

Visualización de datos: Streamlit

Alejandro Paredero

paredero@mbitschool.com

<https://es.linkedin.com/in/paredero>

Importancia en la visualización de datos

Ciclo de vida de un proyecto basado en datos

Representación y visualización de los datos, ese gran olvidado.

FASES DEL CICLO DE VIDA DEL DATO. EP



Visualizaciones

Cada tipo de dato tiene una forma adecuada para ser visualizada

Depende de muchos factores:

- Intervalo
- Frecuencia
- Contexto
- Tendencia
- Público
- *Aceptación social*

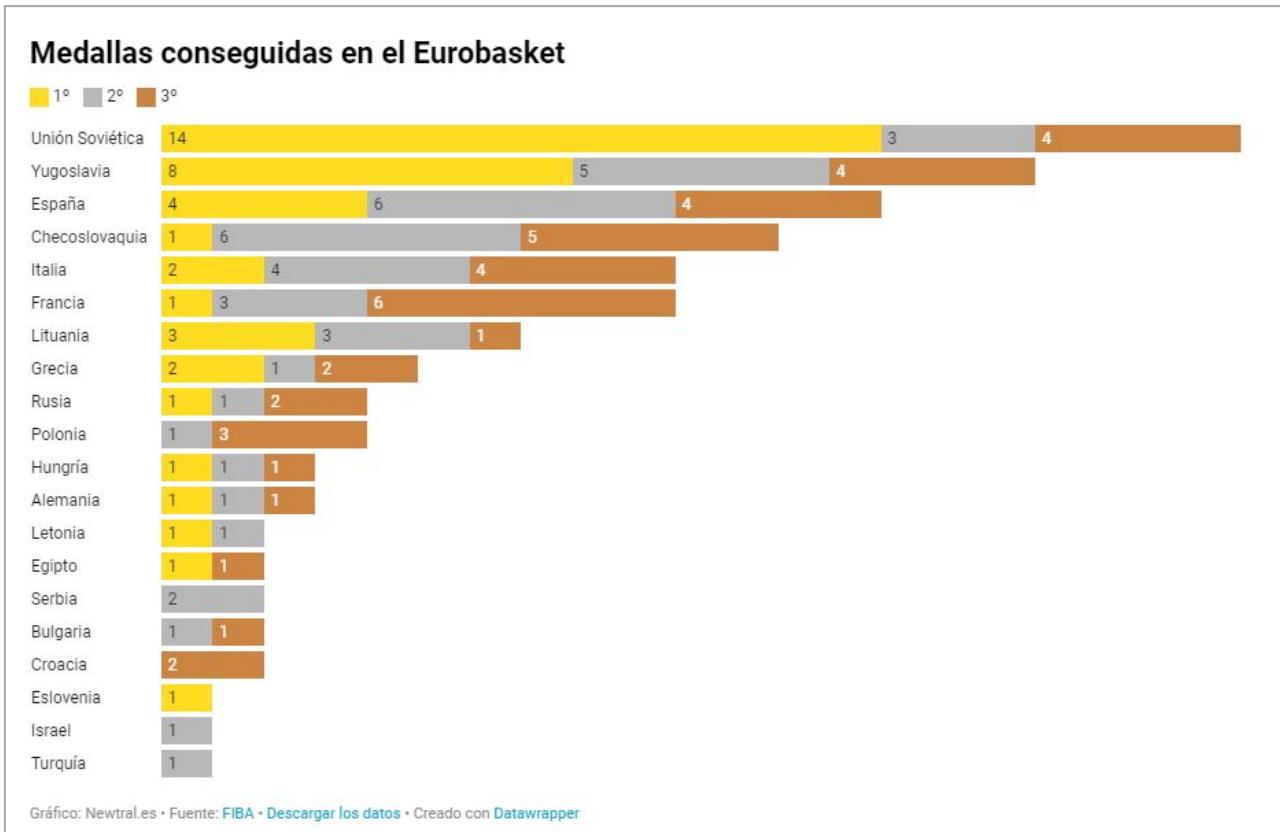


Un orden y representación adecuado de los datos obtenidos ayuda a contar la historia.

