



ACTIVIDADES POR DESARROLLAR:

- Introducción al BackEnd con Python
- Conocer API-REST con Python.
- Desarrollar API de servicios para FRONTEND.
- Implementar el servicio WEB con FLASK
- Desarrollar una API-REST con Python

EVIDENCIA(S) A ENTREGAR:

EV1 Desarrollar la actividad a desarrollar propuesta en el taller

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	JOSE FERNANDO GALINDO SUAREZ	INSTRUCTOR	CGMLTI	12/01/2022

CONTROL DE CAMBIOS (diligenciar únicamente si realizan ajustes al taller)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					

INTRODUCCIÓN.

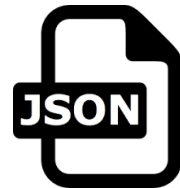
Las API o interfaz de programación de aplicaciones, son un conjunto de funciones que se ofrecen mediante bibliotecas de programación para su uso.

Para entender el concepto de API debemos leer los siguientes términos:

- **Interfaz:** capa de extracción para que las aplicaciones se comuniquen.
- **API:** del inglés Application Programming Interface, que permite compartir datos y procesos entre ellos.
- **Arquitectura de software:** Es la forma como está diseñado un sistema, como se comunican al interior, como están organizados sus componentes.
- **Servicio WEB:** Permite una comunicación entre aplicaciones de una red usando el protocolo HTTP.



- **REST:** Es una arquitectura, del inglés Representational State Transfer o Representación de transferencia de estado, permitiendo que datos puede acceder, revisar o manipular otra aplicación
- **XML:** Formato tradicional para enviar datos, del inglés Extensible Markup Language.
- **JSON:** Formato actual para el intercambio de datos, del inglés JavaScript Object Notation
- **TOKEN:** las API pueden ser públicas o privadas, esta última requiere de una autenticación que al hacerlo te devuelve un token que es un objeto que contiene datos de la autenticación y su formato es JWT.



Tipos de API

- **Locales:** Se ejecutan dentro del mismo entorno. Por ejemplo cuando se desarrolla en Android y se quiere que el celular vibre, entonces debo utilizar la API del smartphone.
- **Remotas:** Cuando se consumen datos que están en otro lugar
 - **Servicios WEB**
 - **SOAP** (Simple Object Access Protocol)
 - **REST** (Representational state transfer) y al utilizarlo lo llamamos RESTFUL



Cada recurso que se consume tiene un identificador único (URI); cuando se consulta un recurso el servidor puede contestar con los siguientes códigos:

- **2XX:** Todo fue exitoso
- **3XX:** Significan redirecciones
- **4XX:** Solicitudes invalidas
- **5XX:** Errores directamente en el servidor

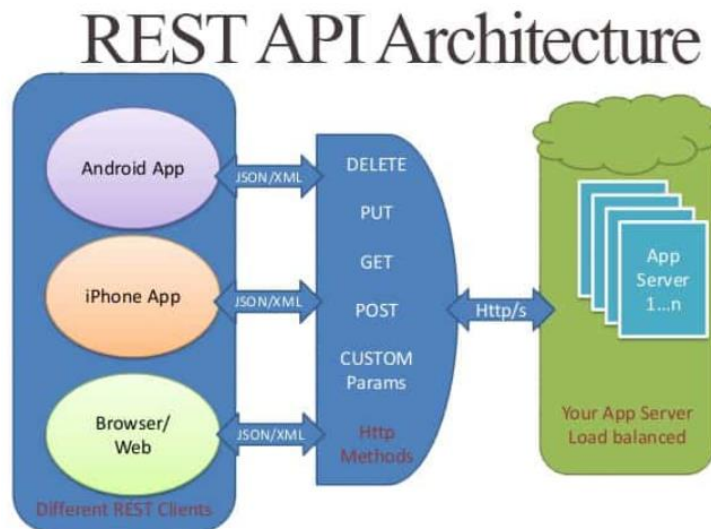
[Fuente](#)

MÉTODOS HTTP



- **GET:** Este método se emplea para leer una representación de un resource. En caso de respuesta positiva (200 OK), devuelve la representación en un formato concreto: HTML, XML, JSON o imágenes, JavaScript, CSS, etc. [Leer mas](#)
- **POST:** Se utiliza POST por las limitaciones de GET. En caso de respuesta positiva devuelve 201 (*created*). Los POST requests se envían normalmente con formularios [Leer mas..](#)
- **PUT:** Utilizado normalmente para actualizar contenidos, pero también pueden crearlos. Tampoco muestra ninguna información en la URL. En caso de éxito devuelve 201 (*created*, en caso de que la acción haya creado un elemento) o 204 (*no response*, si el servidor no devuelve ningún contenido). A diferencia de POST es idempotente, si se crea o edita un resource con PUT y se hace el mismo request otra vez, el resource todavía está ahí y mantiene el mismo estado que en la primera llamada. Si con una llamada PUT se cambia aunque sea sólo un contador en el resource, la llamada ya no es idempotente, ya que se cambian contenidos. [Leer mas...](#)
- **DELETE:** Simplemente elimina un resource identificado en la URI. Si se elimina correctamente devuelve 200 junto con un body response, o 204 sin body. DELETE, al igual que PUT y GET, también es idempotente, [Leer más...](#)

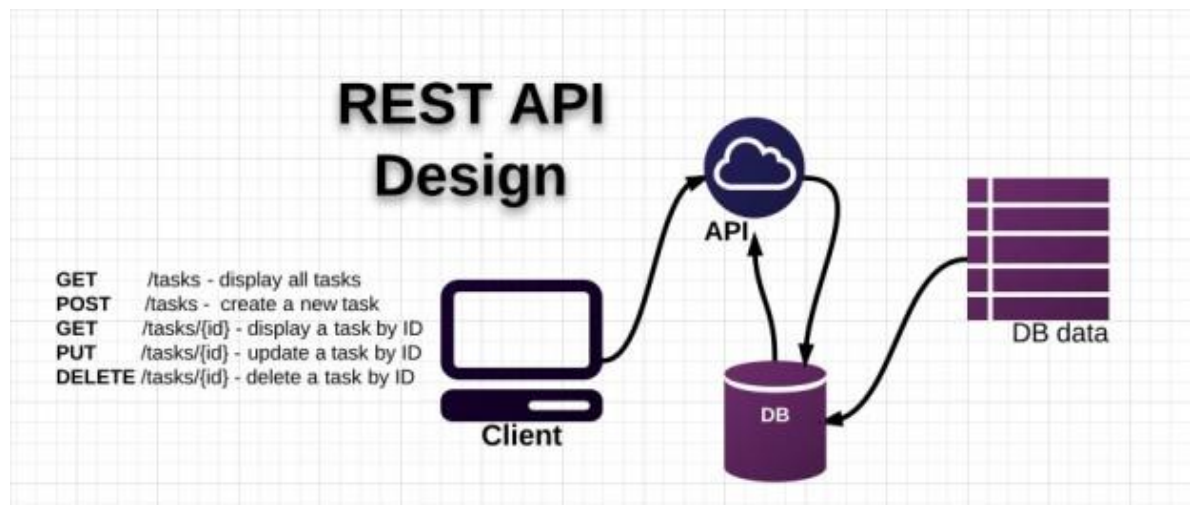
1.API REST EN EL BACKEND.



Fuente: <https://shareurcodes.com/blog/creating%20a%20simple%20rest%20api%20in%20php>

REST API

Crear usuario	POST	/usuarios
Obtener un usuario	GET	/usuarios/{id}
Obtener usuarios	GET	/usuarios
Actualizar usuario	PUT	/usuarios/{id}
Eliminar usuario	DELETE	/usuarios/{id}



Descargar los archivos desde [GITHUB](https://github.com/fegasu/api) para la práctica o utilizando el comando:
“**git clone https://github.com/fegasu/api**”, desde git bash o descargue [GITHUB-DESKTOP](#).

PRACTICA DEL TALLER

Para el desarrollo de ésta practica se debe construir un archivo llamado “servi.py”, que será la API-REST a construir.

Crear la tabla “USUARIO” con “[DB Browse for Sqlite](#)” en la base de datos llamada “init.dat”:

```
1 CREATE TABLE 'USUARIO' (  
2     'idUsuario' INTEGER PRIMARY KEY AUTOINCREMENT,  
3     'login' varchar ( 30 ) UNIQUE,  
4     'nombre' TEXT,  
5     'apellido' TEXT,  
6     'email' TEXT  
7 );
```

