# Presentación - Python (Flask), Graficas y Dashboard y Generar reportes PDF

## 1. Introducción a Flask

### ¿Qué es Flask?

Flask es un microframework para Python diseñado para facilitar el desarrollo de aplicaciones web. Es "micro" porque intenta mantener un núcleo simple pero extensible. No incluye componentes adicionales por defecto, permitiendo a los desarrolladores agregar solo las librerías y herramientas que necesitan.

**Características de Flask:**

- **Ligero y fácil de usar:** Flask es minimalista, lo que lo hace accesible tanto para principiantes como para desarrolladores avanzados. Su simplicidad permite a los desarrolladores concentrarse en la lógica de la aplicación sin tener que preocuparse por la configuración o la sobrecarga de características innecesarias.
- **Simplicidad y flexibilidad:** Flask no impone una estructura rígida de proyecto, permitiendo a los desarrolladores organizar su código de la manera que consideren más conveniente. Esta flexibilidad facilita la adaptación del proyecto a las necesidades específicas de cada desarrollo.
- **Gran comunidad y documentación:** Flask cuenta con una amplia comunidad de usuarios y desarrolladores, lo que se traduce en una abundante cantidad de recursos, tutoriales y extensiones disponibles. La documentación oficial es extensa y clara, facilitando el aprendizaje y la resolución de problemas.
- **Incluye un servidor web de desarrollo:** Flask proporciona un servidor web de desarrollo integrado que permite a los desarrolladores probar sus aplicaciones de manera sencilla. No es necesario configurar una infraestructura de servidor web completa para ver los resultados, lo que agiliza el proceso de desarrollo.
- **Depurador y soporte integrado para pruebas unitarias:** Flask incluye un depurador que facilita la identificación y resolución de errores en el código. Además, permite ver los valores de las variables en tiempo de ejecución. También soporta la integración de pruebas unitarias, lo que es fundamental para asegurar la calidad y estabilidad de la aplicación.

**Casos de uso comunes:**

- **Aplicaciones web simples:** Flask crea varios tipos de aplicaciones web, blogs, páginas personales o sitios de comercio electrónico.
- **Desarrollo de API:** cree **API RESTful** que permitan interactuar con aplicaciones móviles u otros servicios.
- **Prototipos:** Crear prototipos rápidos debido a su naturaleza ligera.
- **Microservicios:** Desarrollar pequeños servicios dentro de una arquitectura de microservicios.

- **Paneles interactivos:** Visualización de datos en tiempo real.
- **Proyectos educativos:** Usado frecuentemente en cursos y tutoriales para enseñar desarrollo web.

## 2. Configuración y Estructura Básica

**Instalación:** Para instalar Flask, se utiliza el gestor de paquetes `pip`:

```
bash
pip install Flask
```

**Estructura de un proyecto Flask:** Un proyecto Flask típico puede organizarse de la siguiente manera:

- **app.py:** Archivo principal que contiene la aplicación Flask.
- **templates/:** Carpeta que contiene las plantillas HTML.
- **static/:** Carpeta para archivos estáticos como CSS, JavaScript e imágenes.

**Primeros pasos:** ¡Un ejemplo básico de una aplicación Flask “Hello, World!" es el siguiente:

```
pyhton
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True)
```

Para ejecutar la aplicación, se guarda este código en un archivo llamado `app.py` y se ejecuta con:

```
bash
python app.py
```

---

## Gráficas y Dashboards

## Gráficas y Dashboards

1. **Gráfica**: Una representación visual de datos numéricos mediante líneas, barras, puntos u otros elementos gráficos. Las gráficas permiten visualizar patrones, tendencias o comparaciones entre diferentes conjuntos de datos.
2. **Dashboard**: Un panel visual que muestra de manera consolidada información clave y métricas de un negocio, sistema o proceso. Los dashboards suelen incluir gráficas,

tablas y otros elementos interactivos que permiten monitorear y analizar datos en tiempo real.

**Librerías Comunes para Gráficos**

- **matplotlib:** Es una de las librerías más utilizadas para crear gráficos en Python. Ofrece una amplia variedad de gráficos y una gran capacidad de personalización.
- **seaborn:** Construida sobre matplotlib, seaborn simplifica la creación de gráficos estadísticos y es ideal para análisis de datos. Proporciona una interfaz más amigable y estilos predeterminados atractivos.
- **plotly:** Permite la creación de gráficos interactivos y de alta calidad. Es muy útil para visualizaciones que necesitan ser exploradas y manipuladas dinámicamente en una página web.

**Integración de Gráficos en Flask**

- **Generación de gráficos en el servidor:** Puedes utilizar las librerías mencionadas para generar gráficos en el servidor de Flask. Los gráficos pueden ser guardados como imágenes y luego enviados al cliente.
- **Visualización en plantillas HTML:** Los gráficos generados pueden ser incrustados en plantillas HTML usando Jinja2, el motor de plantillas de Flask. Alternativamente, plotly ofrece la posibilidad de integrar gráficos directamente en HTML como componentes interactivos.

**Frameworks de Dashboards**

- **Dash:** Desarrollado por los creadores de plotly, Dash es un framework que facilita la creación de aplicaciones web interactivas y dashboards con componentes de gráficos integrados.
- **Bokeh:** Otro framework que permite la creación de gráficos interactivos y dashboards que se pueden integrar fácilmente en aplicaciones web.

**Ejemplo de Dashboards y Gráficas**

Aquí hay un ejemplo básico de cómo generar y mostrar un gráfico usando Flask y matplotlib:

```
python
from flask import Flask, render_template
import matplotlib.pyplot as plt
import io
import base64

app = Flask(__name__)

@app.route('/')
def home():
    # Crear una figura de matplotlib
    fig, ax = plt.subplots()
```

```
    ax.plot([1, 2, 3, 4], [10, 20, 25, 30])
    ax.set_title('Ejemplo de Gráfico')

    # Guardar la figura en un objeto IO
    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)

    # Codificar la imagen en base64
    graph_url = base64.b64encode(img.getvalue()).decode()

    return render_template('index.html', graph_url=graph_url)

if __name__ == '__main__':
    app.run(debug=True)
```

Y en el archivo `index.html`:

```
html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Gráfico con Flask</title>
</head>
<body>
    <h1>Gráfico Generado con Flask</h1>
    <img src="{{ graph_url }}">
</body>
</html>
```

## Generación de Reportes PDF

## Reportes PDF

1. **Reporte PDF**: Un documento digital en formato PDF que presenta de manera estructurada y formal información detallada sobre un tema específico. Los reportes PDF suelen incluir texto descriptivo, tablas, gráficos y otros elementos visuales para comunicar análisis, resultados o conclusiones.

PDF significa "Portable Document Format" (Formato de Documento Portátil). Es un formato de archivo desarrollado por Adobe que permite presentar documentos de manera independiente del software, hardware y sistema operativo utilizados para crearlos. Esto significa que los archivos PDF mantienen la misma apariencia (incluyendo gráficos, imágenes y diseño) independientemente de dónde se abran, asegurando consistencia en la visualización y la impresión. Los archivos PDF son ampliamente utilizados para documentos que necesitan ser compartidos o impresos con exactitud, como informes, manuales, formularios, ebooks y otros tipos de documentos digitales.

### Librerías para Generación de PDF

- **ReportLab:** Una librería robusta para generar PDFs dinámicos. Permite una gran personalización de los documentos.
- **WeasyPrint:** Convierte HTML y CSS en PDFs, lo que facilita la creación de documentos PDF con estilos complejos usando tecnologías web.
- **FPDF:** Una opción ligera y fácil de usar para generar PDFs. Ideal para documentos más simples.