Qué tan importante es la visualización de datos

Ventajas de una visualización de datos **correcta**

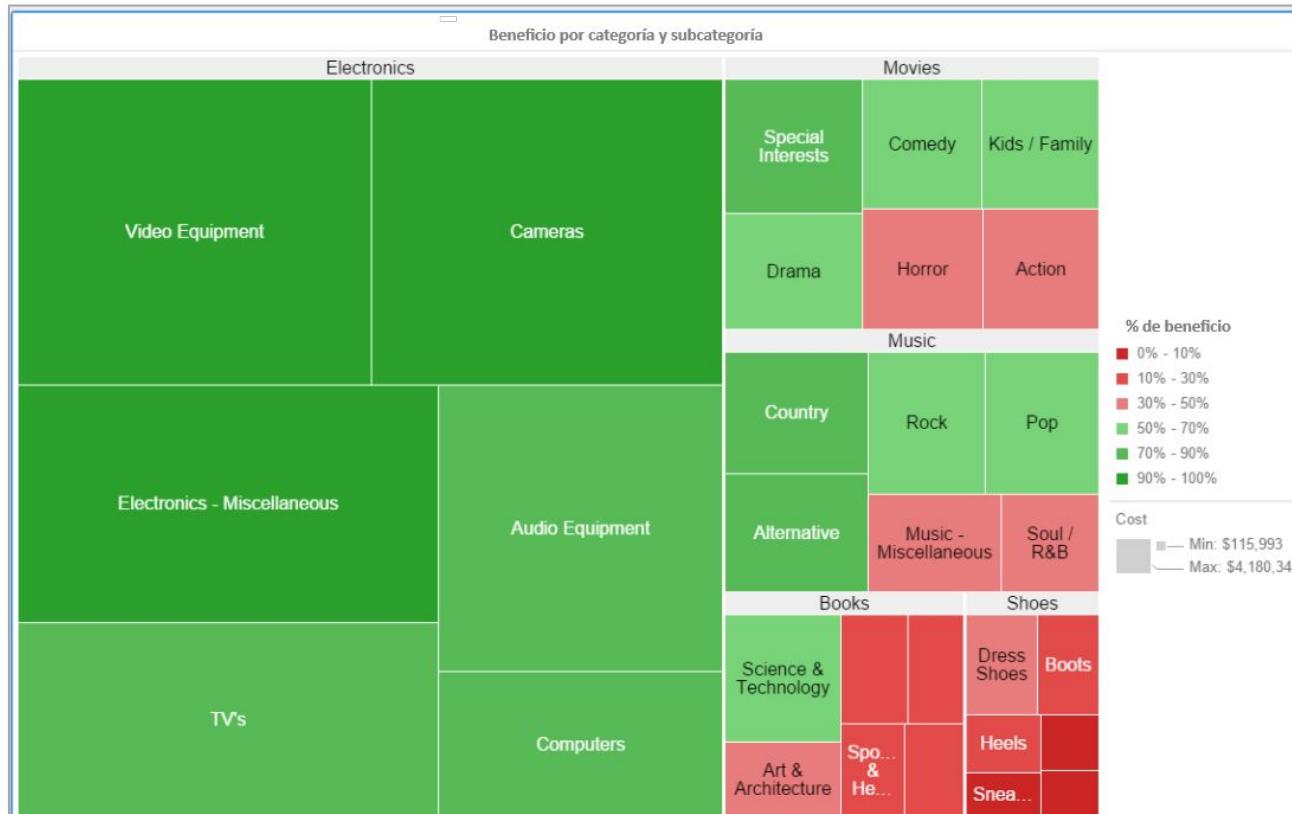
- Facilita la comprensión de los datos



Qué tan importante es la visualización de datos

Ventajas de una visualización de datos **correcta**

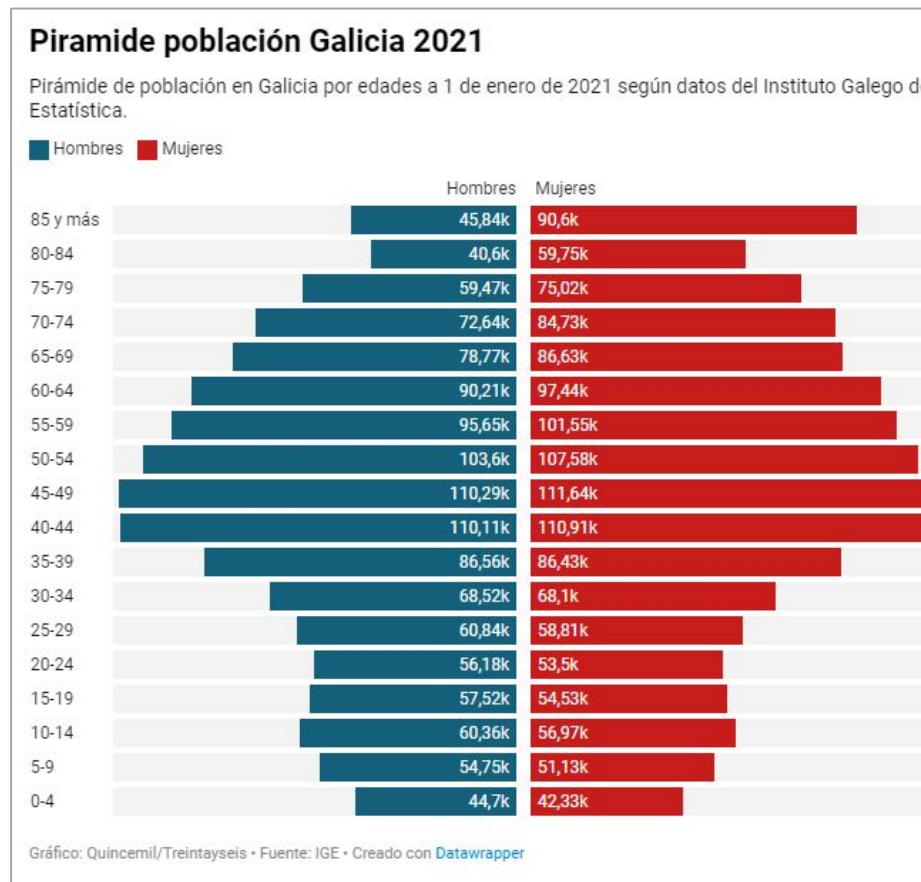
- Mejorar la toma de decisiones



Qué tan importante es la visualización de datos

Ventajas de una visualización de datos **correcta**

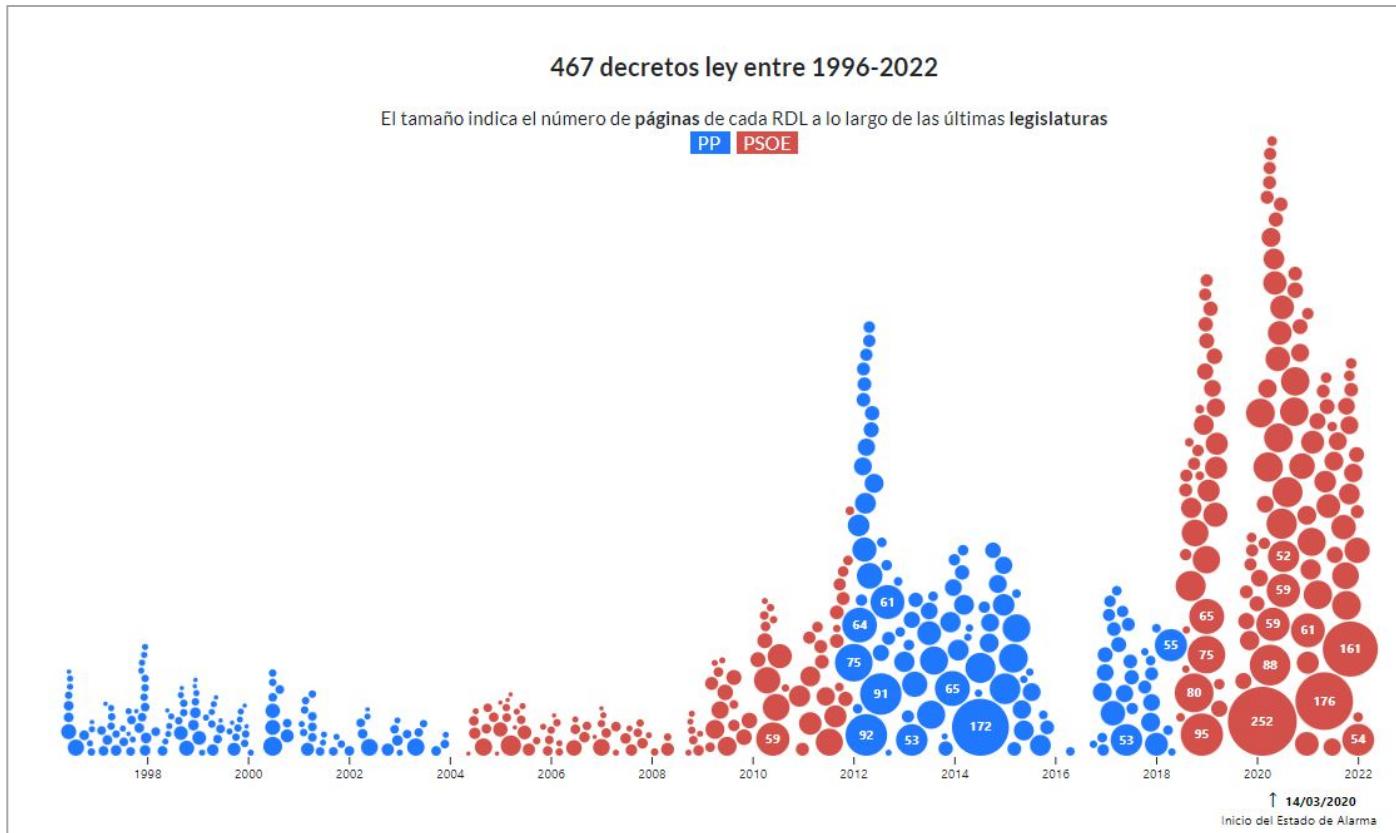
- Identifica tendencias y patrones



Qué tan importante es la visualización de datos

Ventajas de una visualización de datos **correcta**

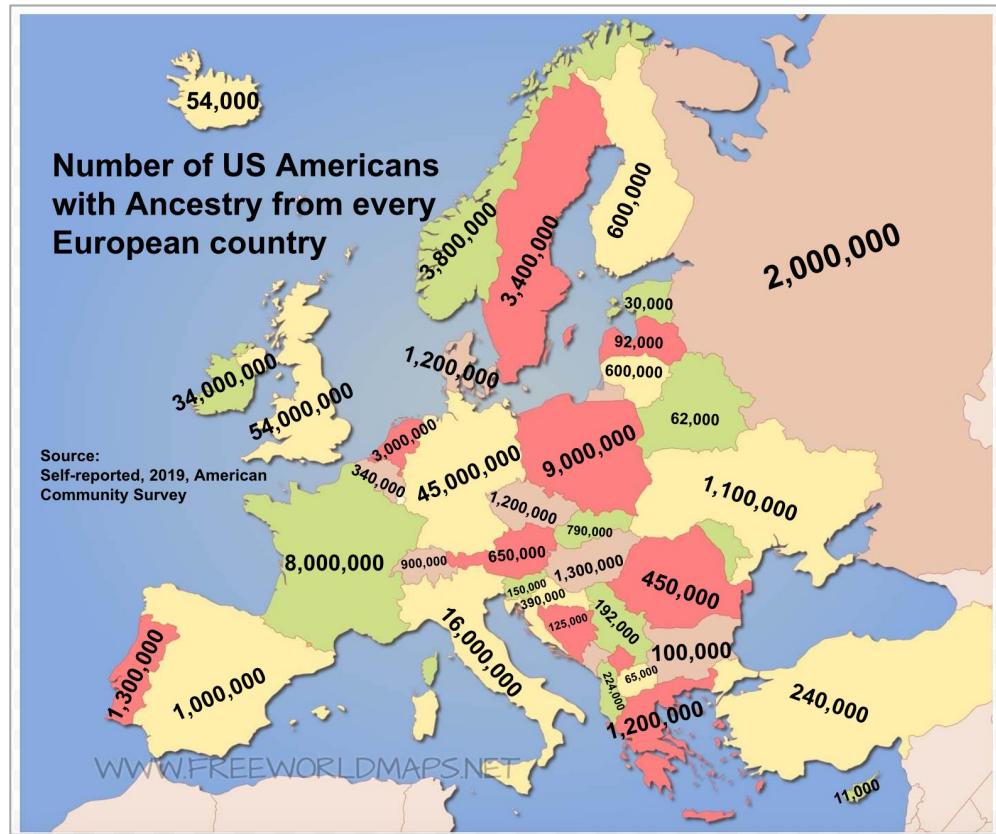
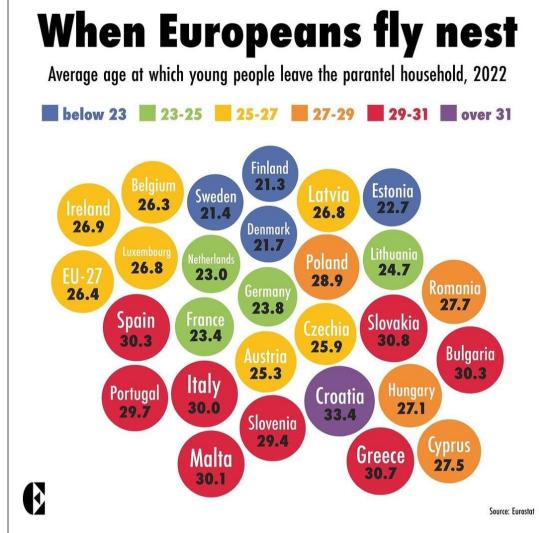
- Aumenta el `engagement visual`



Qué tan importante es la visualización de datos

Qué problemas presenta una visualización de datos **incorrecta**

- Información confusa y engañosa
- Sobrecarga cognitiva
- Falta de contexto
- Uso incorrecto de gráficos



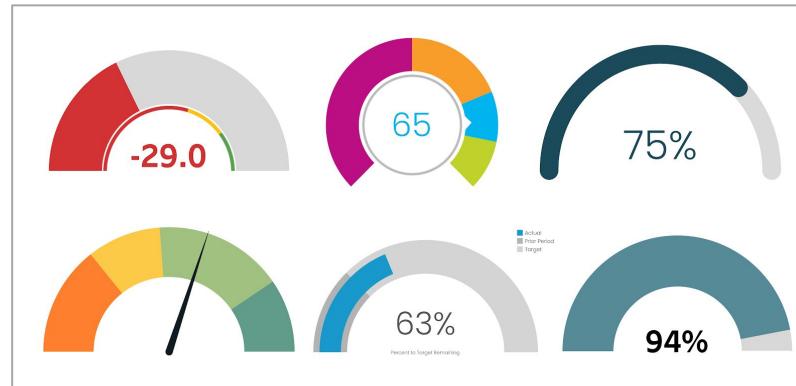
Qué tan importante es la visualización de datos

Para finalizar, un par de consejos:

- Ponte en el lugar del que consumirá tus visualizaciones



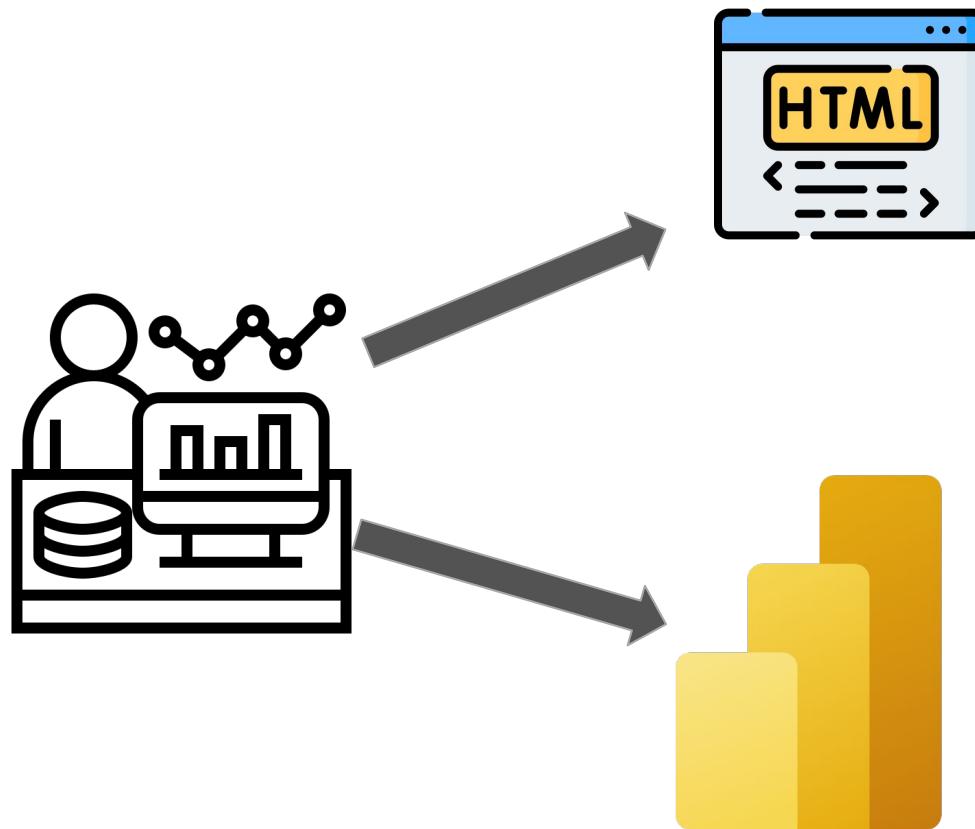
- Busca feedback: Si dudas sobre cómo representar la información, consulta a un compañero.



Transferencia de los datos a negocio

Visualizar los datos: ¿Dónde? Antiguamente...

Entregar un jupyter notebook a negocio está feo:



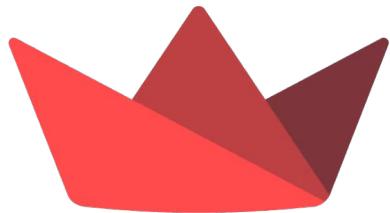
Exportar a HTML / PDF / JPG / PNG

- + Simple de generar
- Es estático
- Con diferente público, generar documentos por separado.

Exportar a Herramientas de BI

- + Es personalizable
- Requiere conocimientos específico herramienta
- Requiere instalar herramientas
- Requiere licencias
- Integraciones no siempre triviales

Bienvenidos a Streamlit



Streamlit

<https://streamlit.io>

¿Qué es Streamlit?

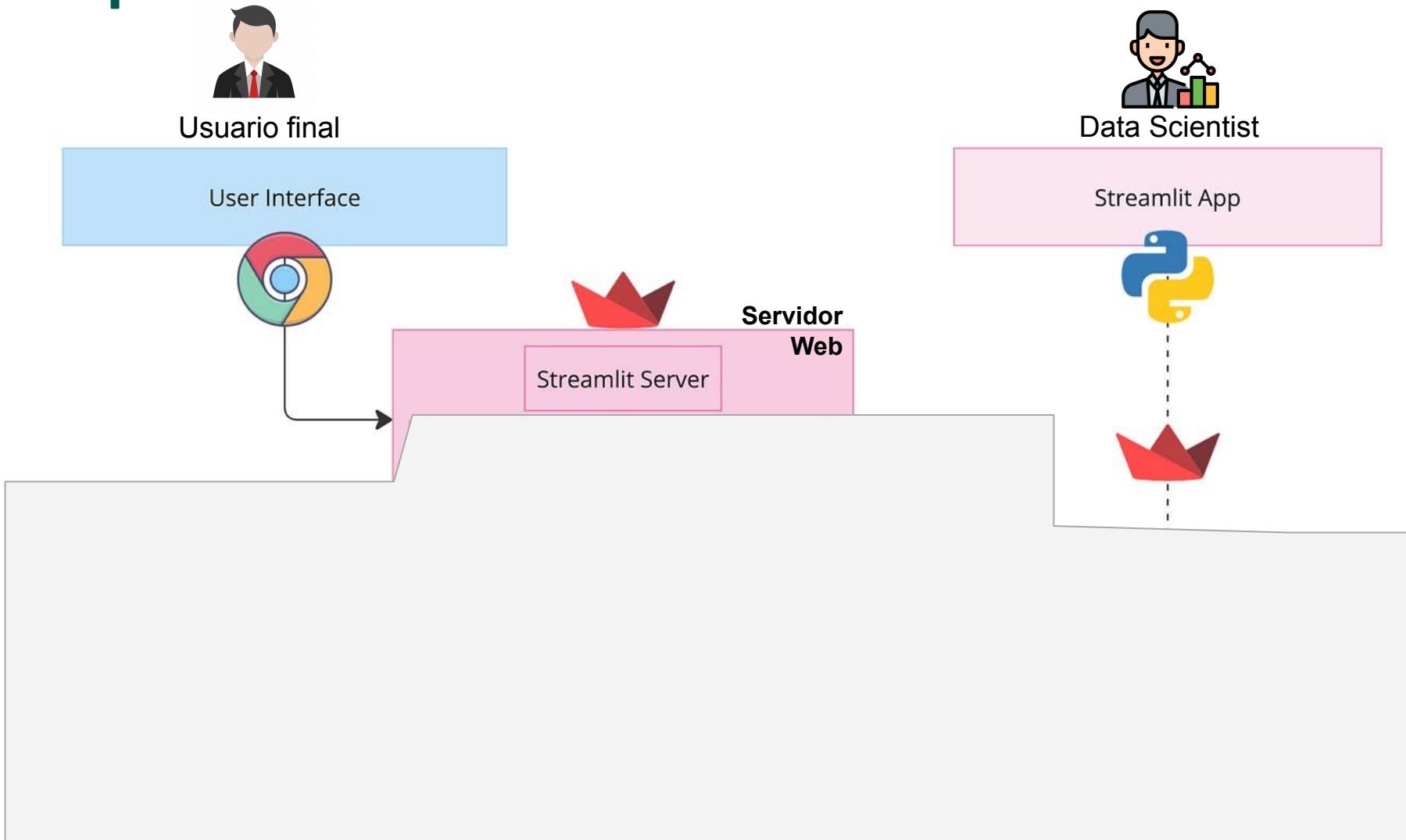
"Framework de aplicaciones de código abierto que ayuda a equipos de ciencia de datos a crear aplicaciones web en cuestión de minutos."

