# Instalación de un sevidor backend Python

## Flask + Jinja2

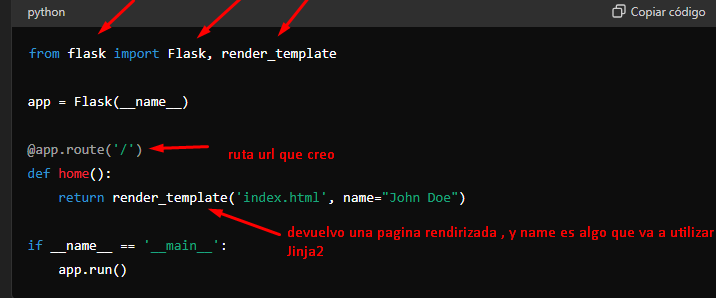
### Flask

Flask provee el soporte para el backend en Python para aplicaciones frontend que consuman servicios. Es una versión simplificado de Django.

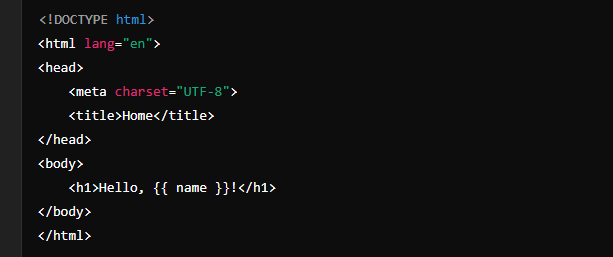
1. **APIs RESTful**: Flask es una excelente opción para desarrollar APIs RESTful debido a su flexibilidad y simplicidad. Permite definir rutas y manejar solicitudes HTTP fácilmente, facilitando la creación de servicios web que pueden ser consumidos por aplicaciones móviles, frontends basados en JavaScript, y otros servicios.
2. **Aplicaciones de Backend para Frontends SPA**: Flask se puede usar para desarrollar el backend de aplicaciones web de una sola página (Single Page Applications - SPA) que usan frameworks de frontend como React, Angular, o Vue.js.
3. **Aplicaciones de Datos y Ciencia de Datos**: Flask se puede utilizar para crear dashboards interactivos y aplicaciones web que muestran análisis de datos, visualizaciones y modelos predictivos. Es comúnmente utilizado en combinación con bibliotecas como Pandas, NumPy, y Matplotlib.

Url Flask https://flask.palletsprojects.com/en/3.0.x/

Para instalar flask usamos pip, que en como el npm de node. Pip ya viene instalado con Python.



Jinja2 lo que hace es que puedo poner datos de manera dinámica como lo hace vue, php etc, para eso enriquece el html con nuevos tags



### Características de Jinja2

1. **Variables**: Puedes insertar variables en las plantillas usando {{ variable }}.
2. **Control de Flujo**: Puedes usar estructuras de control como for y if dentro de las plantillas.

html

Copiar código

<ul>

{% for item in items %}

<li>{{ item }}</li>

{% endfor %}

</ul>

1. **Filtros**: Aplicar filtros a las variables para modificar su salida.

html

Copiar código

{{ name | upper }}

1. **Macros**: Definir macros reutilizables dentro de las plantillas.

html

Copiar código

{% macro input(name, value='') %}

<input type="text" name="{{ name }}" value="{{ value }}">

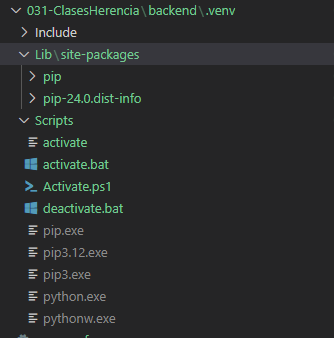
{% endmacro %}

{{ input('username') }}

### Pasos

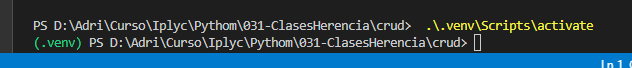
Instalar un entorno virtual, armo un directorio proyecto y ejecuto

* + 1. **python -m venv .venv**

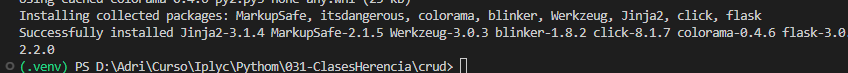


Activo el ambiente voy a la carpeta y ejecuto

* + 1. **.\.venv\Scripts\activate**



* + 1. **Pip install falsk**



**Me instalo flask y otras dependencias ej Jinja2**

**Pip list** -> dependencias que instalo

**Atencion** para GIT tengo que sacar la lib, dado que eso se baja

Creo un directorio app (nombre opcional) y genero un app.py (nombre opcional)

**from** flask **import** Flask

app = Flask**(**\_\_name\_\_**)**

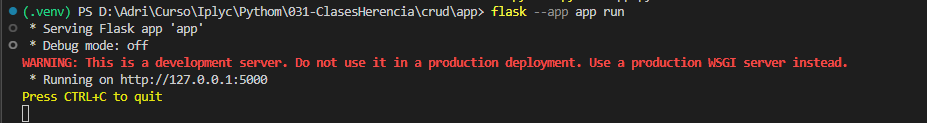
@app.route**(**"/"**)**

**def** hello\_world**():**

**return** "<p>Hello, World!</p>"

**ejecuto** dentro del entorno virtual el archivo

**flask –app “nombrearchivopy” run --debug**



<https://flask.palletsprojects.com/en/3.0.x/quickstart/>

#### Externally Visible Server

If you run the server you will notice that the server is only accessible from your own computer, not from any other in the network. This is the default because in debugging mode a user of the application can execute arbitrary Python code on your computer.

If you have the debugger disabled or trust the users on your network, you can make the server publicly available simply by adding --host=0.0.0.0 to the command line:

$ flask run --host=0.0.0.0

This tells your operating system to listen on all public IPs.

TIPs: Servidor para postear una prueba <https://www.pythonanywhere.com/>

Rendirizado

To render a template you can use the **[render\_template()](https://flask.palletsprojects.com/en/3.0.x/api/" \l "flask.render_template" \o "flask.render_template)** method. All you have to do is provide the name of the template and the variables you want to pass to the template engine as keyword arguments. Here’s a simple example of how to render a template:

Flask will look for templates in the templates folder. So if your application is a module, this folder is next to that module, if it’s a package it’s actually inside your package: