# PETICIONES PYTHON A SERVICIOS API

Un Servicio Api es un origen de datos, pero en la nube.

Nos permite poder acceder a datos desde múltiples aplicaciones y dispositivos ya que no comunicamos con una base de datos directamente, sino que comunicamos a través del servicio.

En realidad son poco pesados y muy óptimos para dispositivos que no tengan muchas memoria.

Son formatos de tipo texto que representan datos. Existen varios formatos, pero el más extendido es el formato JSON.

Api con ejemplos:

<https://apiejemplos.azurewebsites.net/index.html>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

GET: Recuperar datos

POST: Insertar datos

PUT: Modificar datos

DELETE: Eliminar datos

Las peticiones se realizan en la URL (algunas)

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para utilizar servicios desde Python, necesitamos instalar **requests**

Para consumir datos o enviar datos al servicio, debemos hacerlo mediante diccionarios.

Objeto[“CLAVE”]

pip install requests

Vamos a consumir el siguiente servicio.

https://apiempleadosspgs.azurewebsites.net/

Nos devuelve los empleados de una base de datos.

Texto

Descripción generada automáticamente

Comenzamos creando un nuevo programa llamado **python47apiempleados.py**

**PYTHON47APIEMPLEADOS.PY**

**import requests**

**import json**

**print("Consumo Api empleados")**

**url = "https://apiempleadosspgs.azurewebsites.net/"**

**print("1.- Todos los empleados")**

**print("2.- Buscar empleado")**

**print("Seleccione una opción")**

**opcion = *int*(input())**

**if (opcion == 1):**

**peticion = "api/empleados"**

**# REALIZAMOS UNA PETICION GET Y CAPTURAMOS LA RESPUESTA**

**response = requests.get(url + peticion)**

**# CONVERTIMOS LA RESPUESTA A DICCIONARIO JSON**

**empleados = response.json()**

**# UN DICCIONARIO PODEMOS RECORRERLO SI ES UN CONJUNTO**

**print("Listado de empleados Api")**

**for emp in empleados:**

**print(emp["apellido"])**

**elif (opcion == 2):**

**print("Introduzca ID empleado")**

**idempleado = input()**

**peticion = "api/empleados/" + idempleado**

**response = requests.get(url + peticion)**

**empleado = response.json()**

**print(empleado["apellido"] + ", " + empleado["oficio"] + ", " + *str*(empleado["salario"]))**

**else:**

**print("Opción incorrecta")**

**print("Fin de programa")**

El siguiente paso que vamos a realizar es aprender las consultas de acción.

Consumiremos el siguiente servicio:

https://apicruddepartamentoscore.azurewebsites.net

Lo que necesitamos es pedir datos y transformarlos a JSON, ya que el método POST lo que hace es recibir un objeto en formato JSON

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Creamos un nuevo programa llamado **python48apicruddepartamentos.py**

Texto

Descripción generada automáticamente

**import requests**

**import json**

**print("Ejemplo Api Crud Departamentos")**

**url = "https://apicruddepartamentoscore.azurewebsites.net/"**

**print("1.- Mostrar departamentos")**

**print("2.- Insertar departamento")**

**print("3.- Modificar departamento")**

**print("4.- Eliminar departamento")**

**print("Seleccione una opción")**

**opcion = *int*(input())**

**if (opcion == 1):**

**peticion = "api/departamentos"**

**response = requests.get(url + peticion)**

**departamentos = response.json()**

**for dept in departamentos:**

**print(*str*(dept["numero"]) + ", " + dept["nombre"] + ", " + dept["localidad"])**

**elif (opcion == 2):**

**print("-----------Nuevo departamento--------------")**

**print("Introduzca ID departamento")**

**numero = *int*(input())**

**print("Introduzca nombre departamento")**

**nombre = input()**

**print("Introduzca una localidad")**

**loc = input()**

**peticion = "api/departamentos"**

**# CREAMOS UN OBJETO JSON DE TIPO DICCIONARIO**

**departamento = {**

**"numero": numero,**

**"nombre": nombre,**

**"localidad": loc**

**}**

**print(departamento)**

**response = requests.post(url + peticion, *json*=departamento)**

**# DIBUJAMOS LA RESPUESTA DEL SERVICIO**

**print("Status: " + *str*(response.*status\_code*))**

**elif (opcion == 3):**

**peticion = "api/departamentos"**

**print("---------Modificar departamento----------")**

**print("Introduzca ID departamento a modificar")**

**numero = *int*(input())**

**print("Introduzca nuevo nombre departamento")**

**nombre = input()**

**print("Introduzca nueva localidad")**

**loc = input()**

**departamento = {**

**"numero": numero,**

**"nombre": nombre,**

**"localidad": loc**

**}**

**response = requests.put(url + peticion, *json*=departamento)**

**print("Status: " + *str*(response.*status\_code*))**

**elif (opcion == 4):**

**print("------------Eliminar departamento----------")**

**print("Introduzca ID departamento a eliminar")**

**numero = input()**

**peticion = "api/departamentos/" + numero**

**response = requests.delete(url + peticion)**

**print("Status: " + *str*(response.*status\_code*))**

**else:**

**print("Opción incorrecta")**

**print("Fin de programa")**

Vamos a crear un nuevo proyecto de Django para realizar un CRUD de Departamentos

Tabla

El contenido generado por IA puede ser incorrecto.