- Desarrollado en **Python** y Javascript
- Bibliotecas compatible 100% python
- Open Source
- Low code*
- Incorpora servidor web
- Fuerte comunidad desarrollo detrás
- Numerosos ejemplos e integraciones



Streamlit

Arquitectura de Streamlit



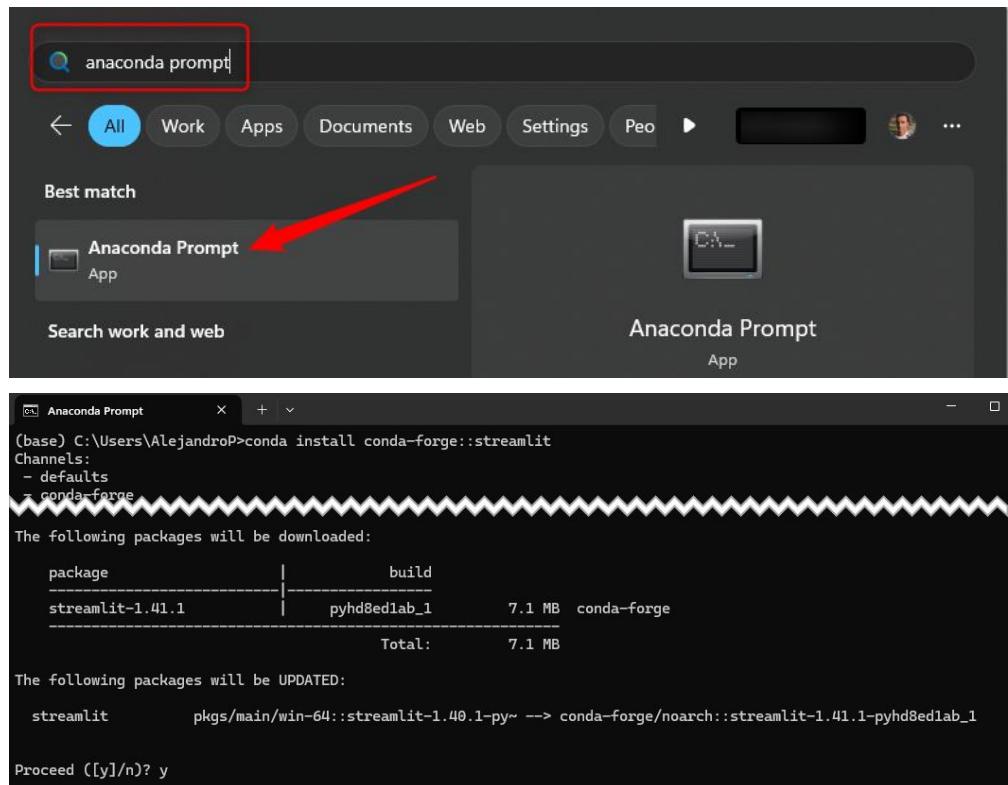
Instalación de Streamlit

Instalación por PIP:

```
conda install conda-forge::streamlit
```

Instalación por Conda (Anaconda prompt):

```
pip install streamlit
```



Otros métodos: <https://docs.streamlit.io/get-started/installation>

Instalación de Streamlit

Comprobamos que streamlit funciona: Ejecuta en **Anaconda prompt**

```
streamlit hello
```

```
(base) C:\Users\AlejandroP>streamlit hello
```

Welcome to Streamlit. Check out our demo in your browser.

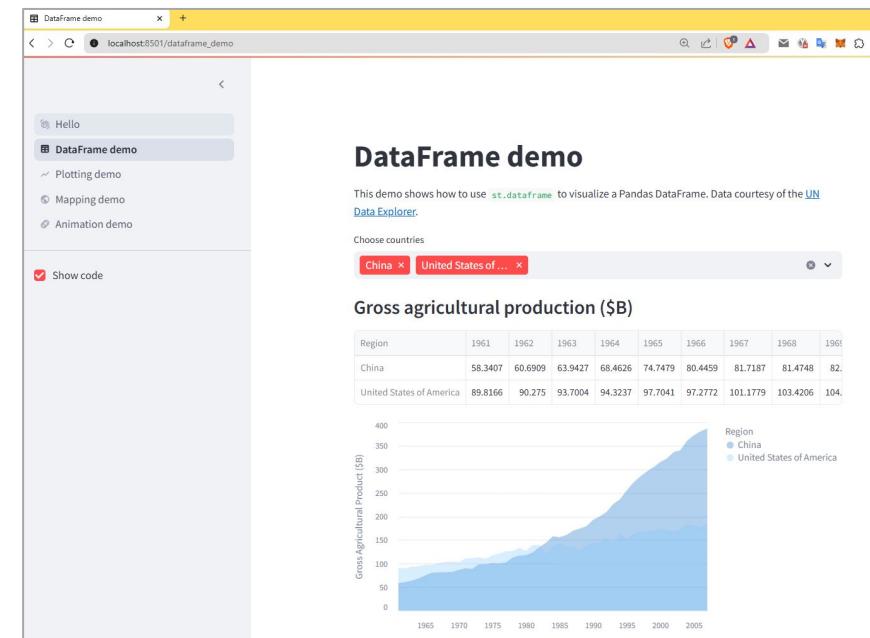
Local URL: <http://localhost:8501>

Network URL: <http://192.168.1.131:8501>

Ready to create your own Python apps super quickly?

Head over to <https://docs.streamlit.io>

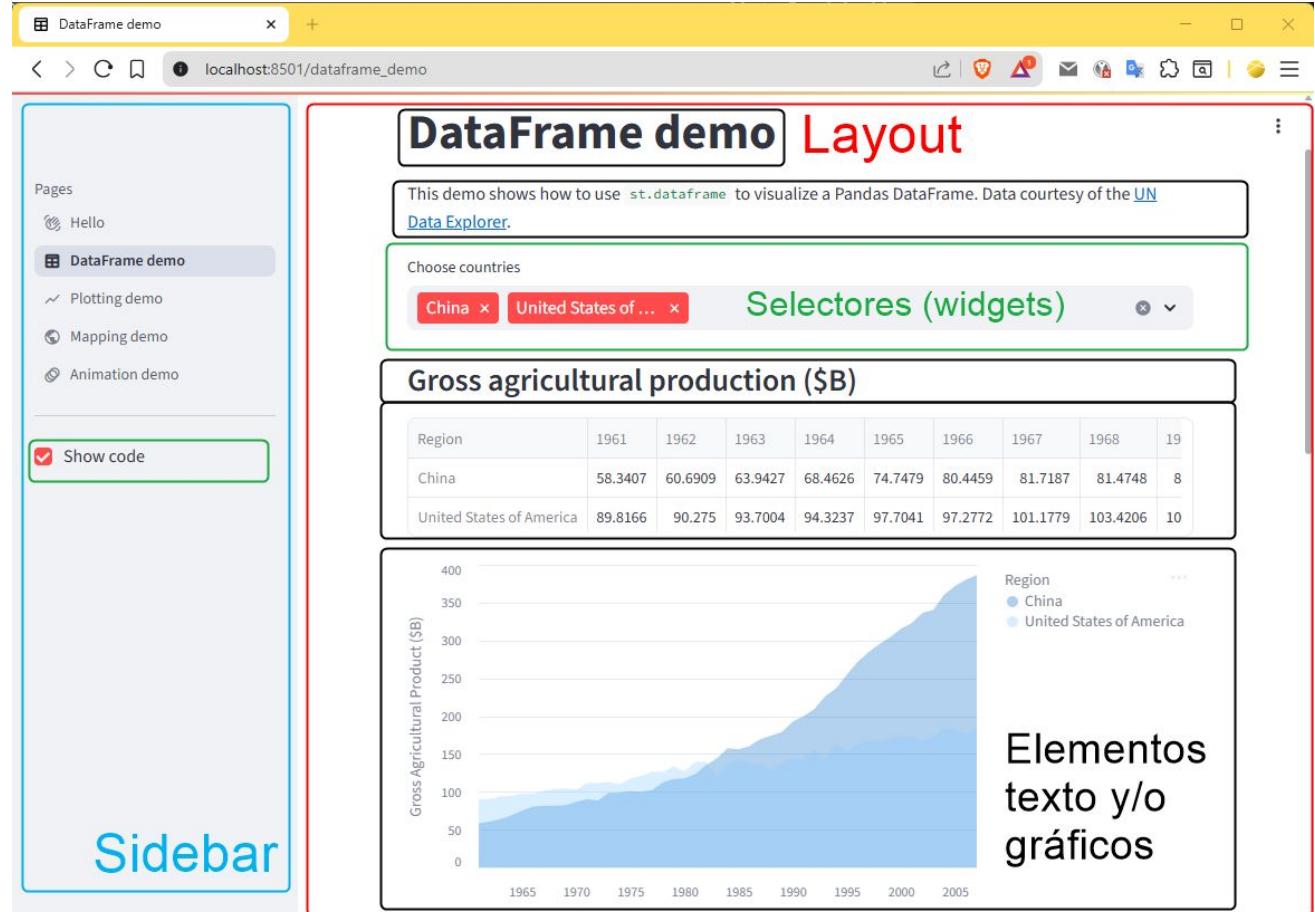
May you create awesome apps!



Componentes de Streamlit

Streamlit tiene cuatro componentes:

- Layout
- Sidebar
- Widgets
- Elementos



Versión de Streamlit

La evolución de Streamlit es muy rápida. Para saber la versión que tenéis instalada:

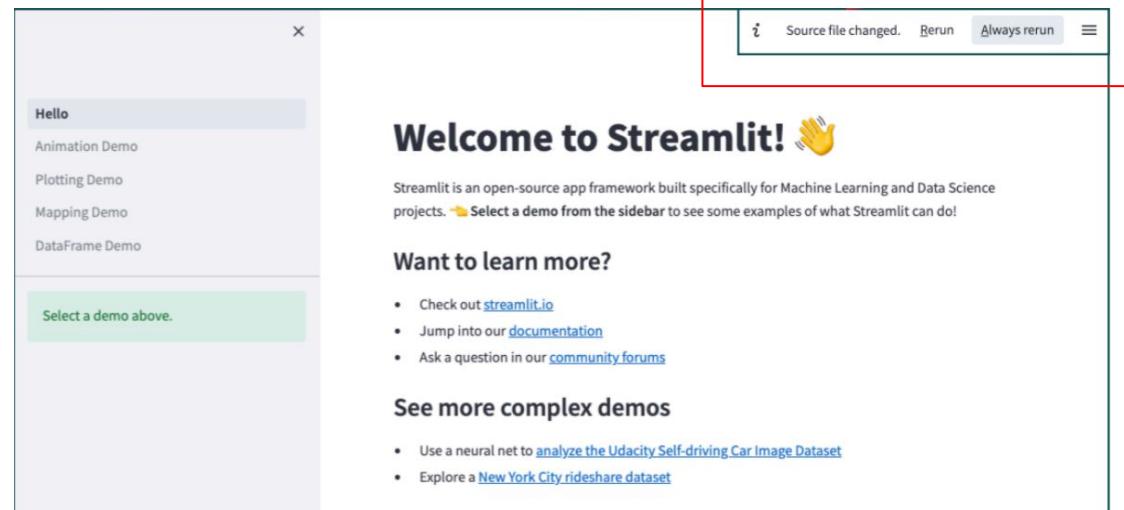


Flujo de ejecución

Streamlit **detecta al momento** los cambios de código fuente, permitiendo una **recarga rápida**.

⚠️⚠️ **Importante:** El script se ejecuta completo cada vez desde el principio. ⚠️⚠️

```
import streamlit as st  
  
#[...]
```



The screenshot shows the Streamlit app running. On the left, a sidebar lists demos: Hello, Animation Demo, Plotting Demo, Mapping Demo, and DataFrame Demo. A green button at the bottom of the sidebar says "Select a demo above.". The main area displays the "Hello" demo, which shows the text "Hello" and a "Select a demo above." message. At the top right of the main area, there is a status bar with a red border containing the text "Source file changed. Rerun Always rerun". Below the status bar, the text "Welcome to Streamlit! 🙌" is displayed, followed by a brief description of Streamlit's purpose and links to documentation and forums.

Welcome to Streamlit! 🙌

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects. 🚀 Select a demo from the sidebar to see some examples of what Streamlit can do!

Want to learn more?

- Check out [streamlit.io](#)
- Jump into our [documentation](#)
- Ask a question in our [community forums](#)

See more complex demos

- Use a neural net to [analyze the Udacity Self-driving Car Image Dataset](#)
- Explore a [New York City rideshare dataset](#)

Creación de aplicación Streamlit

Crea un archivo **app.py** y ejecuta streamlit con el comando **streamlit run app.py**

Crea el esqueleto básico de paginación con la siguiente estructura:

```
import streamlit as st
import requests
import pandas as pd
import numpy as np
```

```
PS G:\My Drive\Clases> streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.136:8501
```

Antes de comenzar: Estructura de ficheros

Según incremente la complejidad, podemos dividir el proyecto en varios ficheros y carpetas

```
streamlit_app  
└── app.py
```

```
streamlit_app  
├── app.py  
├── data1.csv  
├── random_data_file.csv  
└── output.csv
```

```
streamlit_app/  
├── data/  
│   └── data1.csv  
│   └── random_data_file.csv  
├── output/  
│   └── results.csv  
└── app.py  
    └── ejercicio1.py  
    └── ejercicio2.py
```

El punto de entrada será el mismo:

```
streamlit run app.py
```

Interacción con el usuario: Textos y formato

Campos de texto en Streamlit

```
import streamlit as st
```

En el cuerpo principal escribimos con comandos con función **st**.

