

## TAREA MÓDULO 8 – PYTHON- LUCIO ALEJANDRO MOUZO MONTEAGUDO

El enlace a la app es el siguiente <https://appluciomouzo.streamlit.app/>

En el siguiente documento voy a explicar la tarea del módulo 8 que he realizado, que consiste en la creación de una página web mediante Streamlit y Python.

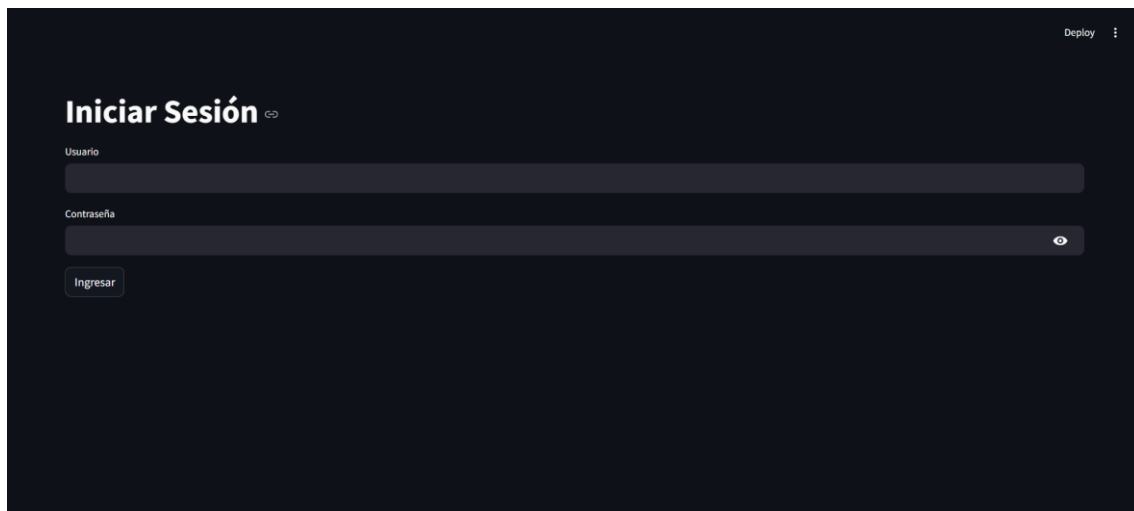
En la página web creada vamos a encontrar distintas ligas de primera división y después unas comparativas entre los equipos de una misma liga y una comparativa entre los equipos, sin influencia de la liga en la que compitan.

El objetivo de la tarea es aplicar los conocimientos adquiridos sobre desarrollo de aplicaciones web utilizando Streamlit para crear una aplicación interactiva que conecte múltiples fuentes de datos, con funcionalidades avanzadas como gestión de sesiones y caché.

Lo primero que vamos a hacer es mencionar cada uno de los apartados, implementando las menciones añadiremos unas capturas de pantalla de los códigos y de las partes referenciadas en la página web.

El primer apartado es el Login y control de sesión, en el que hemos añadido una autenticación a través de usuario y contraseña. También tenemos un botón para cerrar sesión.

```
24 # Estado de sesión
25 if "logged_in" not in st.session_state:
26     st.session_state["logged_in"] = False
27
28 # Función de Login
29 def login():
30     st.title("Iniciar Sesión")
31     username = st.text_input("Usuario", key="username_input")
32     password = st.text_input("Contraseña", type="password", key="password_input")
33     if st.button("Ingresar"):
34         if username == "admin" and password == "admin":
35             st.session_state["logged_in"] = True
36             st.success("Inicio de sesión exitoso")
37         else:
38             st.error("Usuario o contraseña incorrectos. Intenta nuevamente.")
39
```



El siguiente apartado es el menú y navegación, donde hemos desarrollado un menú lateral en el que tenemos varias pestañas, con la página de actualizar los datos, la comparación

de datos entre los equipos de las mismas ligas, la comparación de datos entre los equipos de distintas ligas.

```
264 def main():
265
266     def login():
267         st.title("Iniciar Sesión")
268         username = st.text_input("Usuario", key="username_input")
269         password = st.text_input("Contraseña", type="password", key="password_input")
270         if st.button("Ingresar"):
271             if username == "admin" and password == "admin":
272                 st.session_state["logged_in"] = True
273                 st.success("Inicio de sesión exitoso")
274             else:
275                 st.error("Usuario o contraseña incorrectos. Intenta nuevamente.")
276
277     if st.session_state["logged_in"]:
278         st.sidebar.title("Menú de Navegación")
279         opciones = [
280             "Inicio",
281             "Actualizar Datos",
282             "Visualizar Ligas",
283             "Comparación dentro de una Liga",
284             "Comparación entre Ligas",
285             "Cerrar Sesión"
286         ]
287         eleccion = st.sidebar.radio("Selecciona una opción", opciones)
```



El tercer punto es la creación de una página con conexión a fuentes de datos, donde hemos utilizado el Web Scraping para obtener datos de la API de Football Data, en la que necesitaremos una clave para poder entrar.

```

40 # Función para obtener las ligas por webscraping
41 def obtener_datos_webscraping(headers, guardar_csv=True):
42     # Obtener todas las ligas
43     url_ligas = "https://api.football-data.org/v4/competitions"
44     response_ligas = requests.get(url_ligas, headers=headers, timeout=10)
45
46     if response_ligas.status_code != 200:
47         st.error(f"Error (response_ligas.status_code): {response_ligas.text}")
48         return pd.DataFrame()
49
50     # Filtrar ligas nacionales y excluir selecciones
51     ligas = []
52     for liga in response_ligas.json().get('competitions', []):
53         if liga.get('type') == "LEAGUE" and liga['name'] not in ["European Championship", "FIFA World Cup"]
54     ]
55
56     clasificaciones = pd.DataFrame()
57
58     # Iterar sobre cada liga y obtener su clasificación
59     for liga in ligas:
60         liga_id = liga['id']
61         url_clasificacion = f"https://api.football-data.org/v4/competitions/{liga_id}/standings"
62
63         for intento in range(3): # Reintentos en caso de fallo
64             try:
65                 response_clasificacion = requests.get(url_clasificacion, headers=headers, timeout=10)
66                 if response_clasificacion.status_code == 200:
67                     standings_data = response_clasificacion.json()
68                     if 'standings' in standings_data and standings_data['standings']:

