**Version basica de un proyecto con :**

**── main.py # Punto de entrada del programa**

**── procesamiento.py # Funciones de análisis de datos**

**── reportes.py # Funciones para generar PDFs**

**── app.py # Dashboard con Streamlit**

1. **📁 /mi\_proyecto\_python/procesamiento.py**
2. import pandas as pd
3. def cargar\_datos():
4. # Simulación de carga de datos
5. data = {
6. 'cliente': ['Alice', 'Bob', 'Carlos'],
7. 'ventas': [150, 200, 300]
8. }
9. df = pd.DataFrame(data)
10. return df
11. **📁 /mi\_proyecto\_python/reportes.py**
12. def generar\_pdf(dataframe, archivo="reporte.pdf"):
13. c = canvas.Canvas(archivo, pagesize=letter)
14. c.drawString(100, 750, "Reporte de Ventas")
15. y = 700
16. for index, row in dataframe.iterrows():
17. texto = f"Cliente: {row['cliente']}, Ventas: {row['ventas']}"
18. c.drawString(100, y, texto)
19. y -= 20
20. c.save()
21. print(f"PDF generado: {archivo}")
22. **📁 /mi\_proyecto\_python/app.py**
23. import streamlit as st
24. from procesamiento import cargar\_datos
25. def main():
26. st.title("Dashboard de Ventas")
27. df = cargar\_datos()
28. st.dataframe(df)
29. st.bar\_chart(df.set\_index("cliente"))
30. if \_\_name\_\_ == "\_\_main\_\_":
31. main()
32. **📁 /mi\_proyecto\_python/main.py**
33. import pandas as pd
34. from procesamiento import cargar\_datos
35. from reportes import generar\_pdf
36. def main():
37. df = cargar\_datos()
38. if df is None or df.empty:
39. print("⚠️ No se pudo cargar el DataFrame.")
40. return
41. print("✅ Datos cargados:\n", df)
42. generar\_pdf(df)
43. print("📄 PDF generado con éxito.")
44. if \_\_name\_\_ == "\_\_main\_\_":
45. main()

