



BACKEND CON PYTHON



MVC Y FLASK

- Patrón MVC
- Despliegue local de Flask
- Plantillas y Jinja2



MVC Y FLASK

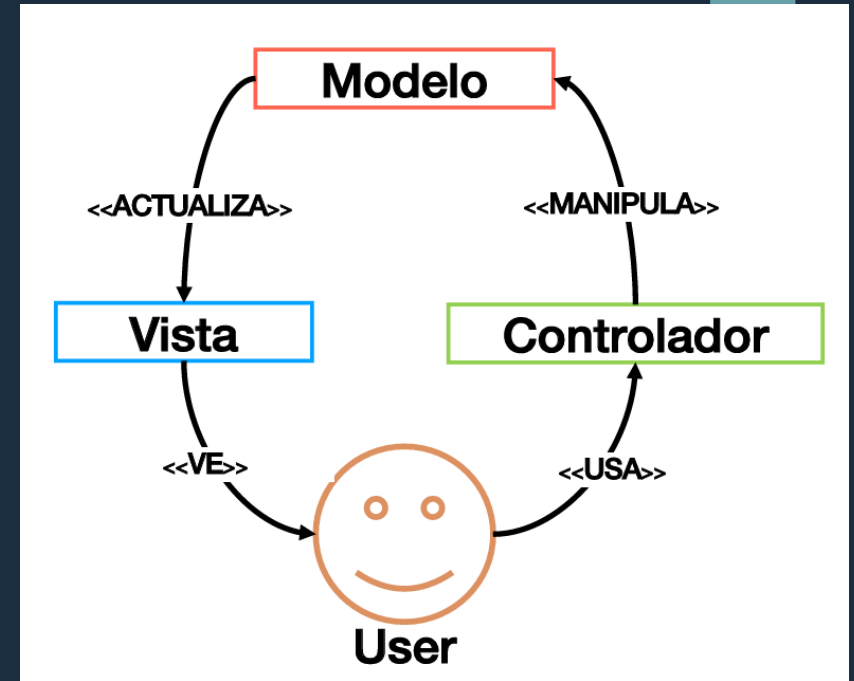
- Patrón MVC
- Despliegue local de Flask
- Plantillas y Jinja2



PATRON MVC

El patrón MVC (Modelo-Vista-Controlador) es una arquitectura que se usa en el desarrollo de aplicaciones para separar y organizar las responsabilidades de una manera modular y escalable.

- **Modelo:** elementos que tienen la información y el estado que se maneja en el programa. También encapsula la lógica del problema
- **Vista:** elementos que se encargan de mostrarle al usuario información sobre el modelo. Son pasivas (no reaccionan a las acciones del usuario) pero se actualizan si hay cambios en el modelo.
- **Controlador:** elementos que reacciones a las acciones y comandos del usuario, con esto producen cambios en el modelo.



PATRON MVC

Modelo

Contiene todos los elementos de la lógica del negocio, estructuras de datos, información del modelo. Es responsable de manejar el estado de la aplicación y de realizar operaciones para manipular la información.

Ventajas:

- Reutilización: al estar separada la lógica del negocio, se puede reutilizar en diferentes ambientes.
- Mantenibilidad: los cambios en la lógica pueden realizarse sin afectar la Interfaz de Usuario (UI).

PATRON MVC

Vista

Contiene todos los elementos gráficos que le muestran al usuario la información de la aplicación. Se actualizan cuando hay cambios en el modelo.

Ventajas:

- Bajo acoplamiento: Los cambios en la interfaz de usuario (UI) no afectan al modelo y viceversa
- Facilidad en el diseño: el diseñador puede trabajar en paralelo en las vistas, mientras que el desarrollador trabaja en el modelo, no habrá conflictos de versiones.

PATRON MVC

Controlador

Actúa como intermediario entre el Modelo y la Vista: recibe las acciones y comando del usuario desde la Vista, usa la lógica del negocio que está en el Modelo y muestra la respuesta nuevamente en la Vista.

Ventajas:

- Reutilización: puede haber varios controladores en diferentes parte de la aplicación.