```

```

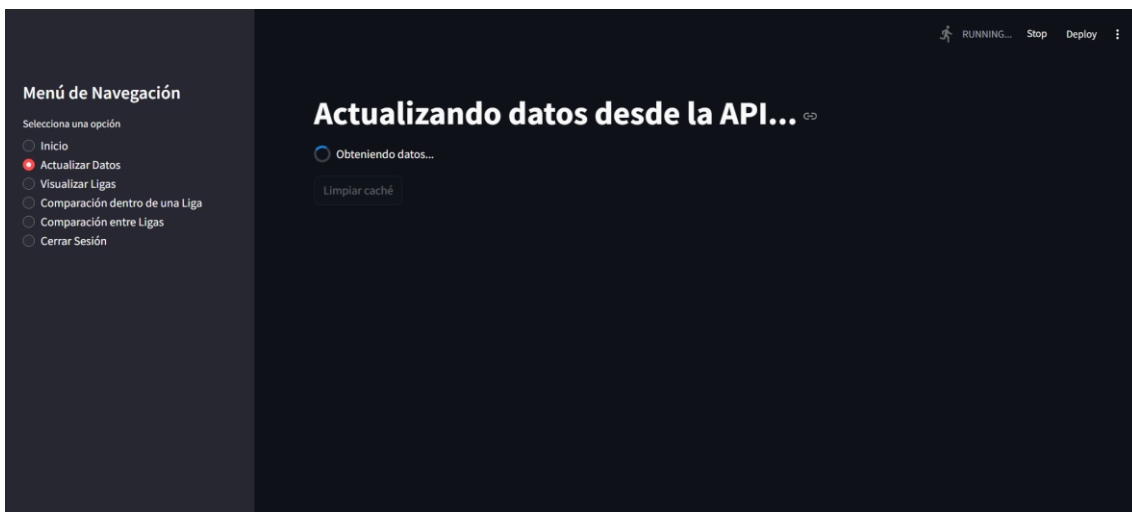
75         'Partidos Jugados': team['playedGames'],
76         'Ganados': team['won'],
77         'Empatados': team['draw'],
78         'Perdidos': team['lost'],
79         'Goles a Favor': team['goalsFor'],
80         'Goles en Contra': team['goalsAgainst'],
81         'Diferencia de Goles': team['goalDifference']
82     } for team in standings])
83
84     clasificaciones = pd.concat([clasificaciones, clasificacion_df], ignore_index=True)
85     break # Salir del bucle de reintentos si tiene éxito
86
87     else:
88         st.warning(f"No se encontraron datos de clasificación para la liga {liga['name']}")
89
90     except requests.exceptions.RequestException as e:
91         st.warning(f"Error (response_clasificacion.status_code) al obtener datos de la liga {liga['name']}")
92         st.warning(f"Error en la solicitud para la liga {liga['name']}: {e}")
93         time.sleep(2)
94
95     # Respetar los límites de la API
96     time.sleep(5)
97
98     # Procesar datos
99     if not clasificaciones.empty:
100         clasificaciones['Efectividad'] = (clasificaciones['Puntos'] / (clasificaciones['Partidos Jugados'] * 3))
101         clasificaciones['Promedio Goles a Favor'] = clasificaciones['Goles a Favor'] / clasificaciones['Partidos Jugados']
102         clasificaciones['Promedio Goles en Contra'] = clasificaciones['Goles en Contra'] / clasificaciones['Partidos Jugados']
103         clasificaciones['Diferencia de Goles por Partido'] = clasificaciones['Diferencia de Goles'] / clasificaciones['Partidos Jugados']

```

```

97     # Procesar datos
98     if not clasificaciones.empty:
99         clasificaciones['Efectividad'] = (clasificaciones['Puntos'] / (clasificaciones['Partidos Jugados'] * 3))
100         clasificaciones['Promedio Goles a Favor'] = clasificaciones['Goles a Favor'] / clasificaciones['Partidos Jugados']
101         clasificaciones['Promedio Goles en Contra'] = clasificaciones['Goles en Contra'] / clasificaciones['Partidos Jugados']
102         clasificaciones['Diferencia de Goles por Partido'] = clasificaciones['Diferencia de Goles'] / clasificaciones['Partidos Jugados']
103
104     # Guardar en CSV si se solicita
105     if guardar_csv:
106         clasificaciones.to_csv('clasificacion_ligas_procesadas.csv', index=False)
107         st.success("Datos guardados en 'clasificacion_ligas_procesadas.csv'.")
108
109     return clasificaciones
110

```



El cuarto apartado es la visualización y funcionalidades interactivas, en las que hemos utilizado las tablas de cada una de las ligas, en las que podemos escoger las variables que