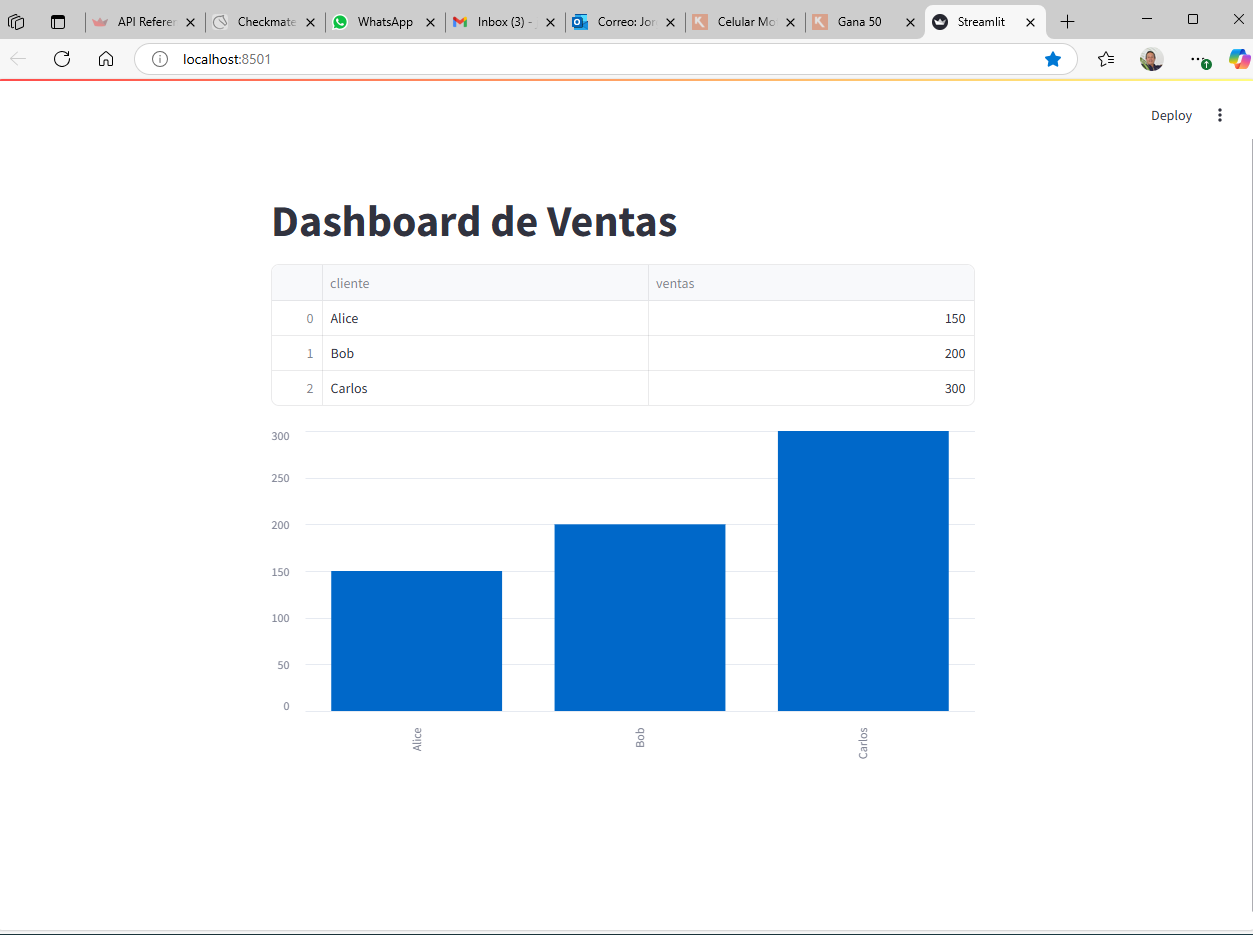
**▶️ ¿Cómo lo ejecuta?**

1. Abre una terminal en /mi\_proyecto\_python
2. Para ejecutar el flujo principal (cargar y generar PDF):

python main.py

Para ver el dashboard en navegador:

streamlit run app.py



**Version 2**

**le agregue generación de gráficos como imagen en el PDF o detección automática de archivos .xlsx en una carpeta:**

* **📂 Detecta automáticamente archivos .xlsx en una carpeta /data.**
* **📊 Genera un gráfico de barras con matplotlib.**
* **🖼️ Guarda el gráfico como imagen.**
* **🧾 Inserta ese gráfico en el PDF generado.**

**🧱 Estructura actualizada del proyecto**

**/mi\_proyecto\_python**

**── data/**

**│ └── ventas.xlsx**

**---- src/ABC\_Proyecto2**

**── main.py # Punto de entrada del programa**

**── procesamiento.py # Funciones de análisis de datos**

**── reportes.py # Genera PDF con gráfico**

**└── app.py # Dashboard con Streamlit**

**✅ /data/ventas.xlsx (simulado)**

**Cree un archivo Excel sencillo con estas columnas:**

| **cliente** | **ventas** |
| --- | --- |
| **Alice** | **150** |
| **Bob** | **200** |
| **Carlos** | **300** |

**✅ /procesamiento.py**

**import pandas as pd**

**import os**

**def detectar\_archivo\_excel(carpeta="data"):**

**for archivo in os.listdir(carpeta):**

**if archivo.endswith(".xlsx"):**

**return os.path.join(carpeta, archivo)**

**raise FileNotFoundError("No se encontró ningún archivo .xlsx en la carpeta 'data'.")**

**def cargar\_datos():**

**archivo = detectar\_archivo\_excel()**

**df = pd.read\_excel(archivo)**

**return df**

**✅ /reportes.py**

**from reportlab.lib.pagesizes import letter**

**from reportlab.pdfgen import canvas**

**import matplotlib.pyplot as plt**

**def generar\_grafico(df, imagen="grafico.png"):**

**plt.figure(figsize=(6, 4))**

**plt.bar(df["cliente"], df["ventas"], color="skyblue")**

**plt.title("Ventas por Cliente")**

**plt.xlabel("Cliente")**

**plt.ylabel("Ventas")**

**plt.tight\_layout()**

**plt.savefig(imagen)**

**plt.close()**

**print(f"Gráfico guardado como {imagen}")**

**def generar\_pdf(df, archivo="reporte.pdf", imagen="grafico.png"):**

**c = canvas.Canvas(archivo, pagesize=letter)**

**c.drawString(100, 750, "Reporte de Ventas")**

**y = 700**

**for index, row in df.iterrows():**

**texto = f"Cliente: {row['cliente']}, Ventas: {row['ventas']}"**

**c.drawString(100, y, texto)**

**y -= 20**

**# Insertar gráfico**

**c.drawImage(imagen, 100, y - 250, width=400, height=300)**

**c.save()**

**print(f"PDF generado: {archivo}")**

**✅ /app.py (Streamlit)**

**import streamlit as st**

**from procesamiento import cargar\_datos**

**def main():**

**st.title("Dashboard de Ventas")**

**df = cargar\_datos()**

**st.dataframe(df)**

**st.bar\_chart(df.set\_index("cliente"))**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**✅ /main.py**

**from procesamiento import cargar\_datos**

**from reportes import generar\_pdf, generar\_grafico**

**def main():**

**print("Cargando datos desde archivo Excel...")**

**df = cargar\_datos()**

**print(df)**

**print("Generando gráfico de ventas...")**

**generar\_grafico(df)**

**print("Generando PDF con gráfico...")**

**generar\_pdf(df)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**▶️ ¿Cómo probarlo?**

1. **Guarde un archivo ventas.xlsx dentro de la carpeta /data.**
2. **Desde la terminal:**

