcionalidad de la aplicación desarrollada para el **backend** del proyecto final de la materia **tecnologías para desarrollo de aplicaciones web**

**NOTA:** para ejecutar la peticiones al servidor usaremos **POSTMAN**

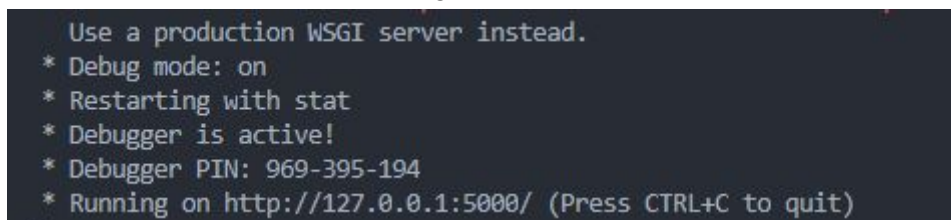
### 1. Iniciando la aplicación:

para iniciar la aplicación se ejecuta el comando `python src/app.py`



```
PS C:\Users\juanm\Desktop\Back> python src/app.py
```

y como respuestas tendremos lo siguiente

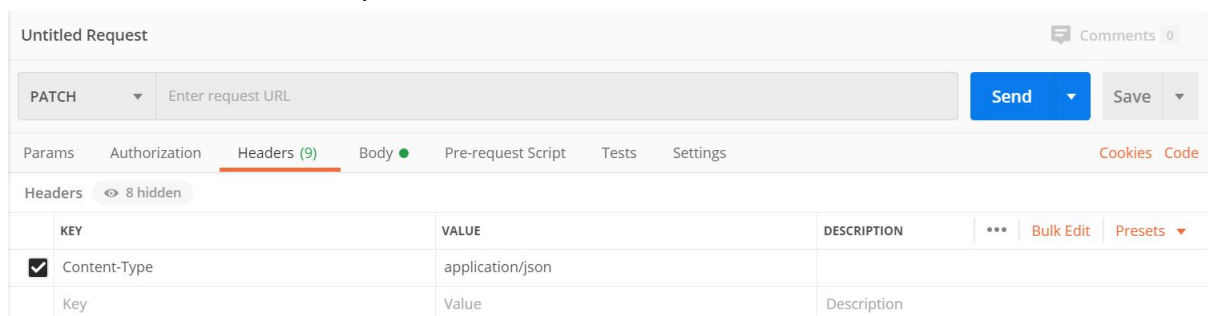


```
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 969-395-194
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

y ya podremos hacer las peticiones necesarias para el servidor para este ejemplo crearemos un usuario y un paciente, los cuales podremos ver, eliminar y elimina.

### 2. Configurando POSTMAN:

para que la aplicación funcione correctamente debemos configurar POSTMAN de la siguiente manera en el Headers para enviar correctamente los datos al servidor y realizar las debidas peticiones



### 3. Agregando un Usuario al sistema:

La aplicación se diseñó para tener un sistema de logueo al sistema paro lo cual se diseñó una función para crear un colección de usuarios.

Usando la siguiente URL en postman y con un esquema de usuario de la siguiente manera

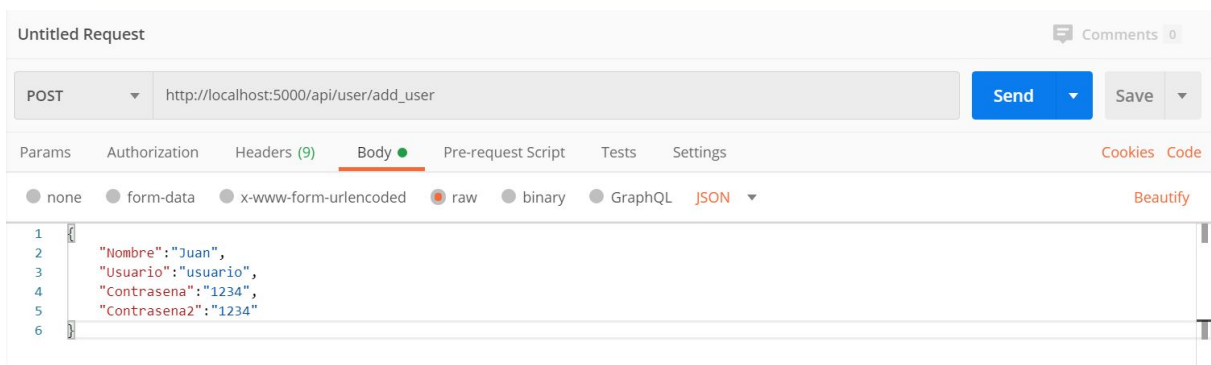
**URL:** [http://localhost:5000/api/user/add\\_user](http://localhost:5000/api/user/add_user)

### Esquema:

