Frameworks

Conectando el mundo web

- Aram Baruch González Pérez




Introducción

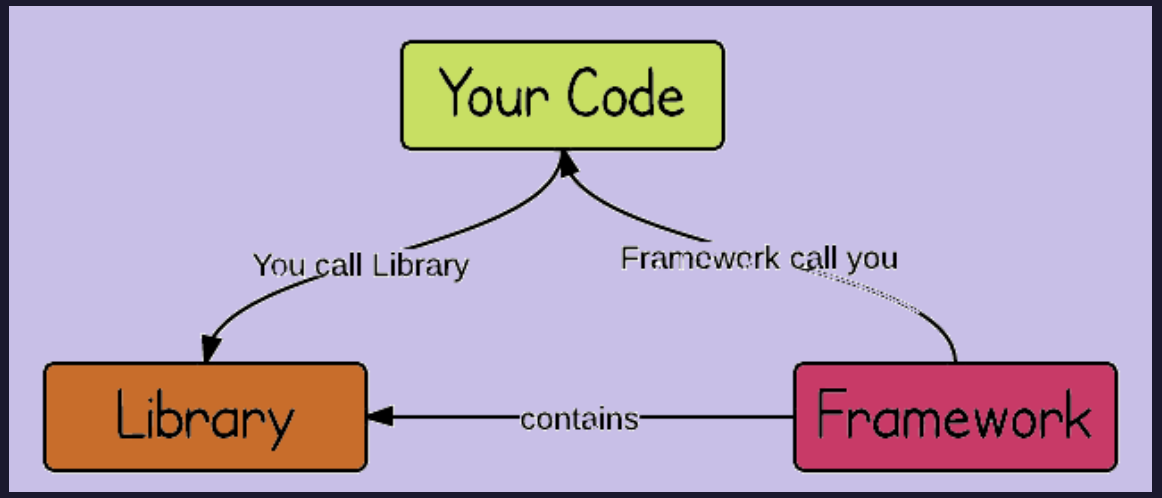
- Los frameworks nos ayudan a agilizar el desarrollo de nuestros proyectos. La mayoría cuenta con una documentación robusta y ejemplos que podemos usar como base.

Framework

¿QUÉ SON?

- Son una estructura preestablecida que nos ayuda a construir software.
- Nos ayudan a ahorrar tiempo y reducir errores.
- No es necesario reinventar la rueda todo el tiempo.

FRAMEWORKS VS BIBLIOTECAS (LIBRERÍAS)



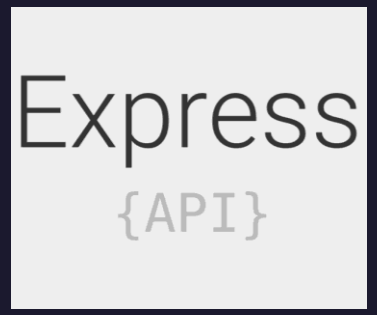
Ejemplos

Flask

Python

Angular

Javascript



Express

Javascript

Rails

Ruby

Desarrollo web

BACKEND

- Es todo el código que se ejecuta en el servidor.
- Se encarga de recibir peticiones, procesarlas y enviar la respuesta adecuada.
- Se comunica con otros recursos como APIs y bases de datos.

FRONTEND

- Es el código que se despliega en la respuesta al cliente.
- Tradicionalmente incluye al código html generado.
- Cuando se genera un código dinámico y responsivo, se habla de aplicaciones web.

Flask

¿QUÉ ES?