- `st.title("")` → Título
- `st.header("")` → Cabecera
- `st.subheader("")` → Subcabecera
- `st.write("")` → Texto normal
- `st.markdown("")` → Texto en formato [markdown](#)
- `st.success("")` → Mensaje en color verde
- `st.info("")` → Mensaje de información
- `st.warning("")` → Mensaje de advertencia
- `st.error("")` → mensaje de error

Título

Cabecera

Subcabecera

Texto normal

Texto en formato [markdown](#)

Mensaje de éxito

Mensaje de información

Mensaje de advertencia

Mensaje de error

Las funciones de streamlit `st.(" "," "," "," ")` admiten mucha propiedades de personalización:
Ejemplo: `st.markdown("Texto en formato **markdown**", help="Con esta ayuda")`

Campos de texto en Streamlit

En la barra lateral escribimos con comandos `st.sidebar.`

- `st.sidebar.title("")` → Título
- `st.sidebar.header("")` → Cabecera
- `st.sidebar.subheader("")` → Subcabecera
- `st.sidebar.write("")` → Texto normal
- `st.sidebar.markdown("")` → Texto en formato [markdown](#)
- `st.sidebar.success("")` → Mensaje en color verde
- `st.sidebar.info("")` → Mensaje de información
- `st.sidebar.warning("")` → Mensaje de advertencia
- `st.sidebar.error("")` → mensaje de error

Título

Cabecera

Subcabecera

Texto normal

Mensaje de éxito

Mensaje de información

Mensaje de advertencia

Mensaje de error

Campos de texto en Streamlit

También podemos insertar contenido HTML con `st.html`

```
import streamlit as st
import requests
import pandas as pd
import numpy as np

contenido_html = """
<p>Esto es <i>un</i> párrafo</p>
<p>Esto es <b>otro</b> párrafo</p>
"""

st.html(contenido_html)
```

Esto es *un* párrafo

Esto es **otro** párrafo

Título y favicon en Streamlit

Podemos personalizar el título y el favicon de la página con

(Debe ser el primer código a ejecutar en Streamlit)

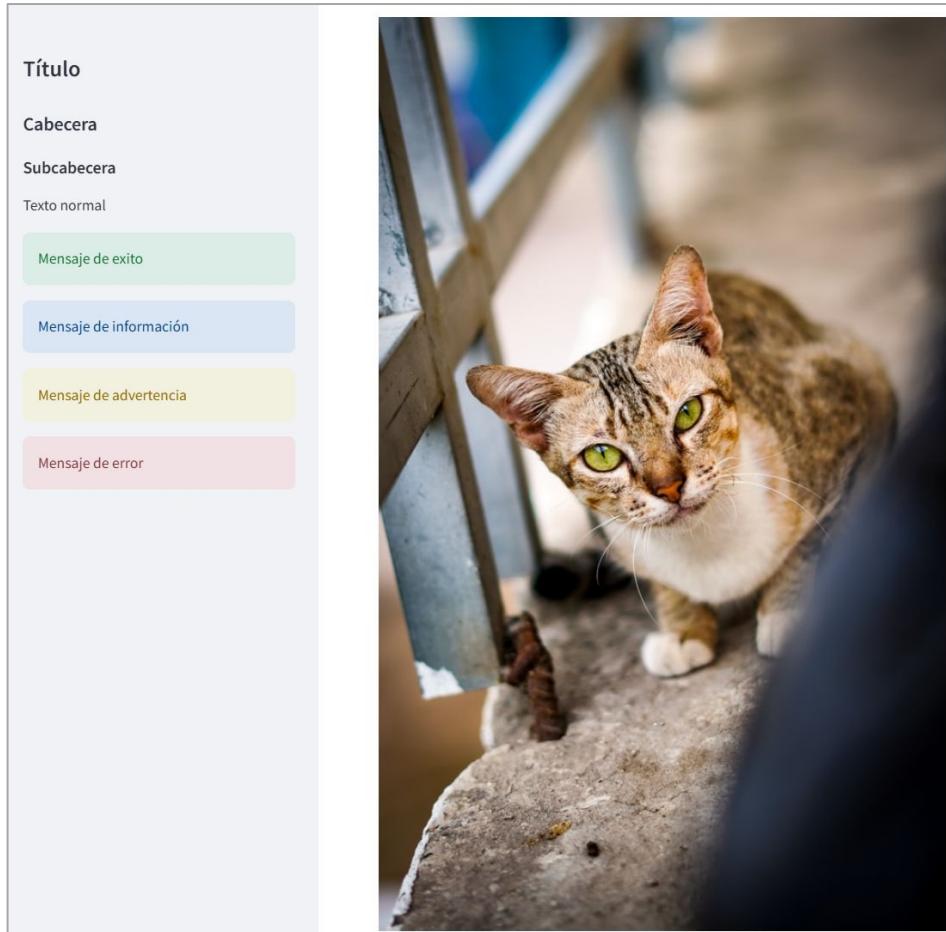
```
import streamlit as st
import requests
import pandas as pd
import numpy as np

st.set_page_config(
    page_title="Ejemplos MBIT School",
    page_icon="chart_with_upwards_trend",
    #layout="wide",
)
# Resto de nuestro programa...
```

Imágenes en Streamlit

con `st.image("")` podemos insertar una imagen.

<https://static.streamlit.io/examples/cat.jpg>



Distribución y formato en Streamlit: Columnas

Por defecto, Streamlit procesa en una columna. Si queremos múltiples columnas, las definimos como `[vars...] = st.columns(n)`:

```
import streamlit as st
col1, col2, col3 = st.columns(3)

with col1:
    # Contenido columna 1
with col2:
    # Contenido columna 2
with col3:
    # Contenido columna 3
```

```
import streamlit as st
col1, col2, col3 = st.columns(3)

with col1:
    st.header("Gato")
    st.image("https://static.streamlit.io/examples/cat.jpg")
with col2:
    st.header("Perro")
    st.image("https://static.streamlit.io/examples/dog.jpg")
with col3:
    st.header("Buho")
    st.image("https://static.streamlit.io/examples/owl.jpg")
```

Gato



Perro



Buho



Distribución y formato en Streamlit: Pestañas

Streamlit también permite estructurar el contenido en pestañas con una sintaxis similar a la anterior: [vars...] = st.tabs(["nombre"...]):

```
import streamlit as st
tab1, tab2, tab3 = st.tabs(["Gato", "Perro", "Buho"])

with tab1:
    # Contenido pestaña 1
with tab2:
    # Contenido pestaña 2
with tab3:
    # Contenido pestaña 3
```

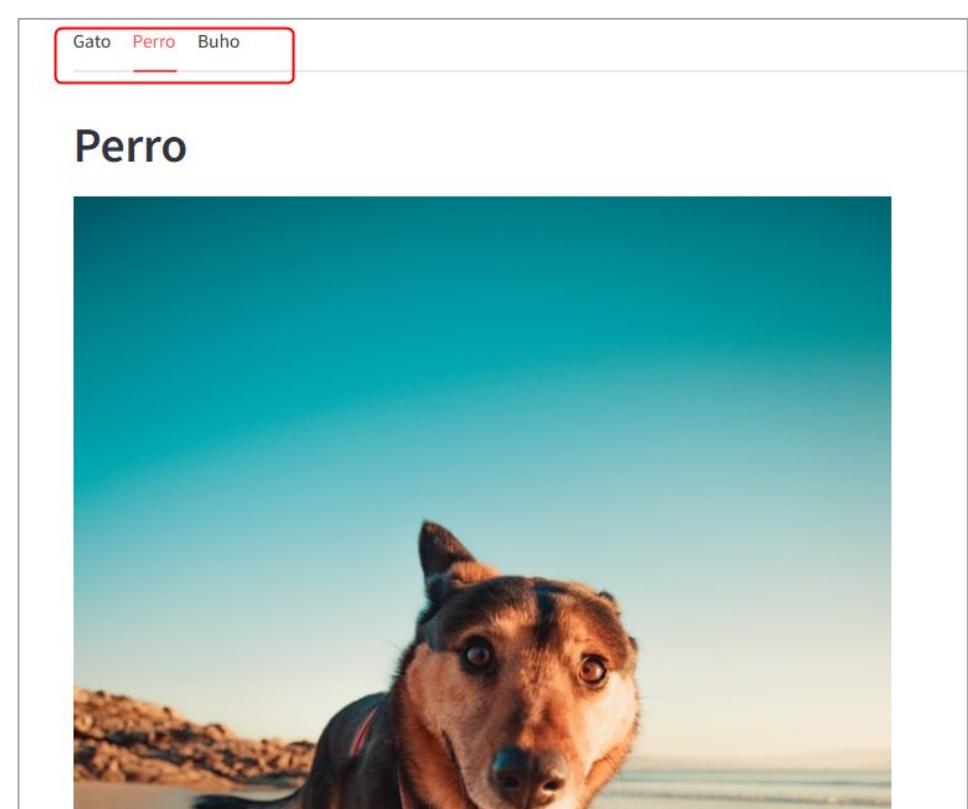
```
import streamlit as st

tab1, tab2, tab3 = st.tabs(["Gato", "Perro", "Buho"])

with tab1:
    st.header("Gato")
    st.image("https://static.streamlit.io/examples/cat.jpg")

with tab2:
    st.header("Perro")
    st.image("https://static.streamlit.io/examples/dog.jpg")

with tab3:
    st.header("Buho")
    st.image("https://static.streamlit.io/examples/owl.jpg")
```



NEXT →

Distribución y formato en Streamlit: Expander

Streamlit ofrece contenedores expandibles: Fragmentos de página que aparecen tras desplegar el fragmento comprimido.

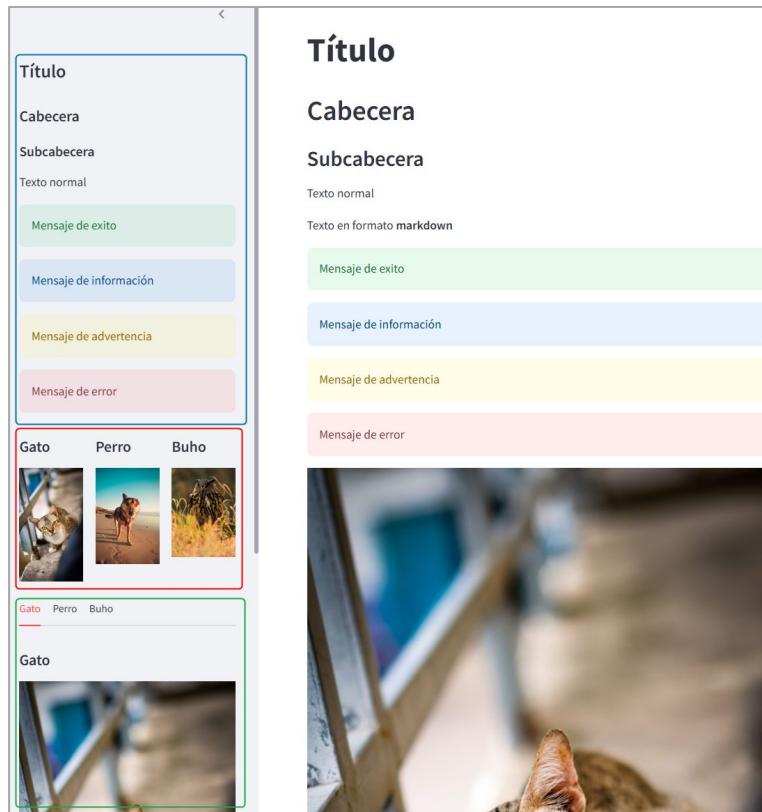
```
import streamlit as st

with st.expander("Jugar al dado"):
    st.markdown("""
        Tras lanzar este dado varias veces,
        estos son los resultados del mismo.
        Prueba tu a lanzar el dado y **superalo**
    """)
    st.image("https://static.streamlit.io/examples/dice.jpg")
    st.bar_chart({"data": [1,5,2,6,2,1]})
```



Distribución y formato en Streamlit: Barra lateral

Todos los elementos anteriores, incluyendo .sidebar. → Crean el objeto en la barra lateral



Interacción con el usuario: Widgets

Elementos de interacción: Dataframe

Que elementos de datos expone la API de streamlit para poder interaccionar con los usuarios?