PATRON MVC

MVC en Flask

Flask tiene la ventaja que es un Framework sumamente flexible, por lo que, no es obligatorio seguir la arquitectura MVC, como si es lo es en Django.

Sin embargo, se pueden aplicar los principios de MVC a un proyecto en Flask para organizar los directorios y separar las responsabilidades como lo hace MVC:

```
|-- /models  
|   |-- guarderia.py  
|   |-- animal.py  
|   |-- perro.py
```

Modelo: se puede crear un directorio models, que contenga los archivos .py con la lógica del programa.

```
|-- /templates  
|   |-- index.html  
|   |-- info_perro.html
```

Vista: se puede crear un directorio templates, que contenga los archivos HTML con la interfaz de usuario.

```
|-- /views  
|   |-- user_views.py  
|   |-- other_views.py
```

Controlador: se puede crear un directorio views, las vistas serán los controladores..

PATRON MVC

MVC en Flask

Acá se puede ver una estructura general donde /myapp es la carpeta raíz del proyecto flask.

- /models es el directorio con el modelo y contiene la lógica en Python.
- /templates es el directorio con las vistas en HTML, CSS y JS.
- /views es el directorio con los controladores en Python.*
- app.py es el archivo que tiene las instrucciones para desplegar la aplicación en Flask

*El directorio “views” no se traduce como “Vistas”, acá van los Controladores.

```
/myapp
|-- /models
|   |-- guarderia.py
|   |-- animal.py
|   |-- perro.py
|-- /templates
|   |-- index.html
|   |-- info_perro.html
|-- /views
|   |-- user_views.py
|   |-- other_views.py
|-- app.py
```

MVC Y FLASK

- Patrón MVC
- Despliegue local de Flask
- Plantillas y Jinja2



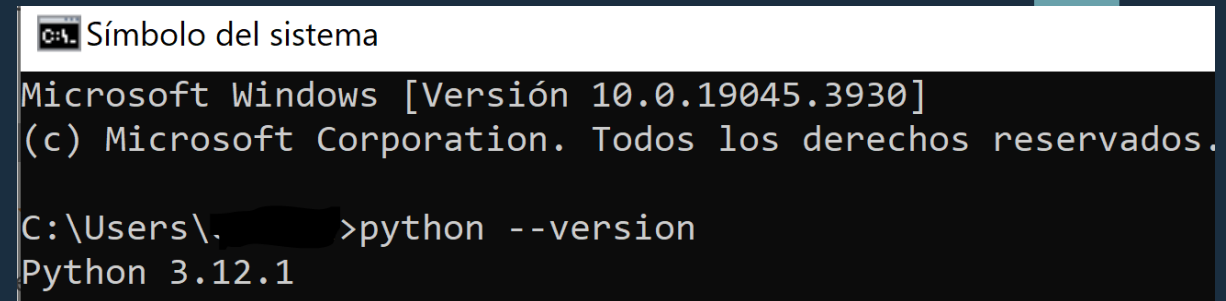
DESPLIEGUE LOCAL DE FLASK

Para poder desarrollar una aplicación web, primero haremos pruebas locales y en un módulo posterior lo haremos en un servidor web.

Primero debemos asegurarnos de tener correctamente configurado Python.

- En Consola (cmd) de Windows o Terminal de mac, ejecutar el comando “Python --version”.