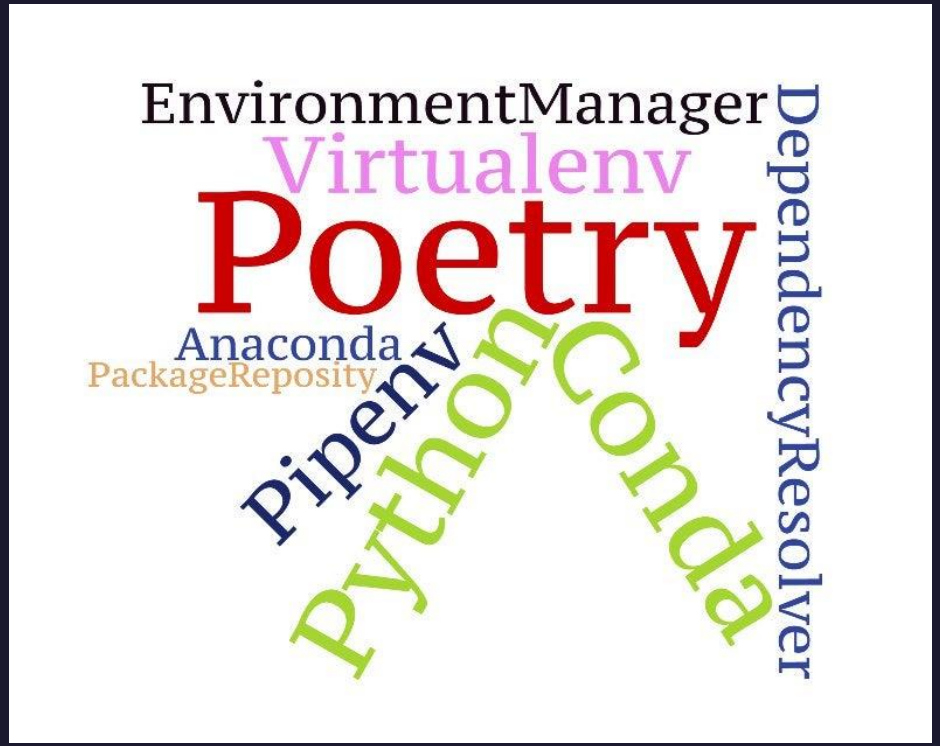
- Es un micro framework en Python para el desarrollo web.
- En el backend utiliza directamente Python.
- Para el frontend utiliza Jinja2.
  - Sistema a base de templates.

Entornos virtuales

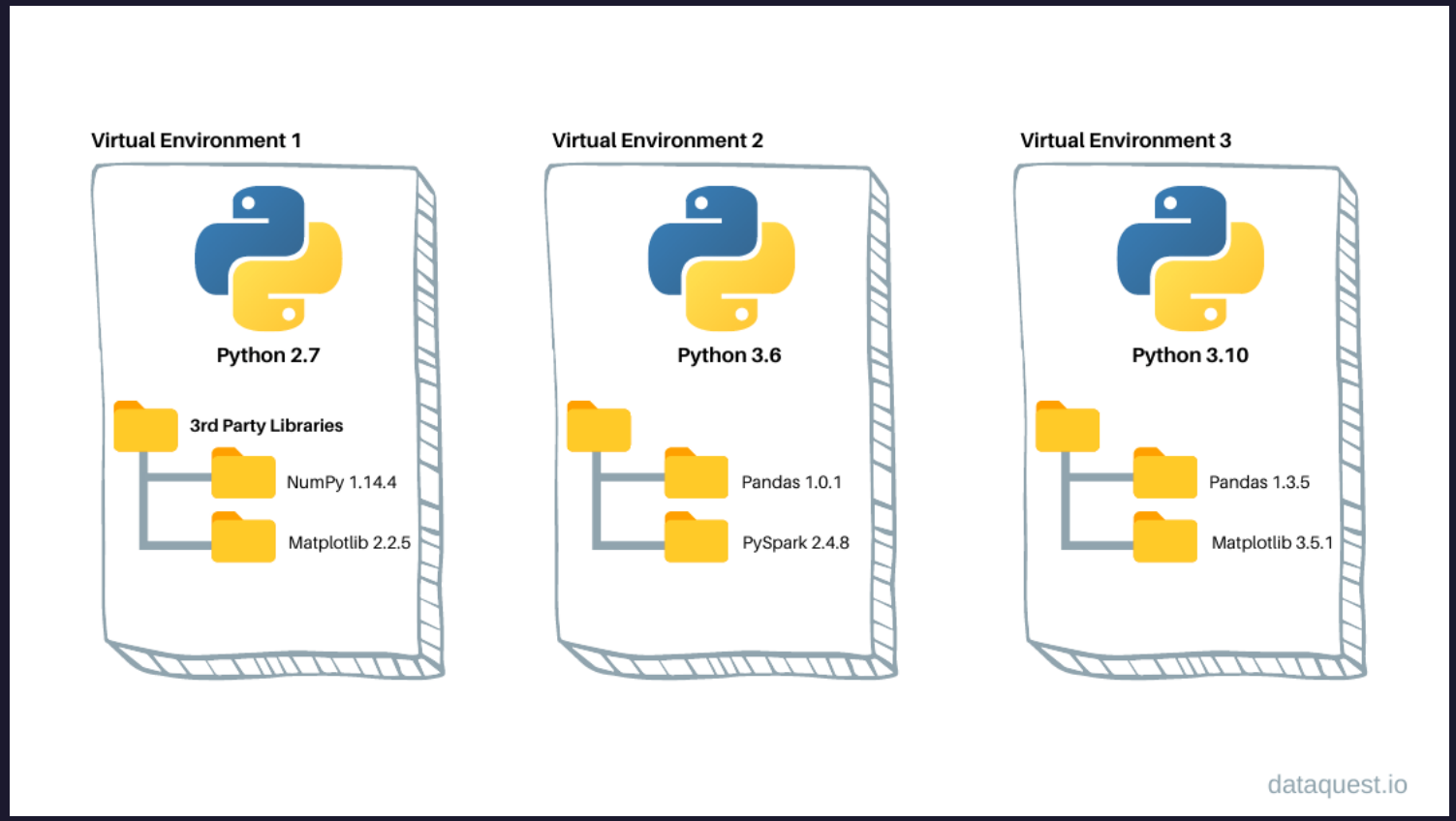
¿POR QUÉ?