INSTALANDO SERVIDOR WEB FLASK

```
C:\Windows\System32\cmd.exe  
F:\SENA\1-ADSI\GUIAS\PYTHON\SENATEST>pip install flask 1  
WARNING: Ignoring invalid distribution -p (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution - (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -p (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution - (c:\python39\lib\site-packages)  
Requirement already satisfied: flask in c:\python39\lib\site-packages (2.0.2)  
Requirement already satisfied: click>=7.1.2 in c:\python39\lib\site-packages (from flask) (8.0.3)  
Requirement already satisfied: Jinja2>=3.0 in c:\python39\lib\site-packages (from flask) (3.0.2)  
Requirement already satisfied: Werkzeug>=2.0 in c:\python39\lib\site-packages (from flask) (2.0.2)  
Requirement already satisfied: itsdangerous>=2.0 in c:\python39\lib\site-packages (from flask) (2.0.1)  
Requirement already satisfied: colorama in c:\python39\lib\site-packages (from click>=7.1.2->flask) (0.4.4)  
Requirement already satisfied: MarkupSafe>=2.0 in c:\python39\lib\site-packages (from Jinja2>=3.0->flask) (2.0.1)  
WARNING: Ignoring invalid distribution -p (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution - (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -p (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution - (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -p (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution - (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -p (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution -ip (c:\python39\lib\site-packages)  
WARNING: Ignoring invalid distribution - (c:\python39\lib\site-packages)  
WARNING: You are using pip version 21.1.3; however, version 21.3.1 is available.  
You should consider upgrading via the 'c:\python39\python.exe -m pip install --upgrade pip' command.
```

INICIANDO EL SERVIDOR WEB FLASK

```
C:\Windows\System32\cmd.exe - flask run  
F:\SENA\1-ADSI\GUIAS\PYTHON\SENATEST>set FLASK_APP=servi.py 1  
F:\SENA\1-ADSI\GUIAS\PYTHON\SENATEST>flask run 2  
* Serving Flask app "servi.py"  
* Environment: production  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

IMPORTADO LAS LIBRERÍA A UTILIZAR

```
1 from flask import Flask,render_template,request
2 import sqlite3
3 import numpy as np
4 import json
```

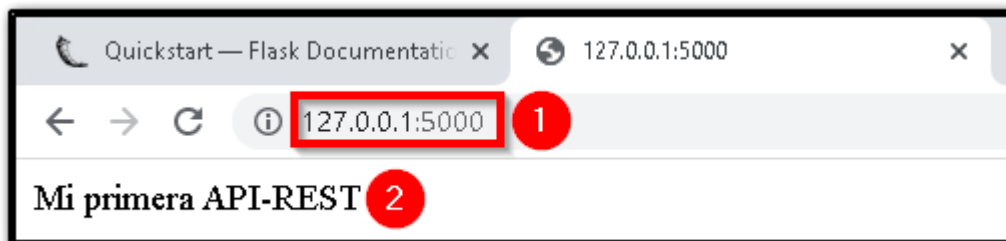
DECLARANDO PAGINA PRINCIPAL EN FLASK

```
app = Flask(__name__) 1
```

CREANDO UNA RUTA RAÍZ

```
@app.route("/") 1
def main(): 2
    return "<p>Mi primera API-REST</p>" 3
```

ENTRANDO A LA PAGINA RAÍZ



CREANDO LOS MÉTODOS DE CONSULTAS DE BASE DE DATOS

```
def EjecutarQuery(bd,sql): 1
    cnx1=sqlite3.connect(bd)
    cursor=cnx1.cursor()
    cursor.execute(sql)
    rows = cursor.fetchall()
    #cnx1.close()
    return json.dumps(rows)

def Ejecutar(bd,sql): 2
    try:
        cnx1=sqlite3.connect(bd)
        cursor=cnx1.cursor()
        cursor.execute(sql)
        cnx1.commit()
        cnx1.close()
        return json.dumps("200")
    except ValueError:
        return sql
```

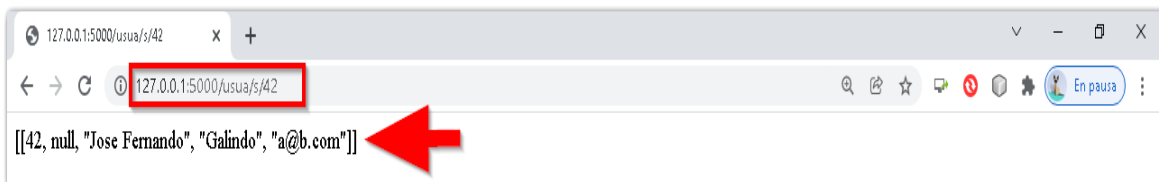
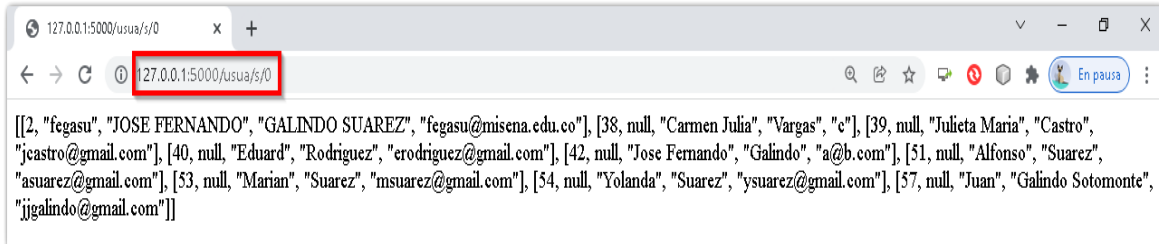
CREANDO LA RUTA PARA CONSULTAR CON EL MÉTODO GET