Si la respuesta no es la versión que tenga instalada (no tiene que ser la 3.12 necesariamente), como se muestra en la imagen si no un mensaje de error, debe verificar la instalación de Python y configurar las variables de entorno en Windows para incluir en PATH la ubicación de Python.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.3930]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\...>python --version
Python 3.12.1
```

DESPLIEGUE LOCAL DE FLASK

Ahora debemos elegir una ubicación en el computador para crear el proyecto Flask, por ejemplo “C:/User/backend/flask_example”.

- En VSCode elegir File>Open Folder, elegir el directorio creado anteriormente.

Ahora procedemos a crear un ambiente virtual para el proyecto:

- En VSCode elegir View > Command Palette y seleccionar la opción Python: Create Environment.
- En el que se muestra elegir la opción venv

Una vez termine la creación del ambiente virtual, instalaremos Flask:

- En VSCode elegir la opción View>Terminal
- En la terminal escribir el comando “python -m pip install flask”.
- Si marca error de no reconocer el comando pip, verifique las variables de entorno y PATH, para asegurarse que esté incluido la ruta de Python y otra con Python/Scripts

DESPLIEGUE LOCAL DE FLASK

Ya tenemos un ambiente de trabajo con Flask instalado. Ahora crearemos un “Hola Mundo”.

- En VSCode elegir la opción File > New File y crear un archivo llama app.py
- Dentro de app.py escribir el siguiente código:
- Guardar cambios presionand Ctrl+S

En la Terminal de VSCode escribir el comando:

`python -m flask run`

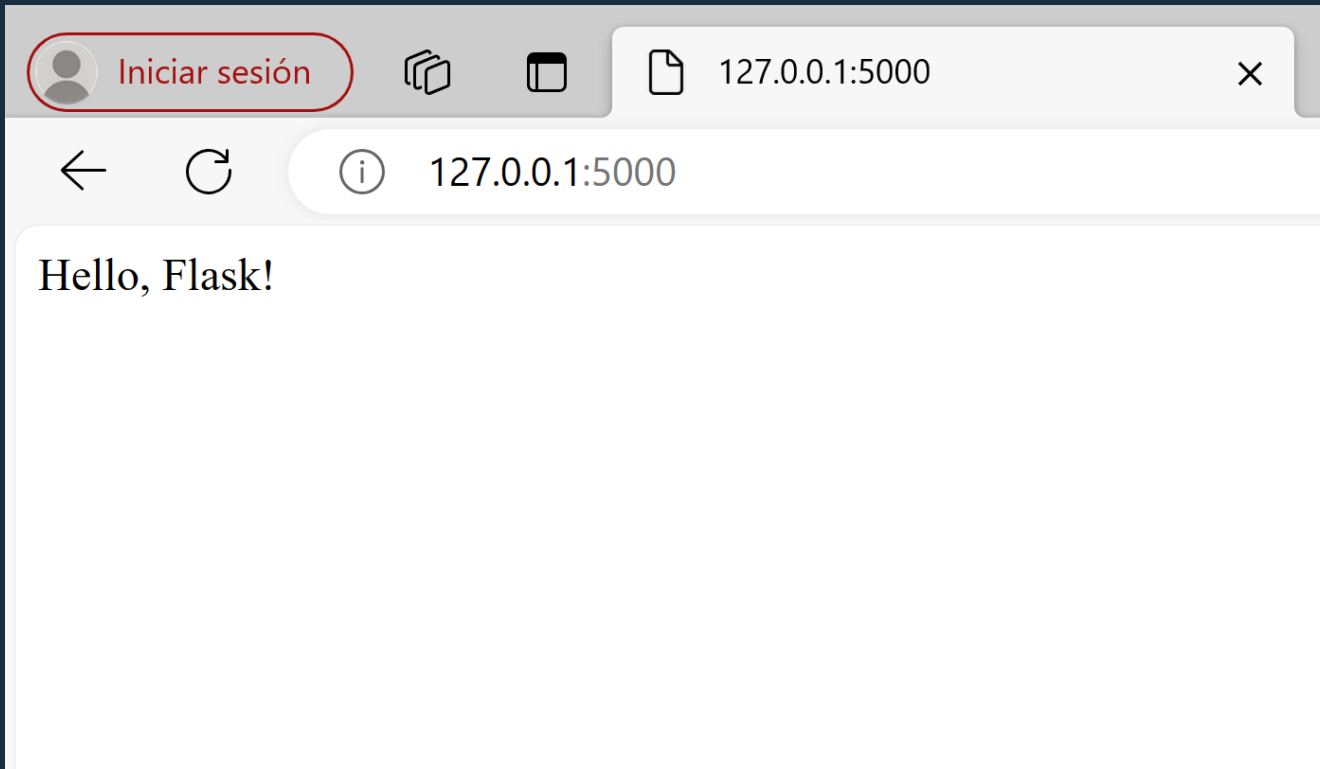
Al finalizar de ejecutarse, la Terminal debe verse algo así:

```
app.py
env > app.py > ...
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route("/")
5  def home():
6      return "Hello, Flask!"
```

```
PS C:\Users\... \Downloads\... \flask_tutorial> python -m flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

DESPLIEGUE LOCAL DE FLASK

Para ver el resultado, podemos hacer Ctrl+clic sobre la dirección en terminal o copiarla y pegarla en el navegador de preferencia.



