Webapp de Consulta de Datos de Contratación

Descripción

Este proyecto desarrolla una webapp utilizando Streamlit para consultar datos de contratación pública de una entidad específica, empleando datos disponibles en el portal Socrata.

Pasos para Configurar el Proyecto

1. Crear el Ambiente de Trabajo

1. Instalar Python 3.10 o superior.
2. Instalar conda si no está instalado. Puede descargarse desde Conda.
3. Crear y activar el ambiente virtual con conda:

```
conda create --name myenv python=3.10
conda activate myenv
```

2. Gestionar Dependencias con Poetry

1. Instalar Poetry siguiendo las instrucciones en su sitio oficial.
2. Iniciar un nuevo proyecto con Poetry:

```
poetry init
```

3. Añadir las dependencias necesarias:

```
poetry add streamlit pandas requests sodapy
```

3. Gestión del Repositorio en GitHub

1. Crear un repositorio en la organización de GitHub de la clase SP2024MINE.
2. Clonar el repositorio en la máquina local:

```
git clone https://github.com/SP2024MINE/your-repo-name.git
cd your-repo-name
```

3. Añadir y realizar commit de los archivos generados por Poetry:

```
git add .
git commit -m "Initial commit"
git push origin main
```

4. Consultar Datos usando Socrata

1. Seleccionar una entidad pública de interés cuyos datos estén disponibles en el portal Socrata.
2. Consultar los datos de contratación utilizando la librería `sodapy`:

```
from sodapy import Socrata
import pandas as pd

client = Socrata("www.datos.gov", None)
results = client.get("xxxx", limit=1000)  # Reemplazar 'xxxx' con el endpoint específico
df = pd.DataFrame.from_records(results)
```

5. Desarrollo de la Webapp con Streamlit

1. Crear un archivo `app.py` en el proyecto y desarrollar la webapp:

```
import streamlit as st
import pandas as pd
from sodapy import Socrata

st.title("Consulta de Datos de Contratación")

# Configuración de la API Socrata
client = Socrata("www.datos.gov", None)

# Solicitud de datos
@st.cache
def load_data():
    results = client.get("xxxx", limit=1000)  # Reemplazar 'xxxx' con el endpoint específico
    return pd.DataFrame.from_records(results)

df = load_data()

# Mostrar datos
st.write(df)
```

2. **Funcionalidades y Métricas Esperadas** La webapp debe incluir las siguientes funcionalidades y métricas para la evaluación de los datos de contratación:
   - **Filtros de Búsqueda**: Permitir filtrar los datos por rango de fechas, tipo de contrato, entidad contratante y contratista.
   - **Visualización de Datos**: Mostrar tablas interactivas con los datos filtrados.
   - **Gráficos y Métricas**: Incluir gráficos y métricas para una mejor visualización y análisis de los datos. Ejemplos sugeridos:
     - **Gráfico de Barras**: Número de contratos por tipo de contrato.
     - **Gráfico de Líneas**: Evolución del monto total contratado a lo largo del tiempo.
     - **Gráfico Circular (Pie Chart)**: Distribución porcentual de los tipos de contratos.
     - **Métricas**: Monto total contratado, número total de contratos, promedio del monto por contrato.
   - **Análisis de Contratistas**: Listar los contratistas con mayor número de contratos y el monto total contratado.

   Ejemplo de código para agregar gráficos en Streamlit:

```
import streamlit as st
import pandas as pd
import altair as alt

# Configuración de la API Socrata
client = Socrata("www.datos.gov", None)

# Solicitud de datos
```

```python
@st.cache
def load_data():
    results = client.get("xxxx", limit=1000)  # Reemplazar 'xxxx' con el endpoint específico
    return pd.DataFrame.from_records(results)

df = load_data()

# Filtros de Búsqueda
st.sidebar.header('Filtros')
start_date = st.sidebar.date_input('Fecha de inicio', pd.to_datetime('2020-01-01'))
end_date = st.sidebar.date_input('Fecha de fin', pd.to_datetime('2021-01-01'))
df_filtered = df[(df['fecha'] >= start_date) & (df['fecha'] <= end_date)]

# Visualización de Datos
st.write(df_filtered)

# Gráficos y Métricas
st.header('Métricas')
total_amount = df_filtered['monto'].sum()
total_contracts = len(df_filtered)
avg_amount = df_filtered['monto'].mean()
st.metric('Monto Total Contratado', f'${total_amount:,.2f}')
st.metric('Número Total de Contratos', total_contracts)
st.metric('Promedio del Monto por Contrato', f'${avg_amount:,.2f}')

st.header('Gráficos')
# Gráfico de Barras
bar_chart = alt.Chart(df_filtered).mark_bar().encode(
    x='tipo_contrato',
    y='count()',
    color='tipo_contrato'
)
st.altair_chart(bar_chart, use_container_width=True)

# Gráfico de Líneas
line_chart = alt.Chart(df_filtered).mark_line().encode(
    x='fecha',
    y='sum(monto)'
)
st.altair_chart(line_chart, use_container_width=True)

# Gráfico Circular (Pie Chart)
pie_chart = alt.Chart(df_filtered).mark_arc().encode(
    theta=alt.Theta(field='monto', type='quantitative'),
    color=alt.Color(field='tipo_contrato', type='nominal')
)
st.altair_chart(pie_chart, use_container_width=True)
```

6. Montaje en GitHub

1. Versionar correctamente el proyecto y realizar commits frecuentes.
2. Documentar el proyecto con un `README.md` detallado.
3. Realizar el push del proyecto final a la organización en GitHub:

```
git add .
git commit -m "Final version of the project"
git push origin main
```

Entrega y Presentación

1. Entregar el link del repositorio de GitHub.
2. Preparar la presentación de la webapp en clase, explicando cada uno de los componentes y el proceso de desarrollo.

Evaluación

La evaluación se centrará en:

- La calidad del código.
- La funcionalidad de la webapp.
- La correcta gestión de dependencias y ambientes.
- El uso adecuado de GitHub.

Releases

No releases published

Packages

No packages published