```
@app.route("/usua/s/<id>", methods=['GET'])
def usuario(id=0):
    global listar
    if id=="0":
        sql='select * from usuario order by 1'
    else:
        sql='select * from usuario where idusuario='+str(id)+' order by 1'
    print(sql)

    listar=EjecutarQuery("init.dat",sql)
    print(listar)
    return listar
```



EJECUTANDO LA CONSULTA





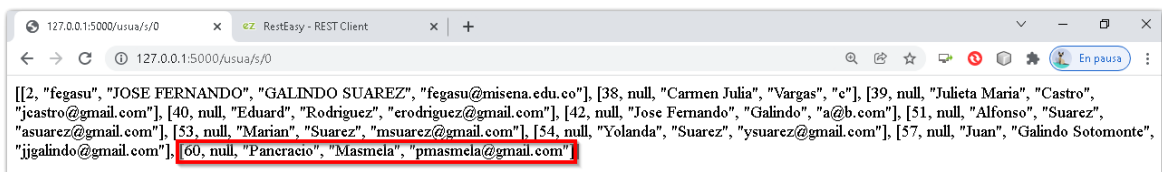
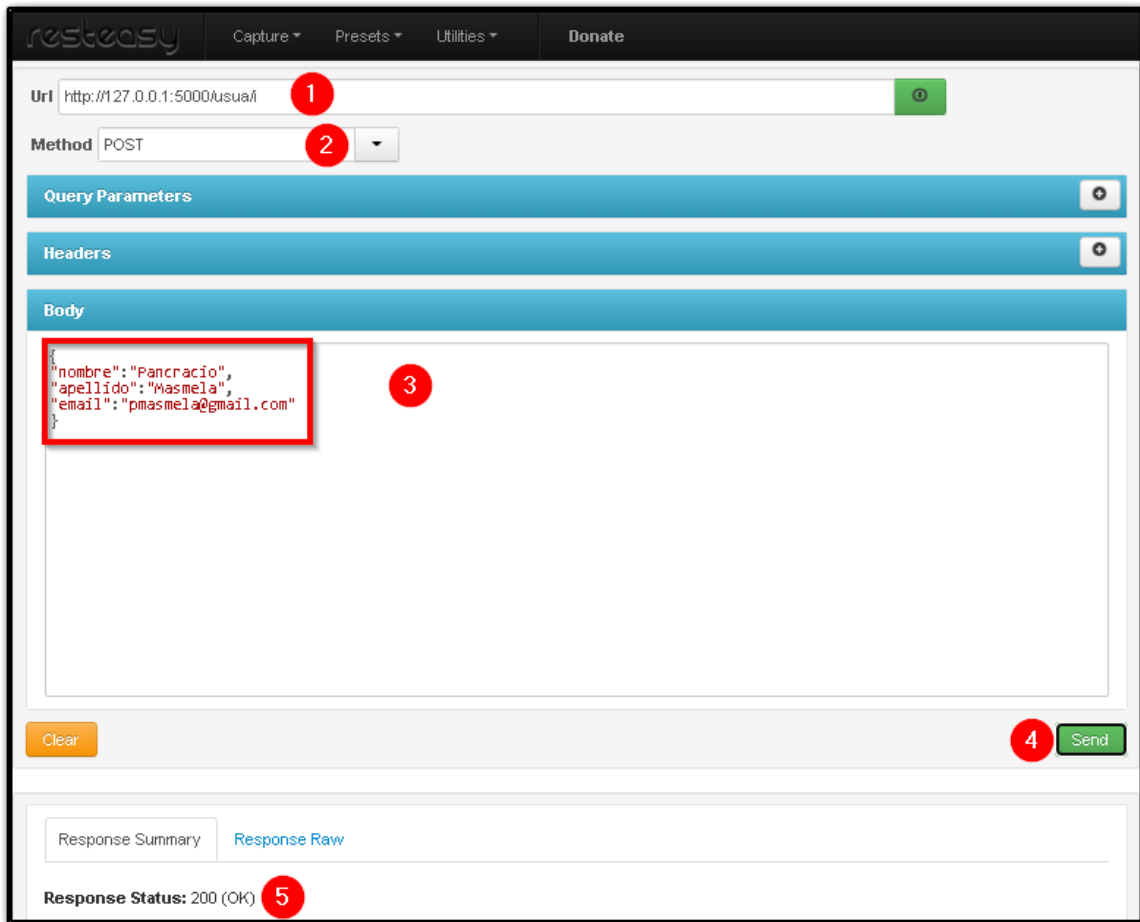
PROBANDO LA API CON EL PROGRAMA RESTEASY CON EL MÉTODO GET

The screenshot shows the Resteasy API client interface. The URL is set to `http://127.0.0.1:5000/usuario/s/0` (1). The method is set to GET (2). The interface includes sections for Query Parameters, Headers, and Body. A 'Send' button (3) is visible. The response status is 200 (OK). The response body (4) is displayed in raw format as a JSON array: `[[{"id": 2, "name": "fegasu", "last_name": "JOSE FERNANDO", "email": "fegasu@misena.edu.co"}, {"id": 38, "name": null, "last_name": "Carmen Julia", "email": "Vargas", "password": "c"}], [{"id": 39, "name": "fegasu", "last_name": "JOSE FERNANDO", "email": "fegasu@misena.edu.co"}]]`.

The screenshot shows the Resteasy API client interface. The URL is set to `http://127.0.0.1:5000/usuario/s/42` (1). The method is set to GET (2). The interface includes sections for Query Parameters, Headers, and Body. A 'Send' button (3) is visible. The response status is 200 (OK). The response body (4) is displayed in raw format as a JSON array: `[[{"id": 42, "name": null, "last_name": "Jose Fernando", "email": "Galindo", "password": "a@b.com"}]]`.

EJECUTANDO EL SERVICIO POST QUE INSERTA UN REGISTRO EN LA BASE DE DATOS