Creamos un nuevo proyecto llamado **proyectoapi**

**django-admin startproject proyectoapi**

Entramos en la carpeta y creamos un nuevo proyecto llamado departamentos

**python manage.py startapp departamentos**

Instalamos la librería para poder consumir APIS

**pip install requests**

Agregamos **urls.py** sobre **departamentos**

Creamos una carpeta llamada **templates** y dentro una carpeta llamada **pages** y en su interior una página llamada **index.html**

Una vez que tenemos montado el proyecto, vamos a configurar las peticiones para llamar a un API.

Un api siempre tendrá la misma URL base. Lo que cambiará será la petición (request) y el tipo de método.

En models creamos una nueva clase llamada **departamento** y otra clase llamada **ServiceDepartamentos**

**MODELS.PY**

**from django.db import models**

**import requests**

**# Create your models here.**

***class* Departamento:**

**numero = 0**

**nombre = ""**

**localidad = ""**

***class* ServiceDepartamentos:**

***def* \_\_init\_\_(*self*):**

***self*.*url* = "https://apiejemplos.azurewebsites.net/"**

***def* getDepartamentos(*self*):**

**peticion = "api/departamentos"**

**response = requests.get(*self*.*url* + peticion)**

**json = response.json()**

**lista = []**

**for dato in json:**

**dept = Departamento()**

**dept.*numero* = dato["idDepartamento"]**

**dept.*nombre* = dato["nombre"]**

**dept.*localidad* = dato["localidad"]**

**lista.append(dept)**

**return lista**

Recuperamos los datos de los departamentos dentro de **views.py** y del método **index**

**VIEWS.PY**

**from django.shortcuts import render**

**from departamentos.models import ServiceDepartamentos**

**# Create your views here.**

***def* index(*request*):**

**servicio = ServiceDepartamentos()**

**departamentos = servicio.getDepartamentos()**

**context = {**

**"departamentos": departamentos**

**}**

**return render(request, 'pages/index.html', context)**

Dibujamos los departamentos en una tabla en **index.html**

**INDEX.HTML**

**<!DOCTYPE html>**

**<html lang*="en"*>**

**<head>**

**<meta charset*="UTF-8"*>**

**<meta name*="viewport"* content*="width=device-width, initial-scale=1.0"*>**

**<title>Document</title>**

**<link href*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"* rel*="stylesheet"* integrity*="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"* crossorigin*="anonymous"*>**

**</head>**

**<body>**

**{% include 'pages/menu.html' %}**

**<h1>Proyecto Api departamentos</h1>**

**<table class*="table table-bordered table-info"*>**

**<tr>**

**<th>Número</th>**

**<th>Nombre</th>**

**<th>Localidad</th>**

**</tr>**

**{% for dept in departamentos%}**

**<tr>**

**<td>{{dept.numero}}</td>**

**<td>{{dept.nombre}}</td>**

**<td>{{dept.localidad}}</td>**

**</tr>**

**{% endfor %}**

**</table>**

**<script src*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"* integrity*="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"* crossorigin*="anonymous"*></script>**

**</body>**

**</html>**

A continuación, vamos a insertar un departamento, escribimos un método dentro del Servicio **models.py**

Texto

El contenido generado por IA puede ser incorrecto.

Sobre **templates/pages** creamos una página llamada **create.html**

**CREATE.HTML**

**<!DOCTYPE html>**

**<html lang*="en"*>**

**<head>**

**<meta charset*="UTF-8"*>**

**<meta name*="viewport"* content*="width=device-width, initial-scale=1.0"*>**

**<title>Document</title>**

**<link href*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"* rel*="stylesheet"* integrity*="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"* crossorigin*="anonymous"*>**

**</head>**

**<body>**

**{% include 'pages/menu.html' %}**

**<h1>Create</h1>**

**<form method*="post"*>**

**{% csrf\_token %}**

**<label>Id</label>**

**<input type*="number"* name*="cajaid"* class*="form-control"*/>**

**<label>Nombre</label>**

**<input type*="text"* name*="cajanombre"* class*="form-control"*/>**

**<label>Localidad</label>**

**<input type*="text"* name*="cajalocalidad"* class*="form-control"*/>**

**<button class*="btn btn-outline-warning"*>**

**Insertar**

**</button>**

**</form>**

**<script src*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"* integrity*="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"* crossorigin*="anonymous"*></script>**

**</body>**

**</html>**

Sobre **views.py** creamos un nuevo método para insertar departamentos.

