# Documentación API Rest en Python

## Requisitos técnicos

Necesarios:

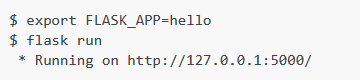
* Python 3
* Flask (pip install flask)

Aconsejables:

* Postman
* DB Browser

## Iniciar el servicio

Es necesario abrir un terminal y situarse en la carpeta del proyecto. Si estamos trabajando en Linux ejecutamos:



Cambiando “hello” por “api”. Si es en Windows, escribimos en el cmd análogamente:



Si por cualquier motivo es necesario restablecer la información de la base de datos, basta borrar el fichero “database.db” y crear un fichero nuevo vacío con la mismo nombre y extensión. Acto seguido, ejecuta el script “script.py”.

## Endpoint */*

Devuelve un array formado por dos array:

* Uno de arquitectos, donde cada elemento es un JSON que contiene todos los datos de un arquitecto en concreto.
* Otro de edificios, donde cada elemento es un JSON que contiene todos los datos de un edificio en concreto.

## Endpoint */Edificio*

Devuelve un array de JSON donde cada JSON es un registro completo en la base de datos de un edificio.

## Endpoint */Arquitecto*

Devuelve un array de JSON donde cada JSON es un registro completo en la base de datos de un arquitecto.

## Endpoint */Edificio/eliminar/<id>*

Escribiendo en la url un valor para *<id>* elimina el registro cuyo id en la base de datos se corresponde con el indicado.

Si el borrado es exitoso redirige a la raíz.

## Endpoint */Arquitecto/eliminar/<arquitecto>*

Escribiendo en la url un valor para *<arquitecto>* elimina el registro de aquel arquitecto se llame tal y como se indique.

Si el borrado es exitoso redirige a la raíz.

## Endpoint */Edificio/buscar/<denominacion>*

Escribiendo en la url un valor para *<denominacion>* devuelve un array de JSON con los registros de la base de datos que cumplen que *<denominacion>* esta contenido en el valor de ese atributo.

## Endpoint */Arquitecto/buscar/<arquitecto>*

Escribiendo en la url un valor para *<arquitecto>* devuelve un array de JSON con los registros de la base de datos que cumplen que *<arquitecto>* esta contenido en la clave primaria.

## Endpoint */Edificio/modificar*

* Si hacemos una petición por GET, es necesario pasar como argumento el id del edificio a modificar. La función devuelve un JSON con los datos que se tienen de ese edificio. Ej: *Edificio/modificar?id=4038*
* Si hacemos una petición por POST, se debe enviar un JSON con los atributos a modificar y los nuevos valores a establecer. Al realizar la actualización satisfactoriamente redirige a la raíz.

# Endpoint */Arquitecto/modificar*

* Si hacemos una petición por GET, es necesario pasar como argumento el “arquitecto” (nombre) del arquitecto a modificar. La función devuelve un JSON con los datos que se tienen de ese arquitecto.
* Si hacemos una petición por POST, se debe enviar un JSON con los atributos a modificar y los nuevos valores a establecer. Al realizar la actualización satisfactoriamente redirige a la raíz.

## Endpoint */Arquitecto/modificar*

Al hacer una petición, en este caso por POST, se debe enviar un JSON con los atributos y sus valores a insertar. Al realizar la inserción adecuadamente redirige al index.

## Endpoint */Edificio/modificar*

Al hacer una petición, en este caso por POST,, se debe enviar un JSON con los atributos y sus valores a insertar. Al realizar la inserción adecuadamente redirige al index.

## A tener en cuenta:

* Notación usada para los atributos de arquitectos y edificios, de cara a establecer los pares clave : valor de los JSON:
  + Arquitecto:
    - “arquitecto”
    - “num\_colegio”
  + Edificio:
    - “id”
    - “denominacion”
    - “otrasDenominaciones”
    - “categorias”
    - “tipologia”
    - “centro”
    - “ubicaciónActual”
    - “acceso”
    - “formaDeIngreso”
    - “procedencia”
    - “volumen”
    - “cronologia”
    - “autores”
    - “descripcion”
    - “historia”
    - “objetoDocumento”
    - “tecnicas”
    - “signatura”
    - “exposicion”
    - “thumbnail”