```
@app.route('/usua/i', methods=['POST'])
def iusuario():
    global listar
    json = request.get_json(force=True)
    sql = insert into usuario(nombre,apellido,email) values('"+json["nombre"]+"', '
    print(sql)
    Ejecutar("init.dat",sql)
    return sql
```





EJECUTANDO EL SERVICIO PUT QUE ACTUALIZA UN REGISTRO EN LA BASE DE DATOS

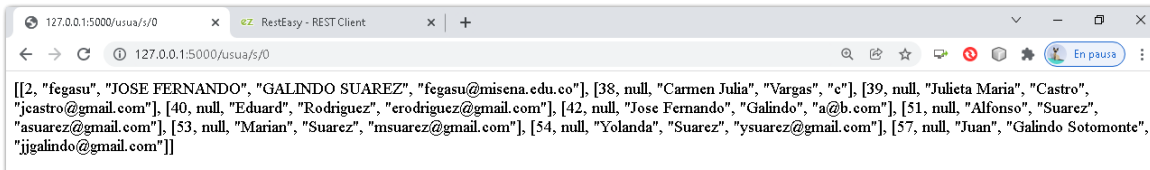
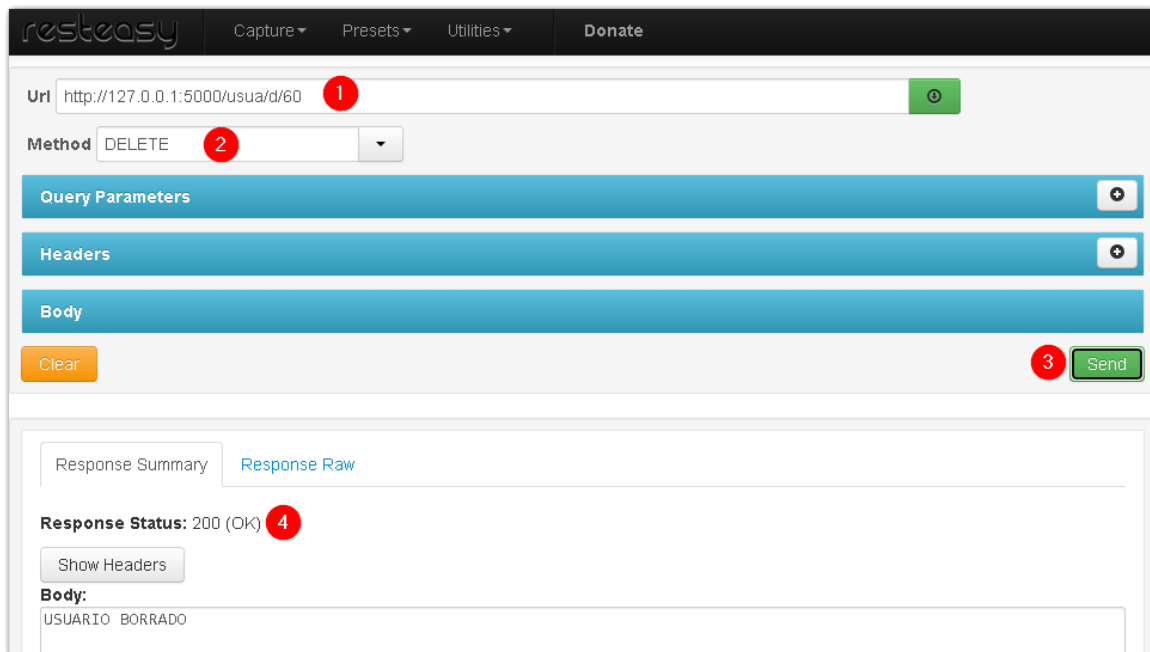
```
@app.route('/usua/u/<id>', methods=['PUT'])
def ausuario(id):
    global listar
    json = request.get_json(force=True)
    sql="update usuario set nombre='"+json["nombre"]+"', apellido='"+json["apellido"]+"', email='"+json["email"]+"' where idusuario="+str(id)
    Ejecutar("init.dat",sql)
    return "200"
```

The screenshot shows the RestEasy REST Client interface. The URL is set to `http://127.0.0.1:5000/usua/u/60` (labeled 1). The Method is set to `PUT` (labeled 2). The Body contains a JSON object: `{ "nombre": "Rosa", "apellido": "Masmela", "email": "pmasmela@gmail.com" }` (labeled 3). The Send button is highlighted (labeled 4). The Response Status is `200 (OK)` (labeled 5).

The screenshot shows a web browser displaying a list of users. The updated record is highlighted in red: `[60, null, "Rosa", "Masmela", "pmasmela@gmail.com"]`.

EJECUTANDO EL SERVICIO DELETE QUE ELIMINA UN REGISTRO EN LA BASE DE DATOS