**VIEWS.PY**

***def* insertarDepartamento(*request*):**

**if ('cajaid' in request.POST):**

**servicio = ServiceDepartamentos()**

**id = request.POST['cajaid']**

**nombre = request.POST['cajanombre']**

**localidad = request.POST['cajalocalidad']**

**servicio.insertarDepartamento(id, nombre, localidad)**

**departamentos = servicio.getDepartamentos()**

**context = {**

**"departamentos": departamentos**

**}**

**return render(request, 'pages/index.html', context)**

**else:**

**return render(request, 'pages/create.html')**

Sobre **urls.py** creamos una nueva ruta.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

El siguiente paso será crear la funcionalidad para eliminar.

Comenzamos dibujando un nuevo Link dentro de **index.html**

**INDEX.HTML**

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

Sobre **models.py** creamos un nuevo método para eliminar un departamento

**MODELS.PY**

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Creamos un nuevo método en **views.py** que eliminará a un departamento por su ID y que nos devolverá a la página index después de eliminar

**VIEWS.PY**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Incluimos dentro de **urls.py** la ruta para eliminar

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

El último paso para el CRUD es realizar un Update

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Lo primero será tener un método para buscar departamento y otro para modificar un departamento dentro de **models.py**

**MODELS.PY**

***def* findDepartamento(*self*, *numero*):**

**peticion = "api/departamentos/" + numero**

**response = requests.get(*self*.*url* + peticion)**

**json = response.json()**

**dept = Departamento()**

**dept.*numero* = json['idDepartamento']**

**dept.*nombre* = json["nombre"]**

**dept.*localidad* = json['localidad']**

**return dept**

***def* update(*self*, *numero*, *nombre*, *localidad*):**

**peticion = "api/departamentos"**

**deptjson = {**

**"idDepartamento": numero,**

**"nombre": nombre,**

**"localidad": localidad**

**}**

**requests.put(*self*.*url* + peticion, *json*=deptjson)**

El siguiente paso es crear un método dentro de **views.py**

Si recibimos una caja con POST, hacemos el UPDATE, si recibimos un ID con GET, devolvemos el departamento al dibujo.

**VIEWS.PY**

***def* modificar(*request*):**

**servicio = ServiceDepartamentos()**

**if ('cajaid' in request.POST):**

**#UPDATE**

**id = request.POST['cajaid']**

**nombre = request.POST['cajanombre']**

**localidad = request.POST['cajalocalidad']**

**servicio.update(id, nombre, localidad)**

**departamentos = servicio.getDepartamentos()**

**context= {**

**"departamentos":departamentos**

**}**

**return render(request, 'pages/index.html', context)**

**elif ('id' in request.GET):**

**#BUSCAR**

**id = request.GET['id']**

**dept = servicio.findDepartamento(id)**

**context = {**

**"departamento": dept**

**}**

**return render(request, 'pages/update.html', context)**

Sobre **templates/pages** agregamos una nueva página llamada **update.html**

**UPDATE.HTML**

**<!DOCTYPE html>**

**<html lang*="en"*>**

**<head>**

**<meta charset*="UTF-8"*>**

**<meta name*="viewport"* content*="width=device-width, initial-scale=1.0"*>**

**<title>Document</title>**

**<link href*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"* rel*="stylesheet"* integrity*="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"* crossorigin*="anonymous"*>**

**</head>**

**<body>**

**{% include 'pages/menu.html' %}**

**<h1>Edit</h1>**

**<form method*="post"*>**

**{% csrf\_token %}**

**<input type*="hidden"* name*="cajaid"***

**value*="{{*departamento*.*numero*}}"*/>**

**<label>Nombre</label>**

**<input type*="text"* name*="cajanombre"* class*="form-control"***

**value*="{{*departamento*.*nombre*}}"*/>**

**<label>Localidad</label>**

**<input type*="text"* name*="cajalocalidad"* class*="form-control"***

**value*="{{*departamento*.*localidad*}}"*/>**

**<button class*="btn btn-outline-info"*>**

**Update**

**</button>**

**</form>**

**<script src*="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"* integrity*="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"* crossorigin*="anonymous"*></script>**

**</body>**

**</html>**

Sobre **urls.py** creamos una nueva ruta para update

**URLS.PY**

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

Sobre **index.html** poner un Link enviando la información del ID del departamento a modificar

**INDEX.HTML**

Escala de tiempo

El contenido generado por IA puede ser incorrecto.