- Se recomiendan en caso de tener dependencias en nuestros proyectos.
- Es posible que tengamos que trabajar con distintas versiones de bibliotecas o incluso de Python.
- Podemos tener en aislamiento estas dependencias.

MANEJADORES DE ENTORNOS VIRTUALES



Entornos virtuales

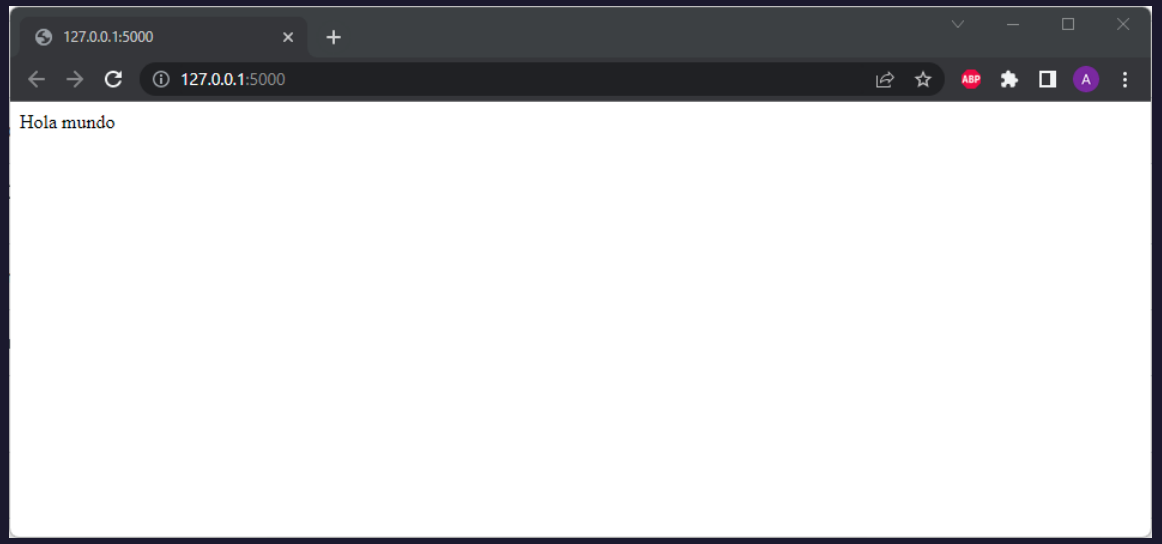


Ejemplo básico

CÓDIGO

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Hola mundo"

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

VISUALIZACIÓN



Contenido estático

STATIC

- Podemos incluir contenido al que queremos dar acceso desde nuestro sitio.
- Es contenido que no cambiará de manera dinámica.
- Pueden ser imágenes, videos, archivos css, etc.
- Se crea una carpeta con nombre static en nuestro proyecto

Definiendo rutas

DECORADOR

- Se debe usar el decorador:
  - @app.route(“ruta”)
- Se pueden especificar los métodos que se reciben.
  - @app.route(“ruta”, methods=[“GET”, “POST”])
  - Por defecto se reciben mensajes tipo GET.

FUNCIÓN

- La función debajo del decorador hará todo el procesamiento necesario.
- Se debe regresar el contenido que se mandará al cliente.
  - Una redirección.
  - Un template.
  - Contenido estático.

Parámetros

URI (PATH) PARAMETER

- Se utiliza para identificar un recurso a acceder.
- Usualmente están ligados a un identificador único.
- Son muy comunes para despachar ítems en APIs tipo REST.

QUERY PARAMETER

- Se emplea comúnmente para indicar datos opcionales en el URL que se accede.
- En APIs tipo REST se emplean para aplicar filtros y ordenamiento.
- Los podemos encontrar después de la URL.