```
@app.route('/usua/d/<id>', methods=['DELETE'])
def dusuario(id):
    try:
        sql="delete from usuario where idusuario="+str(id)
        Ejecutar("init.dat",sql)
        return "USUARIO BORRADO"
    except ValueError:
        return "ERROR AL BORRAR EL USUARIO"
```



CONSUMIENDO LOS SERVICIOS DE LA API REST CONSTRUIDA EN PYTHON

Construir el archivo “ejemplo.py”

IMPORTANDO LAS LIBRERÍAS NECESARIAS

```
import urllib.request
import numpy as np
import json

import sqlite3
import requests
```

CREANDO UN MÉTODO PARA EJECUTAR POR TIPO DE SERVICIO

```
def EjecutaRest(url, metodo, datos):
    url=url.replace(chr(32), '+') ❶
    try:
        if metodo=="POST": ❷
            miurl=requests.post(url, json=datos)
        if metodo=="PUT": ❸
            miurl=requests.put(url, json=datos)
        if metodo=="DELETE": ❹
            miurl=requests.delete(url)
        if metodo=="GET": ❺
            miurl1=urllib.request.urlopen(url)
            miurl=json.loads(miurl1.read().strip())
        return miurl
    except ValueError:
        messagebox.showerror("Rutinas/EjecutaRest", "Ocurrio un error")
```

OBTENEMOS LOS REGISTROS DE LA BASE DE DATOS

```
def CargaDatos(url):  
    try:  
        url=url.replace(chr(32),'+') 1  
        #print("->" +url)  
        miurl=urllib.request.urlopen(url) 2  
  
        miarreglo=json.loads(miurl.read().strip()) 3  
        miarreglo.reverse() 4  
        return miarreglo 5  
    except:  
        messagebox.showerror("Conectando","Ocurrio un error")
```

UTILIZANDO EL SERVICIO

```
print(CargaDatos('http://127.0.0.1:5000/usua/s/0'))
```

```
<Response [200]>  
[[57, None, 'Juan', 'Galindo Sotomonte', 'jjgalindo@gmail.com'], [54, None, 'Yolanda', 'Suarez',  
'ysuarez@gmail.com'], [53, None, 'Marian', 'Suarez', 'msuarez@gmail.com'], [51, None, 'Alfonso', 'Suarez',  
'asuarez@gmail.com'], [42, None, 'Jose Fernando', 'Galindo', 'a@b.com'], [40, None, 'Eduard', 'Rodriguez',  
'erodriguez@gmail.com'], [39, None, 'Julieta Maria', 'Castro', 'jcastro@gmail.com'], [38, None, 'Carmen Julia',  
'Vargas', 'c'], [2, 'fegasu', 'JOSE FERNANDO', 'GALINDO SUAREZ', 'fegasu@misena.edu.co']]  
[Finished in 2.5s]
```

INSERTANDO UN NUEVO REGISTRO