**python main.py**

**PS C:\Users\PC\Documents\mi\_proyecto\_python\src\ABC\_2> python main.py**

**Cargando datos desde archivo Excel...**

**cliente ventas**

**0 Alice 150**

**1 Bob 200**

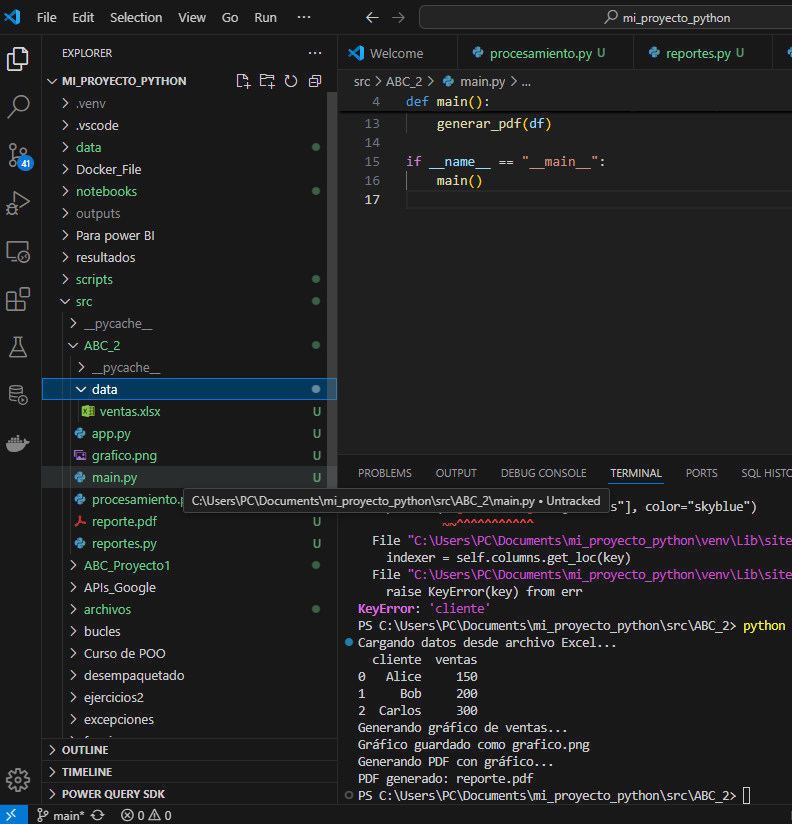
**2 Carlos 300**

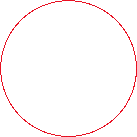
**Generando gráfico de ventas...**

**Gráfico guardado como grafico.png**

**Generando PDF con gráfico...**

**PDF generado: reporte.pdf**





**➡️ Resultado: verá el DataFrame impreso, el gráfico generado como grafico.png y un PDF llamado reporte.pdf con el gráfico incrustado.**

1. **También puede lanzar el dashboard con:**

**streamlit run app.py**

**Version 3 con alerta por correo si las ventas de un cliente bajan mucho, o si no hay datos nuevos**

**¡Perfecto! Vamos a incorporar una alerta por correo si detectamos:**

1. **📉 Que las ventas de un cliente cayeron drásticamente (por ejemplo, más del 20% respecto al archivo anterior).**
2. **⚠️ Que no hay archivos nuevos para procesar.**

**🧩 ¿Qué necesitamos?**

* **Usar smtplib para enviar correos.**
* **Guardar una copia del último DataFrame procesado para comparar.**
* **Añadir lógica para comparar archivos y detectar cambios significativos.**