`st.dataframe()`: El objeto más básico de representación de datos, como tabla.

```
import streamlit as st
import pandas as pd
import numpy as np

df = pd.DataFrame(
    np.random.randn(50, 20),
    columns=('col %d' % i for i in range(20)))

st.dataframe(df)
```

	col 0	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8	col 9	col 10
0	0.7992	0.1401	-0.7756	0.1362	0.0568	0.5485	1.1696	-0.2341	-2.5364	-0.8522	-0.6765
1	0.8236	-1.1647	1.6072	-0.6846	-0.0229	-0.5654	1.4889	-0.3936	0.9351	-2.2278	0.4633
2	-0.0186	1.2675	0.6417	0.6936	0.8117	-0.0511	-1.5626	-0.017	0.7792	-1.1881	0.4252
3	-0.7633	-1.465	-0.6748	0.4865	0.8344	0.8496	0.4156	0.2799	0.0236	0.8674	-2.3394
4	0.5401	-0.8338	-0.8033	0.0535	0.6198	1.3672	-2.3402	2.1889	0.0157	0.4969	-0.7749
5	-1.0838	-2.0621	0.3214	0.1325	-0.4565	-1.2014	0.0601	-1.7124	0.0404	-2.4282	-0.0848
6	0.6125	-1.2111	-0.5468	-0.5227	0.5222	-0.009	0.0234	1.0605	-0.4012	-0.3154	1.5247
7	0.0538	0.0139	-0.7236	1.3794	0.9118	0.1438	-1.4794	1.2302	-1.4863	-1.7181	-1.1972
8	-1.5407	-0.3917	0.318	-2.1377	-0.8311	0.1058	0.4956	0.1337	0.1332	0.1938	-0.5158
9	-0.2359	-0.7341	0.4298	1.5928	2.3513	0.206	-1.3896	-0.8483	0.1379	-0.4253	0.423

```
(data: Data = None, width: int | None = None,
height: int | None = None, *, use_container_width:
bool | None = None, hide_index: bool | None = None,
column_order: Iterable[str] | None = None,
column_config: ColumnConfigMappingInput | None =
None, key: Key | None = None, on_select:
Literal['ignore'] = "ignore", selection_mode:
SelectionMode | Iterable[SelectionMode] = "multi-
row", row_height: int | None = None) ->
DeltaGenerator
```

width : int or None
Desired width of the dataframe expressed in pixels. If width is

1/2

st.dataframe(df, width=1000, height=500)

NEXT →

Elementos de interacción: Data Editor

Que elementos de datos expone la API de streamlit para poder interaccionar con los usuarios?

`st.data_editor()`: Es un dataframe donde podemos manipular y editar datos.

```
import streamlit as st
import pandas as pd

df = pd.DataFrame(
    [
        {"pelicula": "Matrix 1", "rating": 5, "visto": True},
        {"pelicula": "Matrix 2", "rating": 3, "visto": True},
        {"pelicula": "Matrix 3", "rating": 4, "visto": True},
    ]
)

edited_df = st.data_editor(df)

pelicula_favorita = edited_df.loc[edited_df["rating"].idxmax()]["pelicula"]
st.markdown(f"Tu pelicula favorita es **{pelicula_favorita}** 🎉")
```

	≡, pelicula	≡, rating	≡, visto
0	Matrix 1	5	<input checked="" type="checkbox"/>
1	Matrix 2	3	<input checked="" type="checkbox"/>
2	Matrix 3	4	<input type="checkbox"/>

Elementos de interacción: Selector individual

`st.radio()`: Selector individual tipo radio

`st.selectbox()`: selector individual desplegable

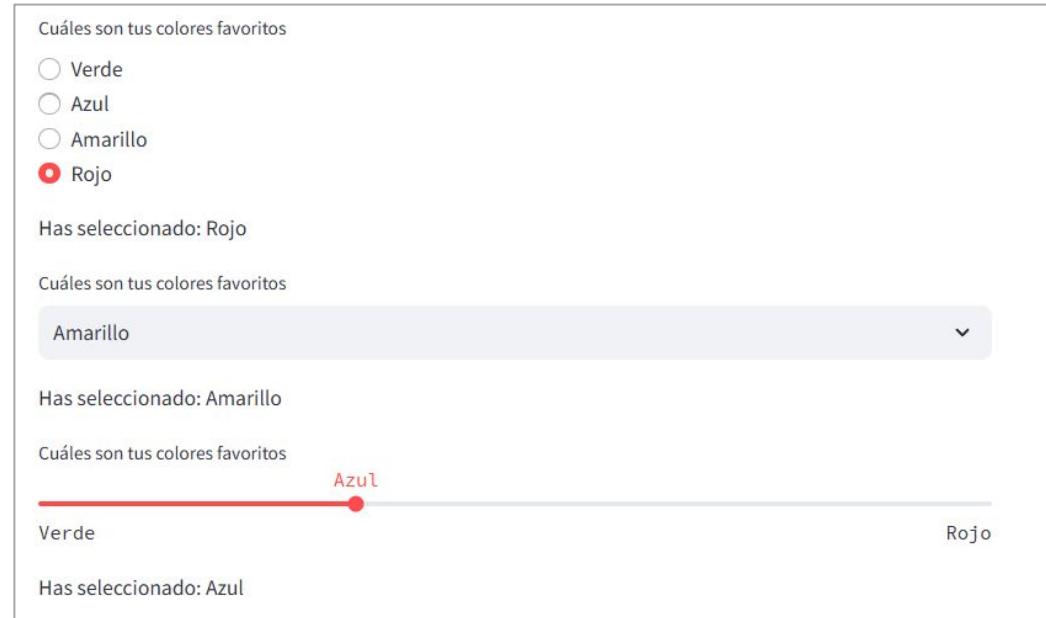
`st.select_slider()`: selector individual slider

```
import streamlit as st

#Variante con botones de radio
options = st.radio(
    'Cuáles son tus colores favoritos',
    ['Verde', 'Azul', 'Amarillo', 'Rojo']
)
st.write('Has seleccionado:', options)

#Variante con selectbox
options2 = st.selectbox(
    'Cuáles son tus colores favoritos',
    ['Verde', 'Azul', 'Amarillo', 'Rojo']
)
st.write('Has seleccionado:', options2)

#Variante con slider
options3 = st.select_slider(
    'Cuáles son tus colores favoritos',
    ['Verde', 'Azul', 'Amarillo', 'Rojo']
)
st.write('Has seleccionado:', options3)
```



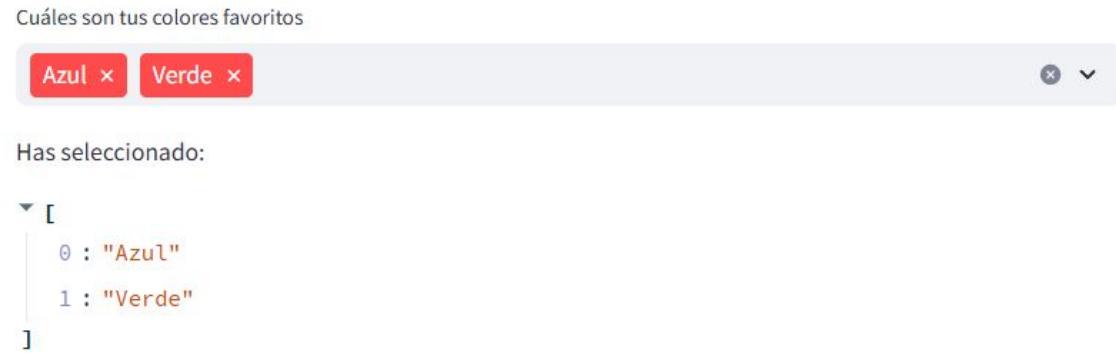
The screenshot shows a Streamlit application interface with three sections:

- Top Section:** A radio button group titled "Cuáles son tus colores favoritos". The options are Verde, Azul, Amarillo, and Rojo. The "Rojo" option is selected, indicated by a red dot.
- Middle Section:** A selectbox titled "Cuáles son tus colores favoritos". The dropdown menu shows "Amarillo" as the selected value.
- Bottom Section:** A slider titled "Cuáles son tus colores favoritos". The slider has tick marks for "Verde", "Azul", and "Rojo". The slider handle is positioned between Verde and Azul, with the word "Azul" displayed above it.

Elementos de interacción: Selector múltiple

`st.multiselect()`: Elemento multiselector

```
options = st.multiselect(  
    'Cuáles son tus colores favoritos',  
    ['Verde', 'Azul', 'Amarillo', 'Rojo'],  
    ['Azul','Verde'])  
st.write('Has seleccionado:', options)
```

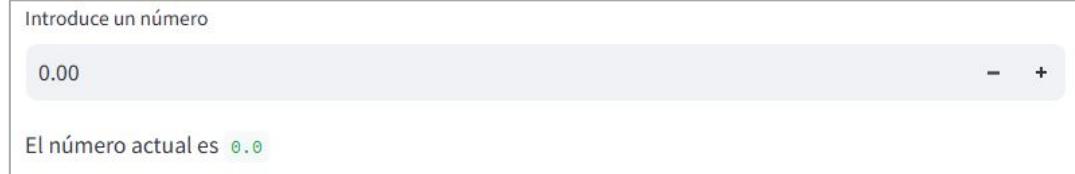


Elementos de interacción: Entradas de texto

`st.number_input()`: Selector numérico

```
import streamlit as st

number = st.number_input('Introduce un número')
st.write('El número actual es', number)
print(f"El número es {number}")
```



```
import streamlit as st

number = st.number_input('Introduce un número',
min_value=0, max_value=10, value=5)
st.write('El número actual es', number)
print(f"El número es {number}")
```



Elementos de interacción: Entradas de texto

`st.text_input()`: Texto única línea

`st.text_area()`: Texto multilínea

```
import streamlit as st

title = st.text_input("Título de la película",
"Matrix")
st.write("El título de la película es", title)

title_todas = st.text_area("Cuenta todas las
películas", "Matrix")
st.write("El título de la película es", title_todas)
```

Título de la película

Matrix

El título de la película es Matrix

Cuenta todas las películas

Matrix

El título de la película es Matrix

Elementos de interacción: Subida de ficheros

`st.file_uploader()`: Procesamiento de ficheros de forma individual o múltiple

```
import streamlit as st

ficheros_a_subir = st.file_uploader("Elije uno o varios ficheros CSV", accept_multiple_files=True)

for fichero in ficheros_a_subir:
    st.write(fichero.name)
    st.dataframe(pd.read_csv(fichero))
```



ejemplo.csv				
	A	B	C	
0	1	2	3	

ejemplo2.csv				
	A	B	C	
0	4	5	6	

Elementos de interacción: Selector de fechas

`st.date_input()`: Selector de fechas

```
import streamlit as st
import datetime

dia_bitcoin = st.date_input(
    "¿En qué fecha apareció la primera criptomonedra, el Bitcoin?",
    datetime.date(2000,1,1))

#Fecha aparición Bitcoin: 31 Octubre 2008
```

¿En qué fecha apareció la primera criptomonedra, el Bitcoin?

2000/01/01

Fue más tarde ➔

```
import streamlit as st
import datetime

dia_bitcoin = st.date_input(
    "¿En qué fecha apareció la primera criptomonedra, el Bitcoin?",
    datetime.date(2000,1,1),
    min_value=datetime.date(2000,1,1),
    max_value=datetime.date(2010,12,31)
)
```

Elementos de interacción: Contenido audiovisual

`st.image()`: Mostrar una imagen

`st.audio()`: Mostrar un reproductor de audio

`st.video()`: Mostrar una ventana con vídeo

```
import streamlit as st
st.image("https://i.imgur.com/vppXpob.png", use_column_width=True, caption="MBIT Data School")

### Audios
import streamlit as st
import requests
st.audio("https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/25.ogg")
st.audio(requests.get("https://raw.githubusercontent.com/PokeAPI/cries/main/cries/pokemon/latest/25.ogg").content)

### Video example
import streamlit as st
import requests
st.video("https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4")
```



Elementos de interacción: st.spinner

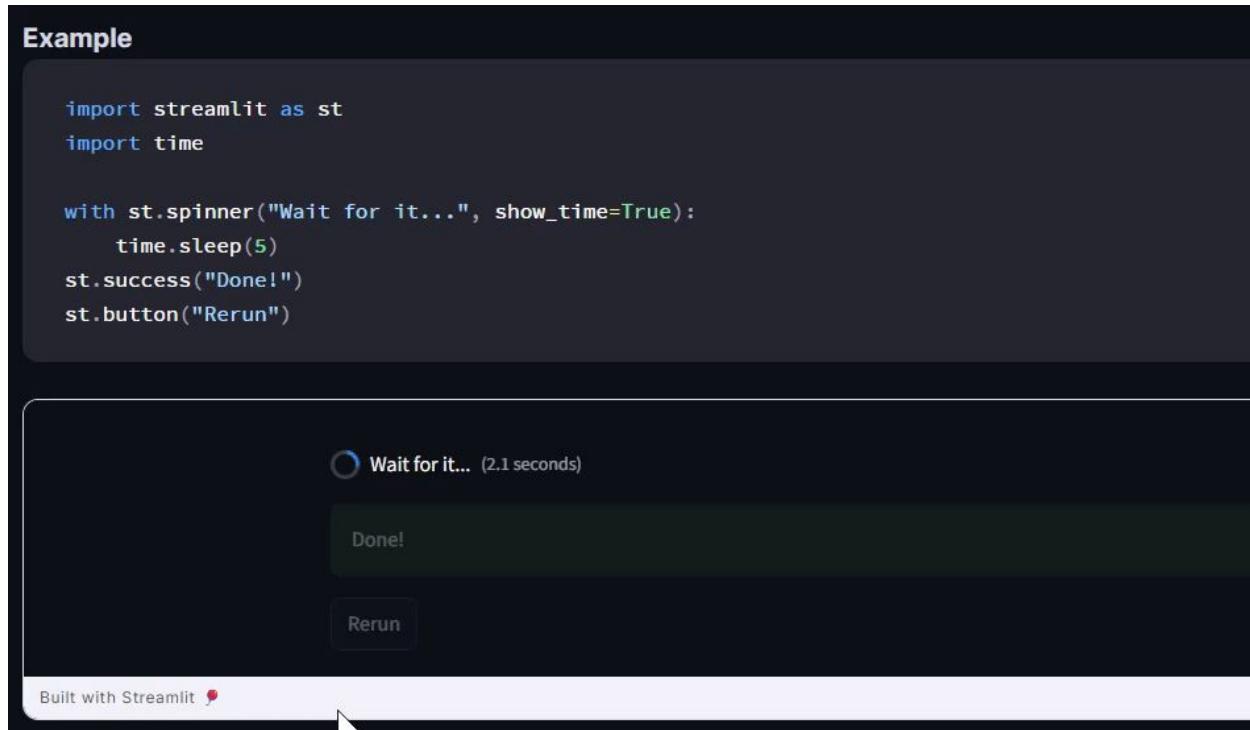
`st.spinner()`: Muestra un elemento visual rotando mientras ejecuta el código fuente

```
import time
with st.spinner("Cargando datos...", show_time=True):
    time.sleep(2)
    st.success("Done!")
    st.button("Rerun")
```