```
datos={
    "nombre": "Rosa",
    "apellido": "Rico",
    "email": "rrico@gmail.com"
}

print(EjecutaRest('http://127.0.0.1:5000/usua/i', 'POST', datos))
print(CargaDatos('http://127.0.0.1:5000/usua/s/0'))
```

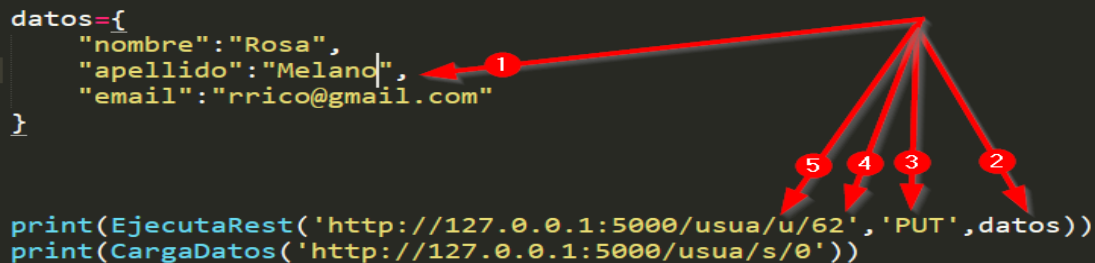


```
<Response [200]>
[[62, None, 'Rosa', 'Rico', 'rrico@gmail.com'], [57, None, 'Juan', 'Galindo Sotomonte', 'jjgalindo@gmail.com'], [
54, None, 'Yolanda', 'Suarez', 'ysuarez@gmail.com'], [53, None, 'Marian', 'Suarez', 'msuarez@gmail.com'], [51,
None, 'Alfonso', 'Suarez', 'asuarez@gmail.com'], [42, None, 'Jose Fernando', 'Galindo', 'a@b.com'], [40, None,
'Eduard', 'Rodriguez', 'erodriguez@gmail.com'], [39, None, 'Julieta Maria', 'Castro', 'jcastro@gmail.com'], [38,
None, 'Carmen Julia', 'Vargas', 'c'], [2, 'fegasu', 'JOSE FERNANDO', 'GALINDO SUAREZ', 'fegasu@misena.edu.co']]
[Finished in 1.9s]
```

ACTUALIZANDO UN REGISTRO DE LA BASE DE DATOS

```
datos={
    "nombre": "Rosa",
    "apellido": "Melano",
    "email": "rrico@gmail.com"
}

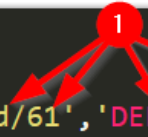
print(EjecutaRest('http://127.0.0.1:5000/usua/u/62', 'PUT', datos))
print(CargaDatos('http://127.0.0.1:5000/usua/s/0'))
```