nos interesan, también tenemos un gráfico de dispersión y un radar plot. También lo podemos realizar entre equipos de distintas ligas.

```
111
112 # Función de comparación dentro de una misma Liga
113 def comparacion_dentro_liga(df):
114     st.title("Comparación dentro de una Liga")
115
116     # Seleccionar liga
117     ligas = df["Liga"].unique()
118     seleccion_liga = st.selectbox("Selecciona una Liga:", ligas)
119
120     if seleccion_liga:
121         df_liga = df[df["Liga"] == seleccion_liga]
122
123         # Tabla personalizada
124         st.subheader(f"Tabla de Clasificación - {seleccion_liga}")
125         columnas_disponibles = df_liga.columns.tolist()
126         columnas_seleccionadas = st.multiselect(
127             "Selecciona las columnas que quieres visualizar:",
128             options=columnas_disponibles,
129             default=columnas_disponibles
130         )
131         st.dataframe(df_liga[columnas_seleccionadas])
132
133         # Gráfico de dispersión
134         st.subheader("Gráfico de Dispersión")
135         x_columna = st.selectbox("Selecciona la columna para el eje X:", options=df_liga.columns)
136         y_columna = st.selectbox("Selecciona la columna para el eje Y:", options=df_liga.columns)
137         fig, ax = plt.subplots()
138         ax.scatter(df_liga[x_columna], df_liga[y_columna], alpha=0.7)
139
140         ax.set_title(f"{x_columna} vs {y_columna}")
141         ax.set_xlabel(x_columna)
142         ax.set_ylabel(y_columna)
143         st.pyplot(fig)
144
145         # Radar plot
146         st.subheader("Radar Plot Comparativo")
147         equipos_seleccionados = st.multiselect(
148             "Selecciona los equipos para comparar:",
149             options=df_liga["Equipo"].unique(),
150             default=df_liga["Equipo"].unique()[:3]
151         )
152         columnas_radar = st.multiselect(
153             "Selecciona las métricas para el radar:",
154             options=[col for col in df_liga.columns if df_liga[col].dtype in ['int64', 'float64']],
155             default=[col for col in df_liga.columns if df_liga[col].dtype in ['int64', 'float64'][:5]
156         )
157
158         if equipos_seleccionados and columnas_radar:
159             df_radar = df_liga[df_liga["Equipo"].isin(equipos_seleccionados)][["Equipo"] + columnas_radar]
160             df_radar.set_index("Equipo", inplace=True)
161
162             num_vars = len(columnas_radar)
163             angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
164             angles += angles[:1]
165
166             fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
```

```

157         if equipos_seleccionados and columnas_radar:
158             df_radar = df_liga[df_liga["Equipo"].isin(equipos_seleccionados)][["Equipo"] + columnas_radar]
159             df_radar.set_index("Equipo", inplace=True)
160
161             num_vars = len(columnas_radar)
162             angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
163             angles += angles[:1]
164
165             fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
166             for equipo in df_radar.index:
167                 valores = df_radar.loc[equipo].values.flatten().tolist()
168                 valores += valores[:1]
169                 ax.plot(angles, valores, label=equipo)
170                 ax.fill(angles, valores, alpha=0.25)
171
172             ax.set_xticks(angles[:-1])
173             ax.set_xticklabels(columnas_radar)
174             ax.legend(loc="upper right", bbox_to_anchor=(1.1, 1.1))
175             st.pyplot(fig)
176

```

```

178 # Función de comparación entre ligas
179 def comparacion_entre_equipos(df):
180     st.title("Comparación entre Equipos de Distintas Ligas")
181
182     # Filtrar solo las columnas numéricas
183     columnas_numericas = df.select_dtypes(include=['int64', 'float64']).columns
184
185     # Seleccionar equipos
186     equipos_seleccionados = st.multiselect(
187         "Selecciona los equipos para comparar:",
188         options=df["Equipo"].unique(),
189         default=df["Equipo"].unique()[:3] # Mostrar los 3 primeros por defecto
190     )
191
192     # Seleccionar métricas para comparar
193     columnas_seleccionadas = st.multiselect(
194         "Selecciona las métricas para la comparación:",
195         options=columnas_numericas,
196         default=columnas_numericas[:5]
197     )
198
199     # Filtrar el DataFrame
200     if equipos_seleccionados and columnas_seleccionadas:
201         df_comparacion = df[df["Equipo"].isin(equipos_seleccionados)][["Equipo", "Liga"] + columnas_seleccionadas]
202         st.subheader("Tabla Comparativa de Equipos")
203         st.dataframe(df_comparacion)
204
205     # Gráfico de dispersión
206