```
{
  "Nombre": "Juan",
  "Usuario": "usuario",
  "Contrasena": "1234",
  "Contrasena2": "1234"
}
```

### MÉTODO: POST

### EJEMPLO:



### RESPUESTA: y obtendremos la siguiente respuesta



### BASE DE DATOS: y en la Base de datos:



## 3. Agregando pacientes a la Base de datos:

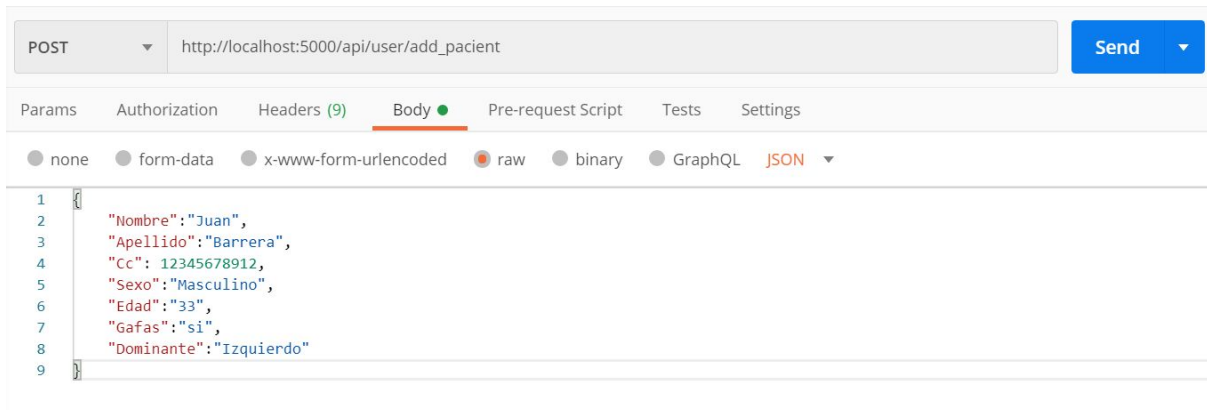
URL: [http://localhost:5000/api/user/add\\_pacient](http://localhost:5000/api/user/add_pacient)

### ESQUEMA:

```
{
  "Nombre": "Juan",
  "Apellido": "Barrera",
  "Cc": 1020465355,
  "Sexo": "Masculino",
}
```

```
"Edad": "33",
"Gafas": "si",
"Dominante": "Izquierdo"
}
```

## EJEMPLO:



## MÉTODO: POST

## RESPUESTA:



## Base de datos:

```
_id: ObjectId("5ef8169687dd511442f4e8e0")
Edad: "33"
Dominante: "Izquierdo"
Apellido: "Barrera"
CC: 12345678912
Nombre: "Juan"
Fecha: "06/27/2020"
Gafas: "si"
Sexo: "Masculino"
```

## 4. Hacer un login al sistema

URL: [http://localhost:5000/api/user/login\\_user](http://localhost:5000/api/user/login_user)

## ESQUEMA:

```
{
  "Usuario": "usuario",
  "Contraseña": "1234"
}
```

```
}
```

**MÉTODO :** GET

**EJEMPLO:**

GET http://localhost:5000/api/user/login\_user Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "Usuario": "usuario",
3   "Contraseña": "1234"
4 }
```

**RESPUESTA:**

Pretty Raw Preview Visualize JSON

```
1 {
2   "nombre": "Juan",
3   "contrasena": "1234",
4   "_id": {
5     "$oid": "5ef814d887dd511442f4e8df"
6   },
7   "usuario": "usuario"
8 }
```