Para finalizar la ejecución, en el Terminal de VSCode presionar Ctrl+C

MVC Y FLASK

- Patrón MVC
- Despliegue local de Flask
- Plantillas y Jinja2



PLANTILLAS Y JINJA2

Para facilitar la creación de Vistas (Templates), una herramienta muy poderosa es Jinja2, que nos permite crear plantillas dinámicas.

- Permite insertar variables, expresiones y estructuras de control en las plantillas para que se evalúen y rendericen dinámicamente cuando se generan las páginas web.
- Inyección de Datos: Facilita la inyección de datos desde el código Python en las plantillas HTML.
- Estructuras de control: proporciona estructuras de control como bucles for y condicionales if, permitiéndote realizar iteraciones y tomar decisiones dentro de las plantillas.
- Herencia de plantillas: permite la creación de plantillas base que contienen el diseño común de tu aplicación. Otras plantillas pueden extender estas plantillas base, heredando su estructura y estilo, personalizando solo las partes específicas.

PLANTILLAS Y JINJA2

Otras ventajas de Jinja2

- PreVENCIÓN de Inyecciones de HTML: Jinja2 proporciona mecanismos que ayudan a prevenir ataques de inyección de HTML, asegurando que el contenido dinámico se escape correctamente y se muestre de manera segura en el navegador.
- Integración con Frameworks Web: Jinja2 se integra con varios frameworks web de Python, como Flask y Django, facilitando la creación de aplicaciones web dinámicas y escalables. Si aprenden Jinja2 para Flask, ya tendrán conocimiento para Django

PLANTILLAS Y JINJA2

Otras ventajas de Jinja2

- PreVENCIÓN de Inyecciones de HTML: Jinja2 proporciona mecanismos que ayudan a prevenir ataques de inyección de HTML, asegurando que el contenido dinámico se escape correctamente y se muestre de manera segura en el navegador.
- Integración con Frameworks Web: Jinja2 se integra con varios frameworks web de Python, como Flask y Django, facilitando la creación de aplicaciones web dinámicas y escalables. Si aprenden Jinja2 para Flask, ya tendrán conocimiento para Django.

PLANTILLAS Y JINJA2

Apliquemos Jinja2 a nuestro proyecto de Flask en VSCode. Para esto realizaremos los siguientes cambios:

- Dentro de la carpeta raíz del proyecto crear la carpeta templates(vistas) y agregar un archivo llamado index.html con el siguiente contenido:

Esta es la plantilla que usará información del modelo: usuario.nombre y usuario.edad. Diferentes módulos pueden usar esta plantilla (Reutilización de código!!!)

```
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Flask y Jinja2</title>
</head>
<body>
  <h1>Bienvenido, {{ usuario.nombre }}!</h1>
  <p>Edad: {{ usuario.edad }}</p>
</body>
</html>
```

