**Informe del Trabajo Desarrollado**

**Portada del Trabajo:**

**Título del Trabajo**:  
Desarrollo de una Aplicación Web Interactiva para la Visualización y Análisis de Datos

**Asignatura**:  
[Nombre de la Asignatura]

**Integrantes**:  
[Nombre del Estudiante 1]  
[Nombre del Estudiante 2]  
... (Añadir todos los nombres de los integrantes)

**Fecha**:  
[Fecha de entrega]

**Docente**:  
[Nombre del Docente]

**Planteamiento del Problema y Objetivo**

**Planteamiento del Problema:**

En el presente trabajo, se ha desarrollado una **aplicación web interactiva** utilizando **Streamlit** para visualizar y analizar datos de una base obtenida desde una API. La aplicación permite al usuario cargar los datos, realizar análisis estadísticos básicos (media, mediana, desviación estándar), y visualizar los datos a través de diferentes tipos de gráficos (barra, dispersión, línea, histograma y pastel). Además, la aplicación permite filtrar los datos por rango y descargar tanto los datos filtrados como los gráficos generados.

**Objetivo:**

El objetivo principal de este trabajo es diseñar e implementar una aplicación web que permita a los usuarios interactuar con los datos de manera intuitiva y dinámica, proporcionando análisis estadísticos básicos, visualizaciones interactivas, y la capacidad de descargar los resultados generados.

**Definición Teórica del Análisis a Realizar**

**Descripción del archivo de datos:**

El archivo de datos utilizado proviene de una **API** que proporciona información sobre diferentes aspectos de países, como **población**, **área**, **fronteras**, **idiomas** y **zonas horarias**. Las principales variables utilizadas en el análisis incluyen:

* **Población**: Número de habitantes en cada país.
* **Área**: Superficie territorial de cada país.
* **Fronteras**: Cantidad de fronteras terrestres de cada país.
* **Idiomas**: Idiomas oficiales hablados en cada país.
* **Zonas Horarias**: Número de zonas horarias que cubre cada país.

**Funciones clave de Python utilizadas:**

En el desarrollo de esta aplicación, se utilizaron varias funciones clave de Python y bibliotecas que no fueron abordadas en clase, como:

* **pandas**: Para manejar y manipular datos en formato **DataFrame**.
* **matplotlib**: Para generar gráficos estáticos como gráficos de barras, dispersión, histograma y pastel.
* **streamlit**: Para crear la interfaz web interactiva, que permite a los usuarios seleccionar columnas de datos, ajustar rangos para los ejes y generar gráficos en tiempo real.
* **st.cache\_data**: Para almacenar en caché los datos y mejorar el rendimiento de la aplicación evitando cargas repetidas de la API.
* **st.download\_button**: Para permitir a los usuarios descargar los datos filtrados y los gráficos generados en formato CSV y PNG, respectivamente.

**Programa en Python Desarrollado**

A continuación, se muestra el código desarrollado en **Python** para la aplicación web interactiva:

**Imagen 1: Código de carga de datos y preprocesamiento**  
*Pie de figura*: Carga de datos desde la API y preprocesamiento de la información.

python

Copy code

@st.cache\_data

def load\_data():

data = get\_data\_from\_api()

df = preprocess\_data(data)

return df

**Imagen 2: Código para la generación de gráficos interactivos**  
*Pie de figura*: Generación de gráficos según la selección del usuario, con opción de elegir entre varios tipos de gráficos.

python

Copy code

if tipo\_grafico == "Barra":

df\_filtrado[columna\_x].value\_counts().plot(kind='bar', ax=ax, color='skyblue', edgecolor='black')

**Imagen 3: Código para descargar gráficos en PNG**  
*Pie de figura*: Función que permite descargar los gráficos generados en formato PNG.

python

Copy code

def download\_graph(fig):

img\_buffer = io.BytesIO()

fig.savefig(img\_buffer, format="png")

img\_buffer.seek(0)

st.download\_button(

label="Descargar gráfico como PNG",

data=img\_buffer,

file\_name="grafico.png",

mime="image/png"

)

**Resultados**

A continuación se muestran imágenes que capturan las funcionalidades implementadas en la aplicación web:

**Figura 1: Gráfico de barras generado por la aplicación**  
*Pie de figura*: Gráfico de barras que muestra la distribución de valores para la columna seleccionada.

**Figura 2: Gráfico de dispersión generado por la aplicación**  
*Pie de figura*: Gráfico de dispersión que visualiza la relación entre dos columnas seleccionadas.

**Figura 3: Filtro de datos y opción de descarga de gráficos**  
*Pie de figura*: Interfaz para filtrar los datos y la opción para descargar el gráfico generado en formato PNG.

**Conclusiones**

* **Contribución principal**: El trabajo proporcionó una solución interactiva para visualizar, analizar y filtrar datos de manera eficiente a través de una interfaz web sencilla. Los usuarios pueden generar gráficos, analizar estadísticas básicas y descargar los resultados de los gráficos y datos procesados.
* **Desafíos enfrentados**: Algunos de los desafíos incluyeron la integración de las diferentes bibliotecas de Python (como pandas, matplotlib y streamlit), así como la gestión del almacenamiento en caché para mejorar el rendimiento de la aplicación.
* **Limitaciones encontradas**: Una de las limitaciones fue la capacidad de manejo de grandes volúmenes de datos. Aunque el caché ayuda, en algunos casos el rendimiento puede verse afectado si los datos son muy grandes.
* **Líneas de trabajo futuro**: Se podría mejorar la aplicación añadiendo más tipos de gráficos, análisis avanzados como regresiones o agrupaciones, y optimización del rendimiento con bases de datos más robustas.

**Referencias Bibliográficas**

1. **Streamlit Documentation**. (n.d.). Retrieved from <https://docs.streamlit.io>
2. **Matplotlib Documentation**. (n.d.). Retrieved from <https://matplotlib.org>
3. **Pandas Documentation**. (n.d.). Retrieved from https://pandas.pydata.org