```

```

205     # Gráfico de dispersión
206     st.subheader("Gráfico de Dispersión entre Equipos")
207     x_columna = st.selectbox("Selecciona la columna para el eje X:", options=columnas_seleccionadas)
208     y_columna = st.selectbox("Selecciona la columna para el eje Y:", options=columnas_seleccionadas)
209
210     fig, ax = plt.subplots()
211     for equipo in equipos_seleccionados:
212         datos_equipo = df_comparacion[df_comparacion["Equipo"] == equipo]
213         ax.scatter(
214             datos_equipo[x_columna],
215             datos_equipo[y_columna],
216             label=equipo,
217             s=100 # Tamaño de los puntos
218         )
219         ax.text(
220             datos_equipo[x_columna].values[0],
221             datos_equipo[y_columna].values[0],
222             equipo,
223             fontsize=9,
224             ha="right"
225         )
226     ax.set_title(f"{x_columna} vs {y_columna} (Comparación entre Equipos)")
227     ax.set_xlabel(x_columna)
228     ax.set_ylabel(y_columna)
229     ax.legend(title="Equipos")
230     st.pyplot(fig)

```

```
232 # Crear el radar plot
233 st.subheader("Radar Plot Comparativo")
234 df_radar = df_comparacion.set_index("Equipo")[columnas_seleccionadas]
235 num_vars = len(columnas_seleccionadas)
236 angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
237 angles += angles[:1]
238
239 fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
240 for equipo in df_radar.index:
241     valores = df_radar.loc[equipo].values.flatten().tolist()
242     valores += valores[:1]
243     ax.plot(angles, valores, label=equipo)
244     ax.fill(angles, valores, alpha=0.25)
245
246 ax.set_xticks(angles[:-1])
247 ax.set_xticklabels(columnas_seleccionadas)
248 ax.legend(loc="upper right", bbox_to_anchor=(1.1, 1.1))
249 st.pyplot(fig)
250
251 else:
252     st.warning("Selecciona al menos un equipo y una métrica para generar la comparación.")
253
```

Menú de Navegación

Selecciona una opción

Inicio

Actualizar Datos

Visualizar Ligas

Comparación dentro de una Liga

Comparación entre Ligas

Cerrar Sesión

Deploy

Selecciona una liga para visualizar

Campeonato Brasileiro Série A

Mostrando datos para la liga: Campeonato Brasileiro Série A

	Equipo	Liga	Posición	Puntos	Partidos Jugados	Ganados	Empatados	Perdidos	Goles a Favor	Goles en Con
0	Botafogo FR	Campeonato Brasileiro Série A	1	79	38	23	10	5	59	
1	SE Palmeiras	Campeonato Brasileiro Série A	2	73	38	22	7	9	60	
2	CR Flamengo	Campeonato Brasileiro Série A	3	70	38	20	10	8	61	
3	Fortaleza EC	Campeonato Brasileiro Série A	4	68	38	19	11	8	53	
4	SC Internacional	Campeonato Brasileiro Série A	5	65	38	18	11	9	53	
5	São Paulo FC	Campeonato Brasileiro Série A	6	59	38	17	8	13	53	
6	SC Corinthians Paulista	Campeonato Brasileiro Série A	7	56	38	15	11	12	54	
7	EC Bahia	Campeonato Brasileiro Série A	8	53	38	15	8	15	49	
8	Cruzeiro EC	Campeonato Brasileiro Série A	9	52	38	14	10	14	43	
9	CR Vasco da Gama	Campeonato Brasileiro Série A	10	50	38	14	8	16	43	

Menú de Navegación

Selecciona una opción

Inicio

Actualizar Datos

Visualizar Ligas

Comparación dentro de una Liga

Comparación entre Ligas

Cerrar Sesión

Deploy

Comparación dentro de una Liga

Selecciona una Liga:

Campeonato Brasileiro Série A

Tabla de Clasificación - Campeonato Brasileiro Série A

Selecciona las columnas que quieres visualizar:

Equipo

Liga

Posición

Puntos

Partidos Jugados

Ganados

Empatados

Perdidos

Goles a Favor

Goles en Contra

Diferencia de Go...

Efectividad

Promedio Goles ...

Promedio Goles ...

Diferencia de Go...