PLANTILLAS Y JINJA2

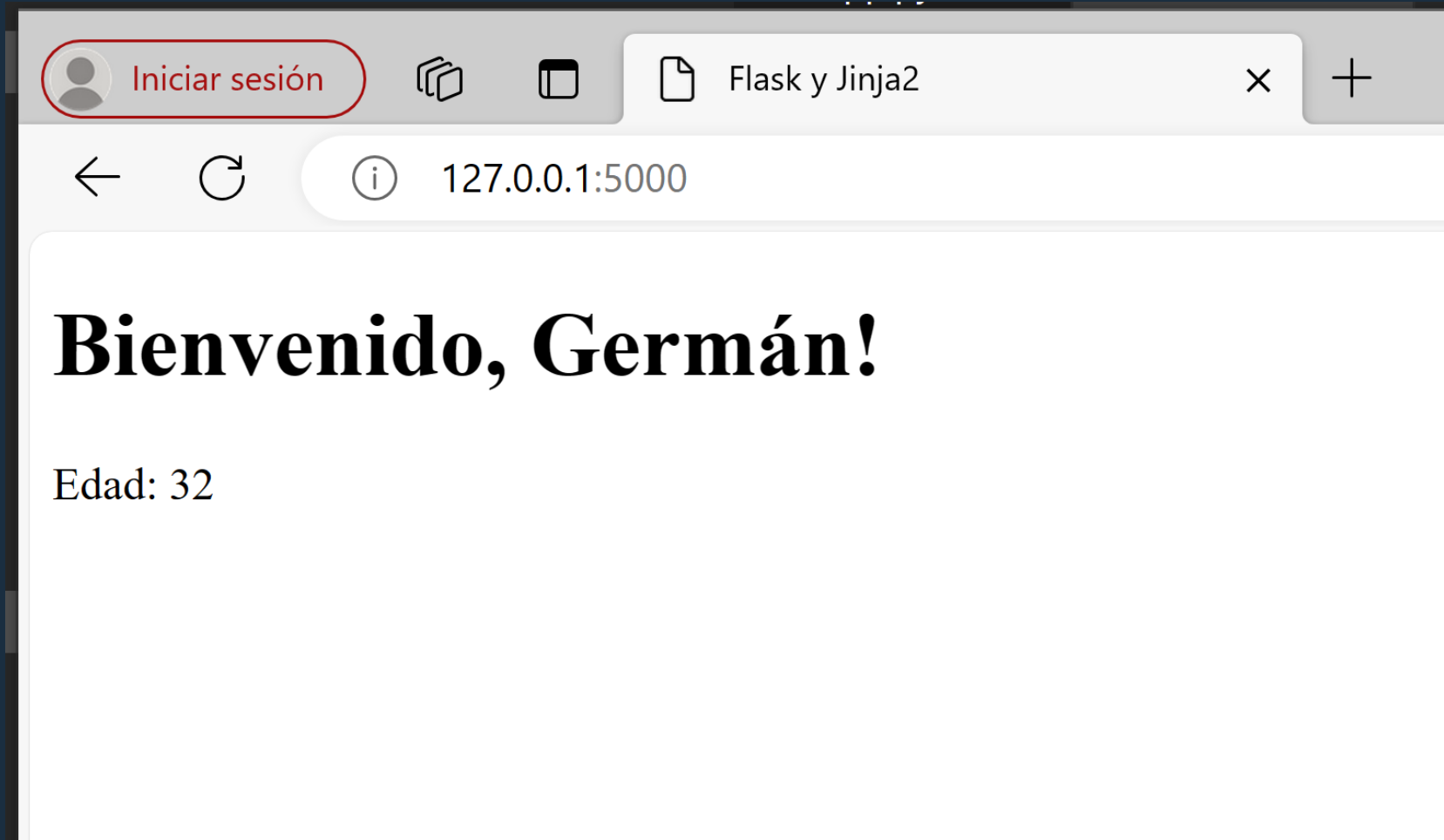
Ahora debemos modifica un poco el archivo app.py, para que se vea algo así:

```
.venv >  app.py > ...  
1  from flask import Flask, render_template  
2  
3  app = Flask(__name__)  
4  
5  @app.route('/')  
6  def index():  
7      # Ejemplo de datos que se pasarán a la plantilla  
8      usuario = {'nombre': 'Germán', 'edad': 32}  
9      return render_template('index.html', usuario=usuario)  
10  
11  if __name__ == '__main__':  
12      app.run(debug=True)
```

Ahora se importa render_template y además se crea el usuario con los valores que queremos mostrar.

PLANTILLAS Y JINJA2

Al ejecutarlo nuevamente por consola con el comando “python -m flask run”, se va a ver algo así en el navegador:



MANOS A LA OBRA

Taller 1 disponible en BloqueNeon