Example

```
import streamlit as st
import time

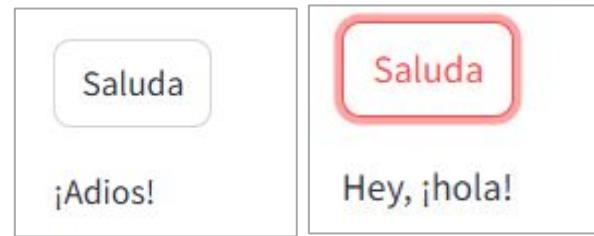
with st.spinner("Wait for it...", show_time=True):
    time.sleep(5)
    st.success("Done!")
    st.button("Rerun")
```



Elementos de interacción: Button

`st.button()`: Objeto botón

```
if st.button("Saluda"):
    st.write("Hey, ¡hola!")
else:
    st.write("¡Adios!")
```



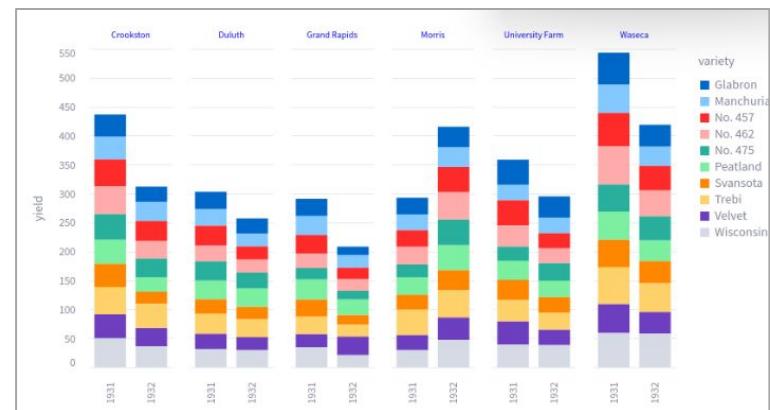
Elementos de visualización: Gráficos

Elementos de visualización - Gráficos

Uno de los puntos fuertes de Streamlit es la capacidad de representar información visual.

Streamlit incorpora gráficos de forma nativa, pero también soporta múltiples bibliotecas populares de visualización:

- altair
- plotly
- bokeh
- matplotlib
- seaborn



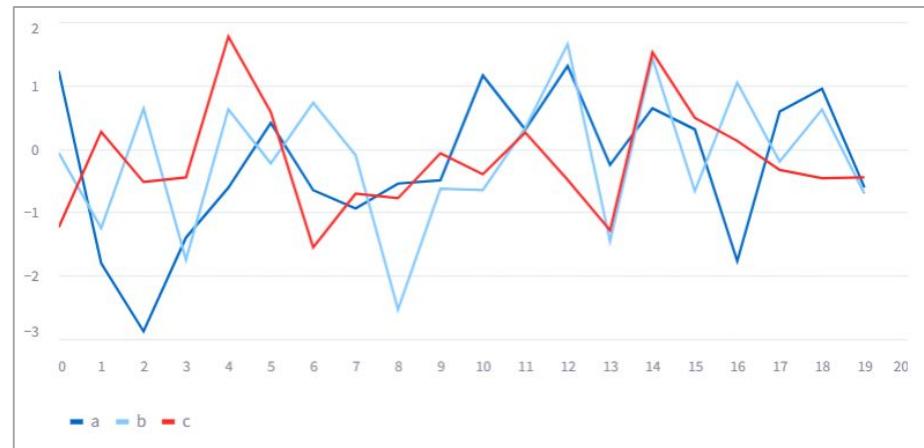
Gráficos nativos: Gráfico de líneas

`st.line_chart()`: Diagrama de líneas

```
import streamlit as st
import pandas as pd
import numpy as np

datos_grafica = pd.DataFrame(
    np.random.randn(20, 3),
    columns=['a', 'b', 'c'])

st.line_chart(datos_grafica)
```



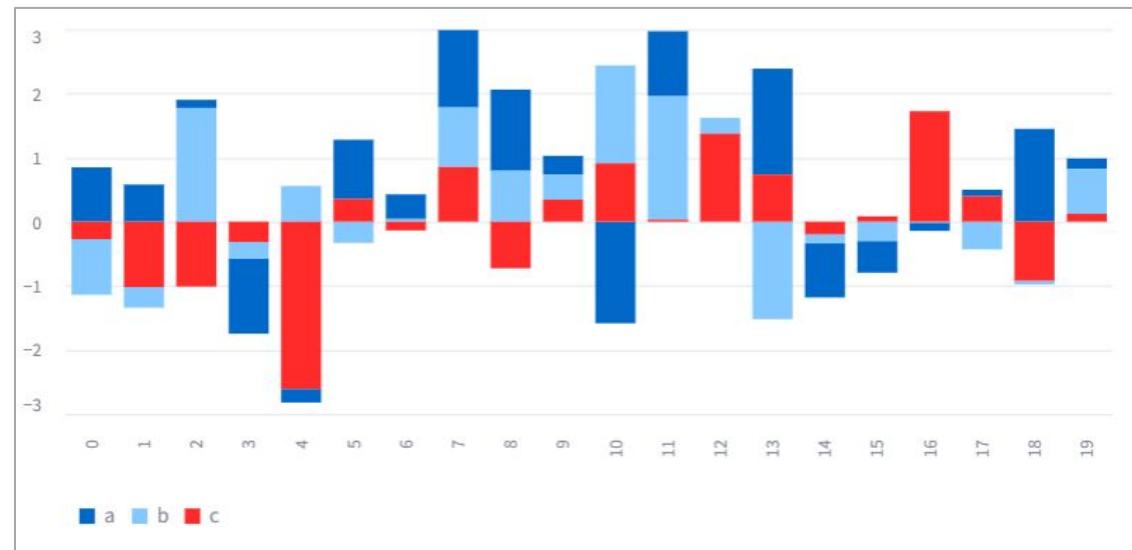
Gráficos nativos: Diagrama de barras

`st.bar_chart()`: Diagrama de barras

```
import streamlit as st
import pandas as pd
import numpy as np

datos_grafica = pd.DataFrame(
    np.random.randn(20, 3),
    columns=['a','b','c'])

st.bar_chart(datos_grafica)
```



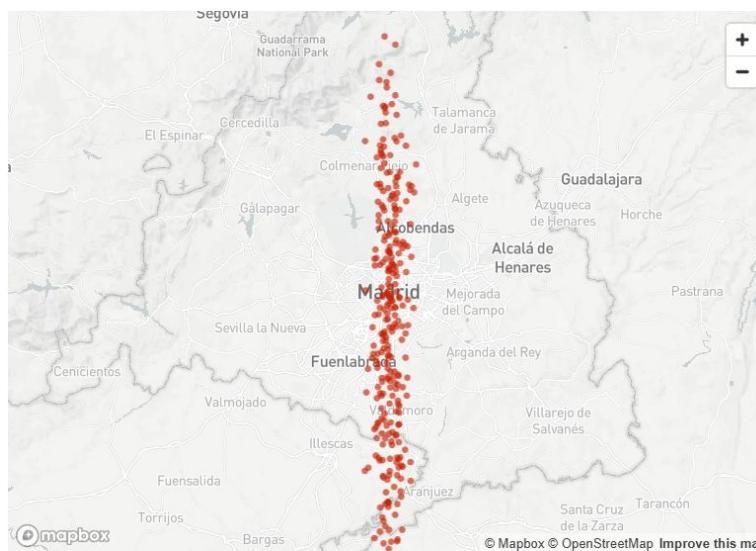
Gráficos nativos: Mapas

`st.map()`: Visualización sobre mapas con Mapbox

```
## Mapas
import streamlit as st
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(300 , 2) / [-3.8, 42] + [40.4, -3.7],columns=['lat','lon'])

st.map(df)
```



Gráficos de terceros: Altair

`st.altair_chart()`: Diagrama de burbujas

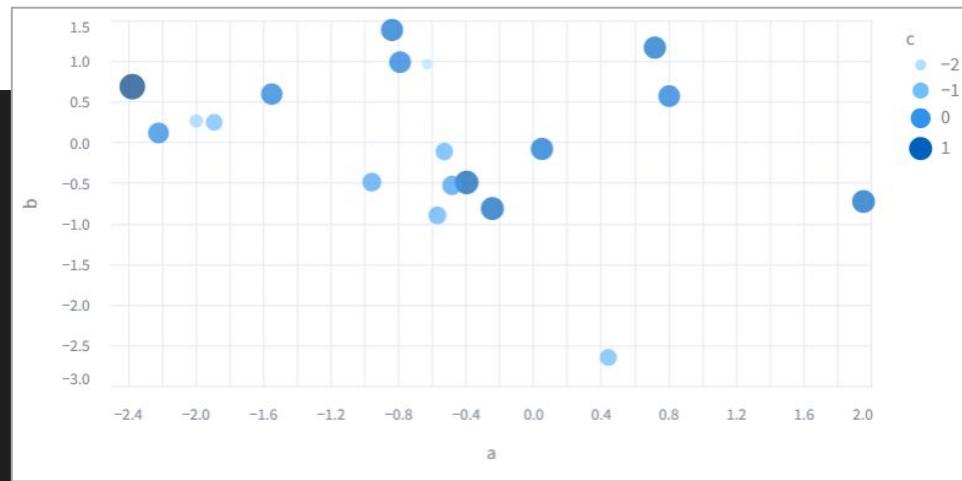
https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html

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

datos_grafica = pd.DataFrame(
    np.random.randn(20, 3),
    columns=['a', 'b', 'c'])

c = alt.Chart(datos_grafica).mark_circle().encode(
    x='a', y='b', size='c', color='c', tooltip=['a', 'b', 'c'])

st.altair_chart(c, use_container_width=True)
```



Gráficos de terceros: Plotly

`st.plotly_chart()`: Diagrama de distribución

https://altair-viz.github.io/user_guide/generated/toplevel/altair.Chart.html

```
# Visualización de terceros: Plotly
import streamlit as st
import numpy as np
import plotly.figure_factory as ff

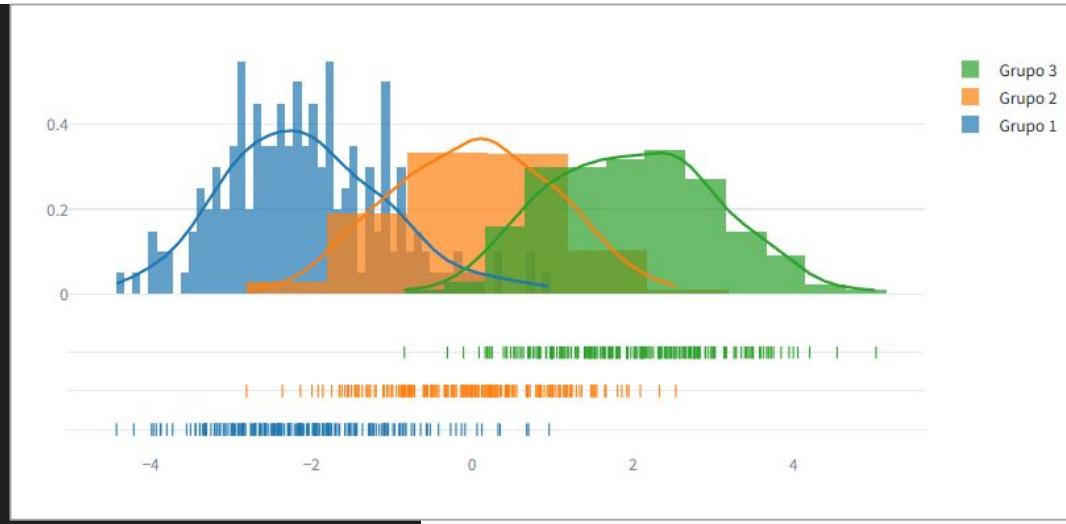
#Creamos histograma
x1 = np.random.randn(200) - 2
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 2

#Agrupa datos
hist_data = [x1, x2, x3]

group_labels = ["Grupo 1", "Grupo 2", "Grupo 3"]

#Crea distplot con tamaño personalizado
fig = ff.create_distplot(hist_data, group_labels, bin_size=[.1, -24, .5])