Parámetros

URI (PATH) PARAMETERS

- Se definen en la URL encerrados entre signos de menor que y mayor que.
  - /ruta/<ejemplo>
- Se mandan a nuestra función como parámetro.
- El tipo de dato es string.

QUERY PARAMETERS

- Se codifican directamente en la URL.
- Ejemplo:
  - /ruta?param1=hola&param2=mundo
- Se obtienen con:
  - request.args

Parámetros

URI (PATH) PARAMETERS

```python
@app.route("/ruta/<parametro>")
def ejemplo1(parametro):
    return parametro
```

QUERY PARAMETERS

```python
@app.route("/otra_ruta")
def ejemplo2():
    param1 = request.args.get("param1")
    param2 = request.args.get("param1")
    return f"{param1} {param2}"
```

Templates

¿QUÉ SON?

- Son archivos con contenido estático e indicaciones para acomodar contenido dinámico.
- Nos encargaremos de mandar los datos que requieren para producir el documento final.
- Se hace por medio de Jinja.
- Se guardan en una carpeta llamada “templates”

SINTAXIS

- Se acostumbra a utilizar HTML como la parte estática del documento.
- Para la parte dinámica la podemos generar con:
  - {% … %} – No produce salidas.
  - {{ … }} – Genera una salida en el documento final.

Ejemplo

CÓDIGO PYTHON

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    lista_ejemplo = ["Armando", "Francisco", "María"]
    return render_template("/index.html", lista=lista_ejemplo)

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

TEMPLATE

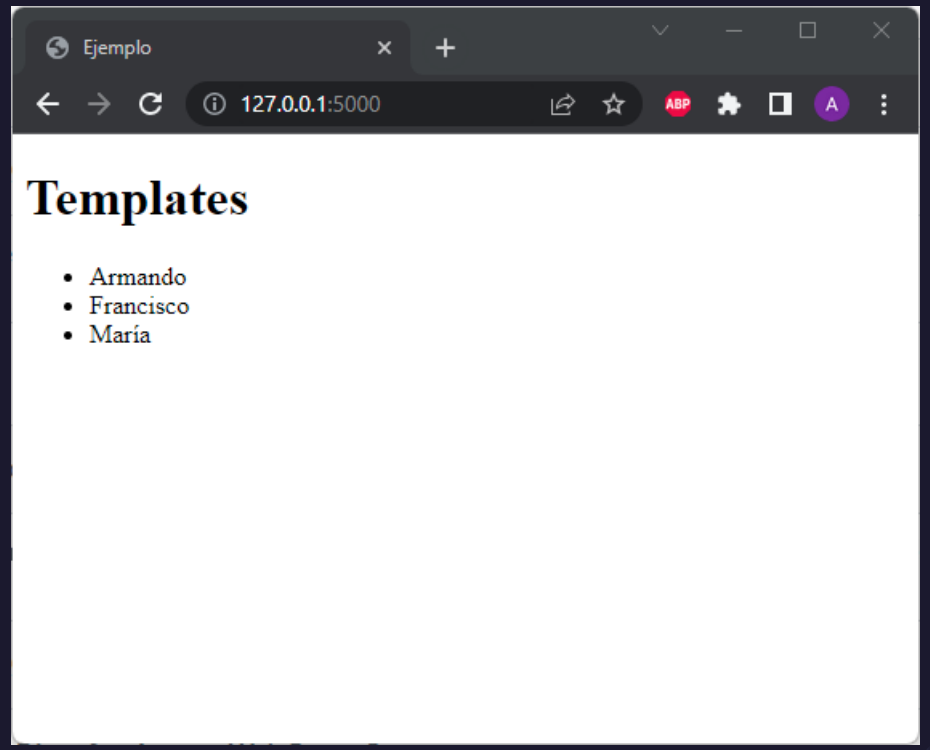
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo</title>
</head>
<body>
    <h1>Templates</h1>
    <ul>
    {% for i in lista %}
        <li>{{ i }}</li>
    {% endfor %}
    </ul>
</body>
</html>
```

Ejemplo

DOCUMENTO FINAL

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo</title>
</head>
<body>
    <h1>Templates</h1>
    <ul>

        <li>Armando</li>

        <li>Francisco</li>

        <li>María</li>

    </ul>
</body>
</html>
```

VISTA



Actividad

- Modificar el código de presentación de forma que:
- La página principal siga siendo la misma.
- En Flask manejar las rutas para los nombres de cada alumno como URI Parameters.
  - El ingresar a la ruta se seguirá haciendo como previamente.
  - Su código en Python debe encargarse de procesar el template para que se despliegue de manera correcta la información.