**Integración con Flask**

- **Creación y descarga de PDFs desde una aplicación Flask:** Puedes usar las librerías mencionadas para generar PDFs en el servidor y ofrecerlos para descarga a través de tu aplicación Flask.
- **Ejemplo de código para generar un reporte simple:**

Aquí hay un ejemplo básico usando ReportLab:

```
python
from flask import Flask, send_file
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
import io

app = Flask(__name__)

@app.route('/download')
def download_pdf():
    buffer = io.BytesIO()
    p = canvas.Canvas(buffer, pagesize=letter)
    p.drawString(100, 750, "Hello, this is a simple PDF report.")
    p.showPage()
    p.save()

    buffer.seek(0)
    return send_file(buffer, as_attachment=True,
download_name='report.pdf', mimetype='application/pdf')

if __name__ == '__main__':
    app.run(debug=True)
```

Este código genera un PDF simple y permite a los usuarios descargarlo desde una ruta específica en la aplicación Flask.

Para desarrollar un proyecto utilizando Flask, Python, dashboards y generación de reportes PDF, aquí tienes algunos conceptos clave que necesitarías saber:

1. **Flask**: Es un framework web ligero para Python que te permite construir aplicaciones web de manera rápida y eficiente. Algunos puntos importantes a conocer:
    - **Routing**: Define las URL de tu aplicación y cómo deben ser manejadas.
    - **Templates**: Utiliza Jinja2 para renderizar HTML dinámico con datos desde Python.
    - **Formularios**: Procesamiento de formularios HTML para interactuar con el usuario.
    - **Extensiones**: Flask tiene una amplia gama de extensiones que facilitan tareas comunes como la autenticación, manejo de bases de datos, etc.
2. **Dashboards**:
    - Un dashboard es una interfaz visual que proporciona una vista consolidada de datos, gráficos y métricas importantes para la toma de decisiones.
    - Puedes construir dashboards en Flask utilizando bibliotecas como Plotly Dash o simplemente integrando gráficos generados con bibliotecas como Matplotlib o Plotly en tus plantillas HTML.
3. **Generación de reportes PDF**:
    - Utilizarás bibliotecas como WeasyPrint o ReportLab para generar documentos PDF desde tus datos y plantillas HTML.
    - Debes saber cómo integrar datos dinámicos en las plantillas HTML utilizando Flask y Jinja2, y luego convertir esas plantillas en PDF.
4. **Manipulación de datos y gráficos**:
    - Conocimientos en Python para manipular datos y generar gráficos utilizando bibliotecas como Matplotlib o Plotly.
    - Integración de estos gráficos en tus plantillas HTML para mostrarlos en el dashboard.
5. **Estructura del proyecto en Flask**:
    - Entender la estructura de directorios y archivos en un proyecto Flask (como mencionaste anteriormente) para organizar tu código de manera eficiente.

## Gráficas y Dashboards

- **Librerías comunes para gráficos**
  - `matplotlib`, `seaborn`, `plotly`.
- **Integración de gráficos en Flask**
  - Generación de gráficos en el servidor.
  - Visualización en plantillas HTML.
- **Frameworks de Dashboards**
  - `Dash` (construido sobre Flask y Plotly).
  - Ventajas de usar Dash para dashboards interactivos.

## Generación de Reportes PDF

- **Librerías para generación de PDF**
  - `ReportLab`, `WeasyPrint`, `FPDF`.
- **Integración con Flask**
  - Creación y descarga de PDFs desde una aplicación Flask.
  - Ejemplo de código para generar un reporte simple.

## 6. Ejemplo de Proyecto

- **Descripción del proyecto**
  - Aplicación web sencilla que muestra un dashboard y permite la descarga de reportes en PDF.
- **Demostración del flujo de trabajo**
  - Registro de datos, visualización de gráficos, generación y descarga de reportes.

## 7. Conclusión y Recursos Adicionales

- **Resumen de puntos clave**
- **Recursos para aprender más**
  - Documentación oficial de Flask.
  - Tutoriales y cursos recomendados.
- **Preguntas y Respuestas**

## Tips para la Presentación:

- **Visuales atractivos**
  - Usa capturas de pantalla y diagramas para ilustrar conceptos.
- **Ejemplos de código**
  - Mantén el código simple y relevante.
- **Interactividad**
  - Si es posible, realiza demostraciones en vivo.
- **Evita sobrecargar**
  - Presenta la información en secciones claras y manejables.