#Muestra en Streamlit
st.plotly_chart(fig, use_container_width=True)
```



Gráficos de terceros: Matplotlib

`st.pyplot()`: Diagrama de tarta

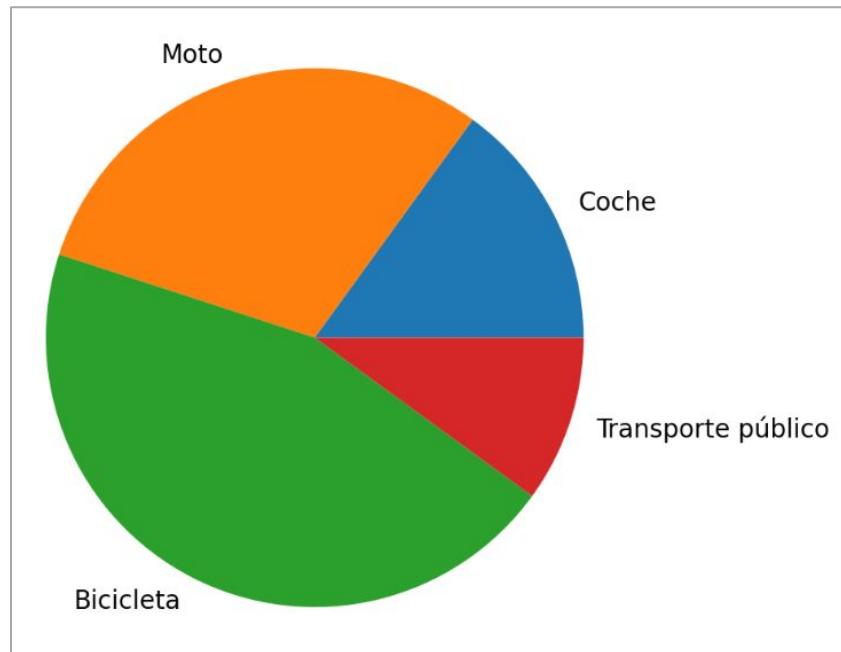
https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html

```
import matplotlib.pyplot as plt

labels = 'Coche', 'Moto', 'Bicicleta',
'Transporte público'
sizes = [15, 30, 45, 10]

fig, ax = plt.subplots()
tarta = ax.pie(sizes, labels=labels)

st.pyplot(fig)
```



Gráficos de terceros: Matplotlib

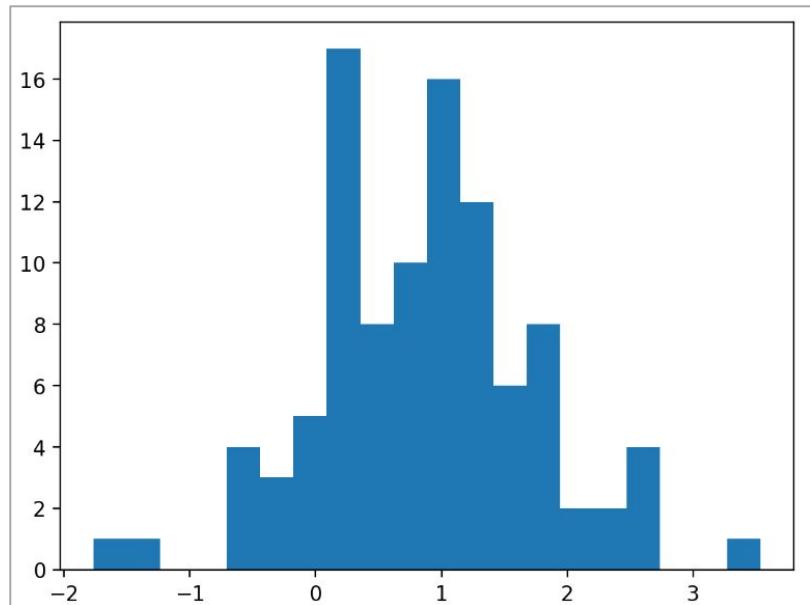
`st.pyplot()`: Histograma

https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html

```
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

arr = np.random.normal(1, 1, size=100)
fig, ax = plt.subplots()
ax.hist(arr, bins=20)

st.pyplot(fig)
```



Cheat Sheet y guía completa

En la documentación de Streamlit hay dos enlaces que agrupan la información resumida (Cheat sheet) y completa de todos elementos disponibles.

- Cheat Sheet: <https://docs.streamlit.io/develop/quick-reference/cheat-sheet>
- Guía completa: <https://docs.streamlit.io/develop/api-reference>

```

st.text("Fixed width text")
st.markdown("_Markdown_")
st.latex(r"""\ e^{\pi} + 1 = 0 """)
st.title("My title")
st.header("My header")
st.subheader("My sub")
st.code("for i in range(8): foo()")
st.html("<p>Hi!</p>")

Display data

st.dataframe(my_dataframe)
st.table(data.iloc[0:10])
st.json({"foo": "bar", "fu": "ba"})
st.metric("My metric", 42, 2)

Display media

st.image("./header.png")
st.audio(data)
st.video(data)
st.video(data, subtitles="./subs.vtt")
st.logo("logo.jpg")

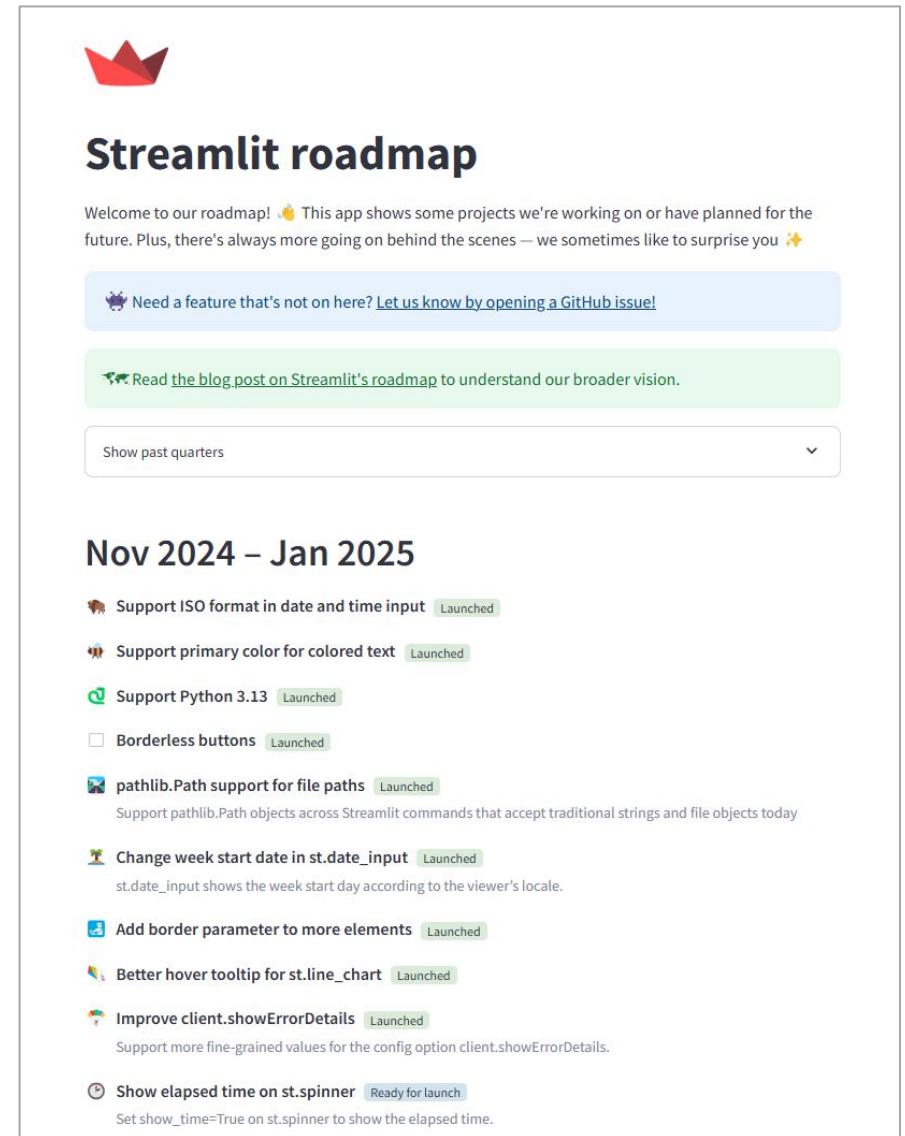
Build chat-based apps

# Insert a chat message container.
with st.chat_message("user"):
    st.write("Hello 👋")
    st.line_chart(np.random.randn(30, 3))

```

Roadmap Streamlit

<https://roadmap.streamlit.app/>



The screenshot shows the Streamlit roadmap interface. At the top is a red crown icon. Below it, the title "Streamlit roadmap" is displayed. A welcome message reads: "Welcome to our roadmap! 🎉 This app shows some projects we're working on or have planned for the future. Plus, there's always more going on behind the scenes — we sometimes like to surprise you 🎉". There are two callout boxes: one with a purple icon and the text "Need a feature that's not on here? [Let us know by opening a GitHub issue!](#)" in blue; another with a green icon and the text "Read [the blog post on Streamlit's roadmap](#) to understand our broader vision." in blue. A dropdown menu is open, showing the option "Show past quarters" with a downward arrow. The main section is titled "Nov 2024 – Jan 2025" and lists several completed features with "Launched" status: "Support ISO format in date and time input", "Support primary color for colored text", "Support Python 3.13", "Borderless buttons", "pathlib.Path support for file paths" (with a note about supporting pathlib.Path objects across Streamlit commands), "Change week start date in st.date_input" (with a note about st.date_input showing the week start day according to the viewer's locale), "Add border parameter to more elements", "Better hover tooltip for st.line_chart", "Improve client.showErrorDetails" (with a note about supporting more fine-grained values for the config option client.showErrorDetails), and "Show elapsed time on st.spinner" (with a note about setting show_time=True on st.spinner to show the elapsed time). Each item has a small icon to its left.

Dos consideraciones importantes en Streamlit

Dos consideraciones importantes en Streamlit

Recuerda, Con cada iteración, Streamlit ejecuta todo el script de arriba a abajo y no almacena estado por defecto (solo información de selectores)

Para ello, la herramienta ofrece dos soluciones:

- Almacenamiento en variables globales de sesión
- Cacheo de información



Almacenamiento de información: Variables de sesión

Streamlit crea un diccionario `st.session_state{}` por cada interacción en todo el sistema a nivel de usuario.

El acceso a la información es `st.session_state["clave"] = 'valor'`

Initialize values in Session State

The Session State API follows a field-based API, which is very similar to Python dictionaries:

```
# Initialization
if 'key' not in st.session_state:
    st.session_state['key'] = 'value'

# Session State also supports attribute based syntax
if 'key' not in st.session_state:
    st.session_state.key = 'value'
```

Reads and updates

Read the value of an item in Session State and display it by passing to `st.write` :

```
# Read
st.write(st.session_state.key)

# Outputs: value
```

Update an item in Session State by assigning it a value:

```
st.session_state.key = 'value2'      # Attribute API
st.session_state['key'] = 'value2'   # Dictionary like API
```

Almacenamiento de información: Variables de sesión

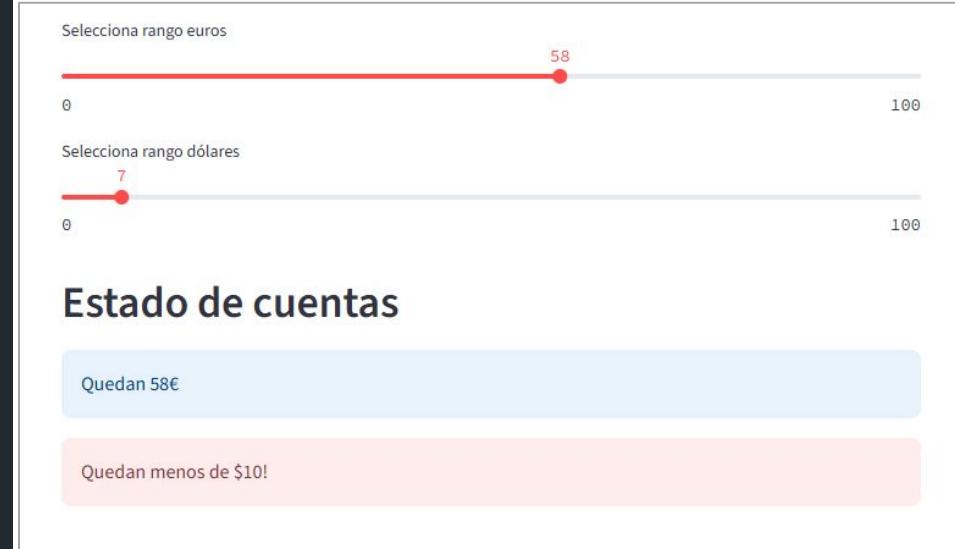
Todos los elementos de Streamlit tienen en común la propiedad **key**. Define de forma única a un objeto de Streamlit.