```
<Response [200]>
[[62, None, 'Rosa', 'Melano', 'rrico@gmail.com'], [57, None, 'Juan', 'Galindo Sotomonte', 'jjgalindo@gmail.com'], [
54, None, 'Yolanda', 'Suarez', 'ysuarez@gmail.com'], [53, None, 'Marian', 'Suarez', 'msuarez@gmail.com'], [51,
None, 'Alfonso', 'Suarez', 'asuarez@gmail.com'], [42, None, 'Jose Fernando', 'Galindo', 'a@b.com'], [40, None,
'Eduard', 'Rodriguez', 'erodriguez@gmail.com'], [39, None, 'Julieta Maria', 'Castro', 'jcastro@gmail.com'], [38,
None, 'Carmen Julia', 'Vargas', 'c'], [2, 'fegasu', 'JOSE FERNANDO', 'GALINDO SUAREZ', 'fegasu@misena.edu.co']]
[Finished in 1.8s]
```



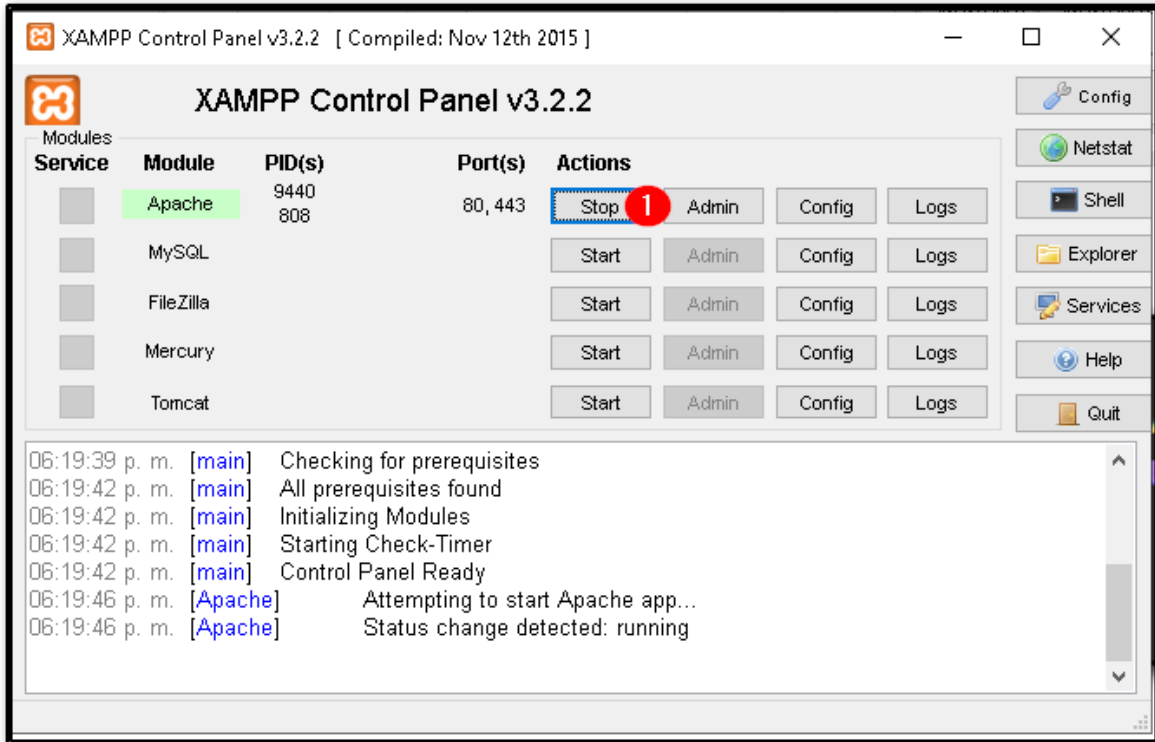
BORRANDO UN REGISTRO DE LA BASE DE DATOS



```
print(EjecutaRest('http://127.0.0.1:5000/usua/d/61', 'DELETE', None))
print(CargaDatos('http://127.0.0.1:5000/usua/s/0'))
```

```
<Response [200]>
[[57, None, 'Juan', 'Galindo Sotomonte', 'jjgalindo@gmail.com'], [54, None, 'Yolanda', 'Suarez',
'ysuarez@gmail.com'], [53, None, 'Marian', 'Suarez', 'msuarez@gmail.com'], [51, None, 'Alfonso', 'Suarez',
'asuarez@gmail.com'], [42, None, 'Jose Fernando', 'Galindo', 'a@b.com'], [40, None, 'Eduard', 'Rodriguez',
'erodriguez@gmail.com'], [39, None, 'Julieta Maria', 'Castro', 'jcastro@gmail.com'], [38, None, 'Carmen Julia',
'Vargas', 'c'], [2, 'fegasu', 'JOSE FERNANDO', 'GALINDO SUAREZ', 'fegasu@misena.edu.co']]
[Finished in 2.5s]
```


CONSUMIENDO SERVICIO DE LA API-REST DESDE PHP



Crear en htdocs la carpeta “ApiRest” y construya el archivo “ejemplo.php”

```
<?php
$url='http://127.0.0.1:5000/usua/s/0';
$JSON = file_get_contents($url,true);
echo $JSON;
?>
```

Ejecute desde el navegador.





Realizado por el instructor José Fernando Galindo Suárez

jgalindos@sena.edu.co 2022



<https://stackoverflow.com/questions/221442/how-do-you-create-a-rest-client-for-java>

<https://oracle-max.com/como-consumir-un-api-restful-usando-el-metodo-post-en-java/>

<https://spring.io/guides/gs/consuming-rest/>

<https://www.it-swarm-es.com/es/java/como-consumir-rest-en-java/1069077733/>

<https://es.stackoverflow.com/questions/330558/c%C3%B3mo-consumir-api-rest-en-java-desktop>

<https://es.stackoverflow.com/questions/230662/invocar-una-llamada-http-desde-una-aplicaci%C3%B3n-java>

<https://www.it-swarm-es.com/es/java/java-como-hacer-una-llamada-api-con-datos/834546114/>



<https://www.ibm.com/docs/es/rtw/9.1.0?topic=services-example-calling-rest-api-java-http>