**✅ Actualización de /procesamiento.py**

**#Agregamos persistencia en .csv y detección de cambios:**

**import pandas as pd**

**import os**

**def detectar\_archivo\_excel(carpeta="data"):**

**archivos = sorted(**

**[f for f in os.listdir(carpeta) if f.endswith(".xlsx")],**

**key=lambda x: os.path.getmtime(os.path.join(carpeta, x)),**

**reverse=True**

**)**

**if archivos:**

**return os.path.join(carpeta, archivos[0])**

**else:**

**raise FileNotFoundError("No se encontró ningún archivo .xlsx en la carpeta 'data'.")**

**def cargar\_datos():**

**archivo = detectar\_archivo\_excel()**

**df = pd.read\_excel(archivo)**

**return df**

**def guardar\_snapshot(df, path="ultimo\_snapshot.csv"):**

**df.to\_csv(path, index=False)**

**def cargar\_snapshot\_anterior(path="ultimo\_snapshot.csv"):**

**if os.path.exists(path):**

**return pd.read\_csv(path)**

**return None**

**✅ Nuevo archivo: /alertas.py**

**import smtplib**

**from email.mime.text import MIMEText**

**def enviar\_alerta(asunto, mensaje, destinatario, remitente, clave):**

**msg = MIMEText(mensaje)**

**msg['Subject'] = asunto**

**msg['From'] = remitente**

**msg['To'] = destinatario**

**with smtplib.SMTP\_SSL('smtp.gmail.com', 465) as server:**

**server.login(remitente, clave)**

**server.send\_message(msg)**

**print(f"Correo enviado a {destinatario} con asunto: {asunto}")**

**def verificar\_disminucion(df\_actual, df\_anterior, umbral=0.2):**

**caidas = []**

**if df\_anterior is not None:**

**for \_, fila in df\_actual.iterrows():**

**cliente = fila['cliente']**

**venta\_actual = fila['ventas']**

**anterior = df\_anterior[df\_anterior['cliente'] == cliente]**

**if not anterior.empty:**

**venta\_anterior = anterior['ventas'].values[0]**

**if venta\_anterior > 0 and (venta\_actual < venta\_anterior \* (1 - umbral)):**

**caidas.append((cliente, venta\_anterior, venta\_actual))**

**return caidas**

**✅ Actualización de /main.py**

**from procesamiento import cargar\_datos, guardar\_snapshot, cargar\_snapshot\_anterior**

**from reportes import generar\_pdf, generar\_grafico**

**from alertas import verificar\_disminucion, enviar\_alerta**

**# Configuración**

**REMITENTE = "tu\_correo@gmail.com"**

**CLAVE = "tu\_clave\_app" # Use una clave de aplicación**

**DESTINATARIO = "destinatario@correo.com"**

**def main():**

**print("Cargando datos...")**

**df = cargar\_datos()**

**print(df)**

**# Comparar con snapshot anterior**

**df\_anterior = cargar\_snapshot\_anterior()**

**caidas = verificar\_disminucion(df, df\_anterior)**

**if caidas:**

**cuerpo = "Se detectaron caídas significativas en ventas:\n\n"**

**for cliente, antes, ahora in caidas:**

**cuerpo += f"• {cliente}: de {antes} a {ahora}\n"**

**enviar\_alerta("⚠️ Alerta de ventas bajas", cuerpo, DESTINATARIO, REMITENTE, CLAVE)**

**print("Generando gráfico y reporte PDF...")**

**generar\_grafico(df)**

**generar\_pdf(df)**

**# Guardar nuevo snapshot**

**guardar\_snapshot(df)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**✅ ¿Cómo lo prueba?**

1. **Cree un Excel con ventas normales y ejecútelo una vez.**
2. **Modifique ese Excel bajando las ventas de un cliente en más del 20%.**
3. **Ejecute de nuevo main.py, y debería dispararse un correo con la alerta.**

**🛡️ Seguridad**

**Para Gmail, debe usar una clave de aplicación y activar la verificación en dos pasos.**

**🛡️ 1. Activa la verificación en dos pasos en tu cuenta de Google**

1. **Entra a 👉 https://myaccount.google.com/security**
2. **Busca la sección "Verificación en dos pasos" y haz clic.**
3. **Sigue el proceso (normalmente te pedirá tu número de celular o dispositivo confiable).**
4. **Una vez activada, podrás generar claves de aplicación.**

**🔑 2. Genera una clave de aplicación**

1. **Ve a 👉 https://myaccount.google.com/apppasswords**
   * **Si te pide iniciar sesión otra vez, hazlo.**
   * **Solo aparecerá si ya tienes activa la verificación en dos pasos.**
2. **En la opción "Seleccionar aplicación", elige Correo.**
3. **En "Seleccionar dispositivo", puedes poner un nombre como: "Script Python".**
4. **Haz clic en Generar.**
5. **Google te mostrará una contraseña de 16 caracteres, como:**

**nginx**

**CopiarEditar**

**abcd efgh ijkl mnop**

**✅ Guárdala: es lo que debes poner como CLAVE en tu script.**

**📌 Ejemplo en el código:**

**python**

**CopiarEditar**

**REMITENTE = "tu\_correo@gmail.com"**

**CLAVE = "abcdefghi1234567" # La clave de aplicación, sin espacios**

**DESTINATARIO = "cliente@ejemplo.com"**