	Equipo	Liga	Posición	Puntos	Partidos Jugados	Ganados	Empatados	Perdidos	Goles a Favor	Goles en Con
0	Botafogo FR	Campeonato Brasileiro Série A	1	79	38	23	10	5	59	
1	SE Palmeiras	Campeonato Brasileiro Série A	2	73	38	22	7	9	60	
2	CR Flamengo	Campeonato Brasileiro Série A	3	70	38	20	10	8	61	
3	Fortaleza EC	Campeonato Brasileiro Série A	4	68	38	19	11	8	53	
4	SC Internacional	Campeonato Brasileiro Série A	5	65	38	18	11	9	53	
5	São Paulo FC	Campeonato Brasileiro Série A	6	59	38	17	8	13	53	

**Menú de Navegación**

Selecciona una opción

- ☐ Inicio
- ☐ Actualizar Datos
- ☐ Visualizar Ligas
- ☐ Comparación dentro de una Liga
- ☒ Comparación entre Ligas
- ☐ Cerrar Sesión

## Comparación entre Equipos de Distintas Ligas

Selecciona los equipos para comparar:

Botafogo FR x SE Palmeiras x CR Flamengo x

Selecciona las métricas para la comparación:

Posición x Puntos x Partidos Jugados x Ganados x Empatados x

### Tabla Comparativa de Equipos

	Equipo	Liga	Posición	Puntos	Partidos Jugados	Ganados	Empatados
0	Botafogo FR	Campeonato Brasileiro Série A	1	79	38	23	10
1	SE Palmeiras	Campeonato Brasileiro Série A	2	73	38	22	7
2	CR Flamengo	Campeonato Brasileiro Série A	3	70	38	20	10

### Gráfico de Dispersión entre Equipos

Selecciona la columna para el eje X:

Posición

El último punto que vamos a tratar es la optimización y publicación, donde podemos encontrar la optimización que se ha realizado del código para no tener funciones que no se utilizan en el código, y también se ha incluido un botón para limpiar el caché de los datos.

```
if eleccion == "Inicio":
    st.title("Bienvenido a la Aplicación Deportiva")
    st.write("Navega por el menú para explorar las funcionalidades.")
    if st.button("Limpiar caché"):
        st.cache_data.clear()
        st.success("Caché limpiado correctamente.")
```

**Menú de Navegación**

Selecciona una opción

- ☒ Inicio
- ☐ Actualizar Datos
- ☐ Visualizar Ligas
- ☐ Comparación dentro de una Liga
- ☐ Comparación entre Ligas
- ☐ Cerrar Sesión

## Bienvenido a la Aplicación Deportiva

Navega por el menú para explorar las funcionalidades.

Limpiar caché

El siguiente punto en el que nos vamos a centrar es en la reflexión sobre los desafíos que he enfrentado durante el desarrollo y como han sido resueltos.

Los puntos o desafíos que nos hemos encontrado han sido el uso de Git, el uso de Streamlit Cloud, el código y las visualizaciones.

El uso de Git ha sido un desafío por la implementación y la creación del usuario en el que tendremos todo, y el uso de los distintos comandos para conectar Visual Studio Code con GitHub y de esta forma tener ambas cosas en el medio local y en la red.

Con Streamlit Cloud ha sido algo similar, era una aplicación con la que no tenía experiencia, y con la que hemos utilizado el usuario de GitHub, consiguiendo de este modo la aplicación y que se pueda compartir el enlace con gente que nos interese.

El siguiente punto es el código, en el que se tiene la lógica y la estructura, hemos creado el inicio de sesión, la actualización de los datos y las visualizaciones, teniendo un flujo claro para la aplicación. A esto hay que añadirle que he intentado que tengamos funciones y en la parte de la aplicación, únicamente encontremos las funciones mencionadas, para que a la hora de modificar o realizar mejoras solo tengamos que ir a la función elegida.

El último punto son las visualizaciones que he escogido, que han sido las tablas, los gráficos de dispersión y el radar plot, en lo que nos podemos mover con las variables o estadísticas que más nos influyan estudiar en el momento indicado, pudiendo hacer comparaciones entre equipos de mismas ligas y de distintas ligas.