## 5. Ver todos los Pacientes en el sistema

**URL:** <http://localhost:5000/api/api/user/list>

**ESQUEMA:** no es requerido

**MÉTODO:** GET

**RESPUESTA:**

```
[
  {
    "Edad": "33",
    "Gafas": "si",
    "Apellido": "Barrera",
    "Dominante": "Izquierdo",
    "Fecha": "06/27/2020",
    "Sexo": "Masculino",
    "CC": 12345678912,
    "Nombre": "Juan",
    "_id": {
      "$oid": "5ef8169687dd511442f4e8e0"
    }
  },
  {
    "Edad": "23",
    "Gafas": "si",
    "Apellido": "Garcia"
```

## 6 Llamar un usuario por cédula:

**URL:** [http://localhost:5000/api/user/list/one\\_user/12345678912](http://localhost:5000/api/user/list/one_user/12345678912)

**Nota:** el último campo es la cédula del paciente que se desea consultar

**ESQUEMA:** no es requerido

**MÉTODO:** GET

**RESPUESTA:**

```
[
  {
    "Edad": "33",
    "Gafas": "si",
    "Apellido": "Barrera",
    "Dominante": "Izquierdo",
    "Fecha": "06/27/2020",
    "Sexo": "Masculino",
    "CC": 12345678912,
    "Nombre": "Juan",
    "_id": {
      "$oid": "5ef8169687dd511442f4e8e0"
    }
  }
]
```

## 7. Actualizar contraseña de un usuario:

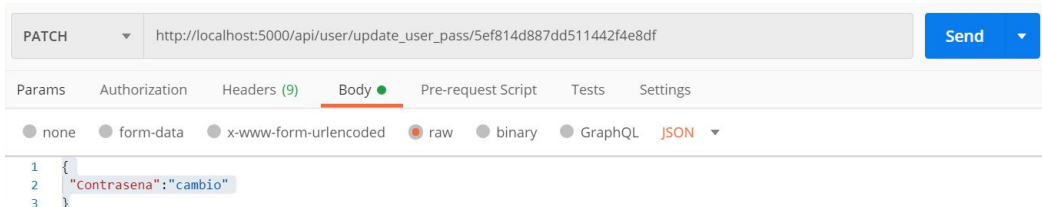
**URL:** [http://localhost:5000/api/user/update\\_user\\_pass/5ef814d887dd511442f4e8df](http://localhost:5000/api/user/update_user_pass/5ef814d887dd511442f4e8df)

**nota:** donde el último campo es el id dado por mongo

**ESQUEMA:**

```
{
  "Contrasena": "cambio"
}
```

**EJEMPLO:**



**MÉTODO:** PATCH

**RESPUESTA:**

```
1 Contraseña modificada
```

## BASE DE DATOS:

```
1  _id:ObjectId("5ef814d887dd511442f4e8df")
2  nombre : "Juan "
3  contrasena : "cambiada "
4  usuario : "usuario "
```

## 8 Eliminar un usuario del sistema

URL:

[http://localhost:5000/api/user/delete\\_user/5ef814d887dd511442f4e8df](http://localhost:5000/api/user/delete_user/5ef814d887dd511442f4e8df)

ESQUEMA: NA

MÉTODO: DELETE

RESPUESTA:

```
1  Usuario Eliminado
```

## BASE DE DATOS:



This collection has no data

It only takes a few seconds to import data from a JSON or CSV file

Import Data

## 9 Eliminar un paciente:

URL: [http://localhost:5000/api/user/delete\\_pacient/12345678912](http://localhost:5000/api/user/delete_pacient/12345678912)

ESQUEMA: NA

MÉTODO: DELETE

BASE DE DATOS:

```
1  _id:ObjectId("5ef816cf87dd511442f4e8e1")
2  +
3  Edad : "23 "
4  Dominante : "Izquierdo "
5  Apellido : "Ospina "
6  CC : 987654321
7  Nombre : "Juana "
8  Fecha : "06/27/2020 "
9  Gafas : "si "
10 Sexo : "Femenino "
```