```
st.slider("Selecciona rango euros", 0, 100, 50, key="slider_euros")
st.slider("Selecciona rango dólares", 0, 100, 50, key="slider_dolares")

st.header("Estado de cuentas")

if st.session_state["slider_euros"] <= 10:
    st.error("Quedan menos de 10€!")
else:
    st.info(f"Quedan {st.session_state['slider_euros']}€")

if st.session_state["slider_dolares"] <= 10:
    st.error("Quedan menos de $10!")
else:
    st.info(f"Quedan ${st.session_state['slider_dolares']}")
```



Almacenamiento de información: Variables de sesión

Para crear directamente el valor y mantener persistencia, incluye el atributo key="valor". Accederás a ella a través de st.session_state.valor // st.session.state["valor"]

Every widget with a key is automatically added to Session State:

```
st.text_input("Your name", key="name")  
  
# This exists now:  
st.session_state.name
```

Para eliminar valores en st.session_state, utiliza del st.session_state.valor

Delete items

Delete items in Session State using the syntax to delete items in any Python dictionary:

```
# Delete a single key-value pair  
del st.session_state[key]  
  
# Delete all the items in Session state  
for key in st.session_state.keys():  
    del st.session_state[key]
```

Almacenamiento de información: Variables de sesión

Debug de todas las variables de session state

Curious about what is in Session State? Use `st.write` or magic:

```
st.write(st.session_state)
```

With magic:

st.session state

```
  "df_data" :  
    "  id      nombre                      imagen visto  
0   1  bulbasaur  https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/001.png  True  
1   2  ivysaur   https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/002.png False  
2   3  venusaur  https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/003.png False  
3   4  charmander https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/004.png False  
4   5  charmeleon https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/005.png False  
5   6  charizard  https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/006.png False  
6   7  squirtle   https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/007.png False  
7   8  wartortle  https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/008.png False  
8   9  blastoise  https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/009.png False  
9  10  caterpie  https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/010.png False
```

Almacenamiento de información: Cacheo

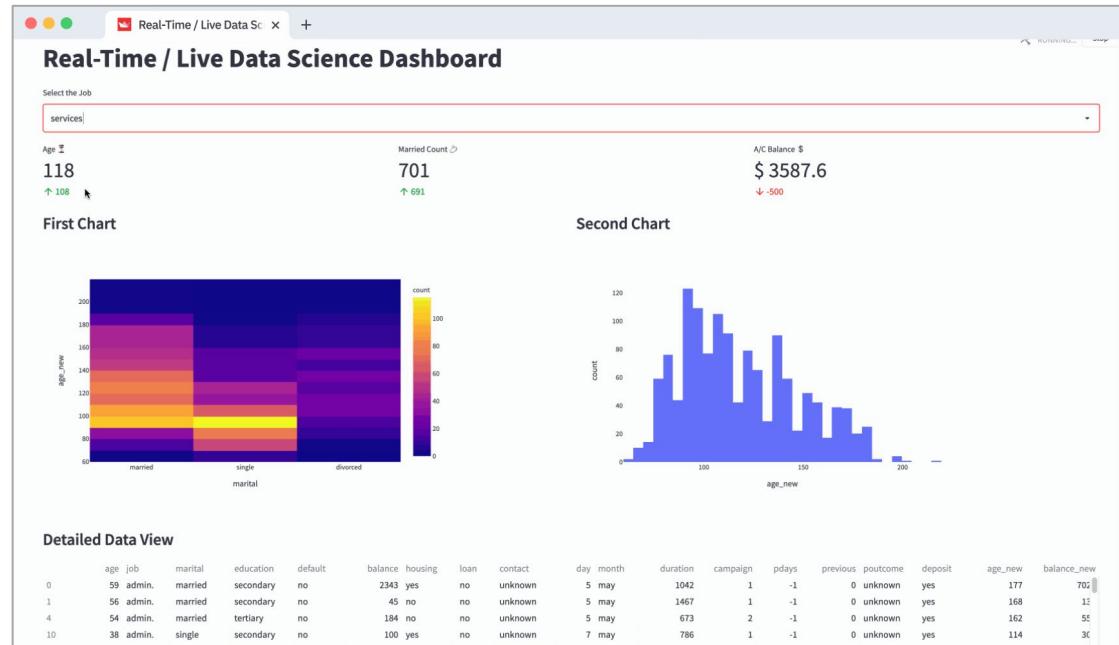
Reitero: Streamlit ejecuta **todo** el script en cada interacción de usuario.
 (A las API no les gusta eso, o sí, si son de pago por uso; entonces no te gustará a ti)

Ventajas:

- Muy útil para desarrollo rápido

Inconvenientes:

- En conjuntos de datos grandes la ejecución se repite una y otra vez.
- Los objetos se consultan infinidad de veces



Almacenamiento de información: Cacheo

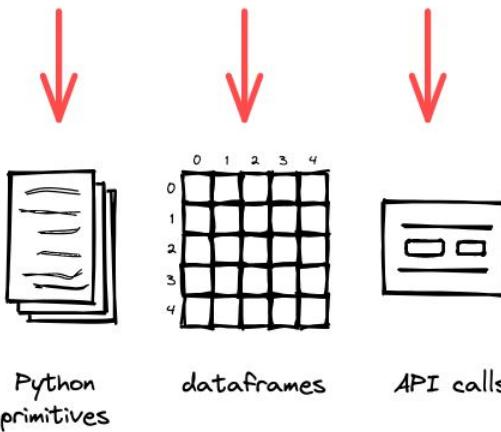
Streamlit puede cachear información obtenida por **funciones**

Se aplica un decorador @... sobre la función en concreto, y hay dos tipos:

- `@st.cache_data` (*Objetos de sesión de usuario y serializables*)
- `@st_cache_resource` (*Objetos globales, no serializables (uso concreto)*)

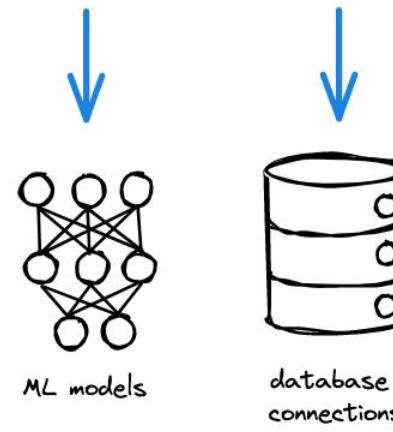
st.cache_data

anything you CAN store in a database



st.cache_resource

anything you CAN'T store in a database



st.cache_data()

Ejemplo con dataset de Uber con tamaño 50 MB

```
def load_data(url):
    df = pd.read_csv(url)
    return df

df = load_data("https://github.com/plotly/datasets/raw/master/uber-rides-data1.csv")
st.dataframe(df)

st.button("Rerun")
```

```
@st.cache_data
def load_data(url):
    df = pd.read_csv(url)
    return df

df = load_data("https://github.com/plotly/datasets/raw/master/uber-rides-data1.csv")
st.dataframe(df)

st.button("Rerun")
```

st.cache_data()

Ejemplo con APIs:

```
@st.cache_data
def obten_post(num_post):
    response = requests.get(f"https://jsonplaceholder.typicode.com/posts/{num_post}")
    return response.json()

st.dataframe(obten_post(5))
```

	value
userId	1
id	5
title	nesciunt quas odio
body	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse volup

st.cache_data()

Database queries

You usually make SQL queries to load data into your app when working with databases. Repeatedly running these queries can be slow, cost money, and degrade the performance of your database. We strongly recommend caching any database queries in your app. See also [our guides on connecting Streamlit to different databases](#) for in-depth examples.

```
connection = database.connect()

@st.cache_data
def query():
    return pd.read_sql_query("SELECT * from table", connection)
```



Tip

You should set a `ttl` (time to live) to get new results from your database. If you set

`st.cache_data(ttl=3600)`, Streamlit invalidates any cached values after 1 hour (3600 seconds) and runs the cached function again. See details in [Controlling cache size and duration](#).

st.cache_data()

Ejemplo:

(¡Violación de caché! - Propósito educativo: No introducir cambios de variables st.session_state en funciones cacheadas!)

```
import streamlit as st

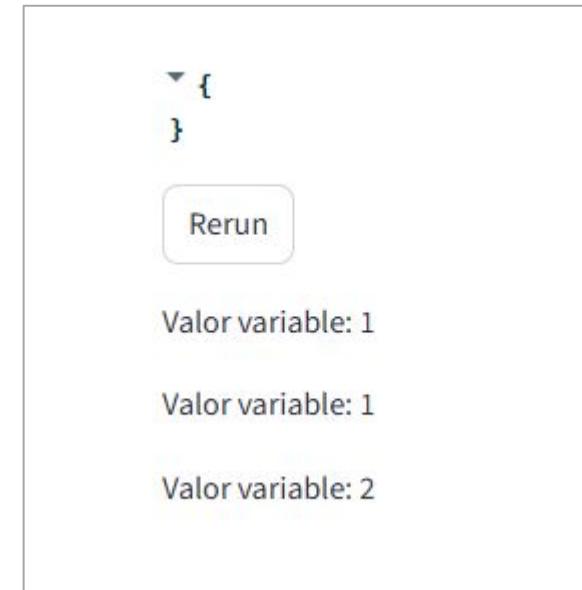
st.session_state["contador"] = 0

@st.cache_data
def obten_post(num_post):
    response = requests.get(f"https://jsonplaceholder.typicode.com/posts/{num_post}")
    st.session_state["contador"] = st.session_state["contador"] + 1
    return response.json()

d1 = obten_post(5)
#Primera ejecución, ejecuta función
st.write(f'Valor variable: {st.session_state["contador"]}')

d2 = obten_post(5)
#Segunda ejecución, al ser mismo valor de entrada, toma la caché
st.write(f'Valor variable: {st.session_state["contador"]}')

d3 = obten_post(3)
#Tercera función, el valor de entrada es diferente, ejecuta la función
st.write(f'Valor variable: {st.session_state["contador"]}'')
```



st.cache_data()

Para datos sensibles (passwords, tokens, bases de datos), añadimos el prefijo `_` al parámetro de entrada sensible.

By default, all parameters to a cached function must be hashable. Any parameter whose name begins with `_` will not be hashed. You can use this as an "escape hatch" for parameters that are not hashable:

```
import streamlit as st

@st.cache_data
def fetch_and_clean_data(_db_connection, num_rows):
    # Fetch data from _db_connection here, and then clean it up.
    return data

connection = make_database_connection()
d1 = fetch_and_clean_data(connection, num_rows=10)
# Actually executes the function, since this is the first time it was
# encountered.

another_connection = make_database_connection()
d2 = fetch_and_clean_data(another_connection, num_rows=10)
# Does not execute the function. Instead, returns its previously computed
# value - even though the _database_connection parameter was different
# in both calls.
```

Para eliminar un dato cacheado y forzar la descarga, eliminamos caché con `.clear()`

```
fetch_and_clean_data.clear()
# Clear all cached entries for this function.
```

st.cache_resource()

st.cache_resource() sirve para cachear un recurso que debe estar disponible a todos los usuarios, sesiones y repeticiones.

```
from transformers import pipeline

@st.cache_resource # 👈 Add the caching decorator
def load_model():
    return pipeline("sentiment-analysis")

model = load_model()

query = st.text_input("Your query", value="I love Streamlit! 🎉")
if query:
    result = model(query)[0] # 👈 Classify the query text
    st.write(result)
```

N usuarios * N ejecuciones * N interacciones...

st.cache_resource()

Ejemplo para base de datos:

```
@st.cache_resource
def init_connection():
    host = "hh-pgsql-public.ebi.ac.uk"
    database = "pfmegrnargs"
    user = "reader"
    password = "NWDMCE5xdipIjRrp"
    return psycopg2.connect(host=host, database=database, user=user,
password=password)

conn = init_connection()
```

Almacenamiento de información: Otros tips

Puedes establecer expiración a la caché tras un TTL determinado: `@st.cache_data(ttl=3600)`

Durante el desarrollo de tu aplicación, limita la carga a una pequeña muestra representativa de datos:

- Los cambios los verás más rápido
- Consumirás menos recursos
 - Equipo propio / Computación en la nube
 - Recursos de pago por uso (nº de llamadas)

```
import pandas as pd
n_cols=20
data =
pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv",
nrows=20)
st.dataframe(data)
```

	sepal_length	sepal_width	petal_length	petal_width	species
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3	1.4	0.1	setosa
13	4.3	3	1.1	0.1	setosa
14	5.8	4	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa

Forzar refresco de datos: st.rerun()

Relanza la ejecución del script de nuevo. Recomendable para refrescar datos ante un evento en concreto:

st.rerun Version 1.46.0 ▾

Rerun the script immediately.

When `st.rerun()` is called, Streamlit halts the current script run and executes no further statements. Streamlit immediately queues the script to rerun.

When using `st.rerun` in a fragment, you can scope the rerun to the fragment. However, if a fragment is running as part of a full-app rerun, a fragment-scoped rerun is not allowed.

Function signature	[source]
<code>st.rerun(*, scope="app")</code>	

Parameters

scope ("app" or "fragment") keyword-only	Specifies what part of the app should rerun. If <code>scope</code> is <code>"app"</code> (default), the full app reruns. If <code>scope</code> is <code>"fragment"</code> , Streamlit only reruns the fragment from which this command is called. Setting <code>scope="fragment"</code> is only valid inside a fragment during a fragment rerun. If <code>st.rerun(scope="fragment")</code> is called during a full-app rerun or outside of a fragment, Streamlit will raise a <code>StreamlitAPIException</code> .

Ejercicios

Ejercicio 00: Preparativos

Crea un archivo **ejercicios.py** y ejecuta streamlit con el comando **streamlit run ejercicios.py**

Crea el esqueleto básico de paginación con la siguiente estructura:

```
import streamlit as st
import requests
import pandas as pd

ejercicio = st.sidebar.radio('Ejercicio', ["1", "2", "3", "4", "5", "6", "7"])

if ejercicio == "1":
    st.header("Ejercicio 1")

if ejercicio == "2":
    st.header("Ejercicio 2")

if ejercicio == "3":
    st.header("Ejercicio 3")

if ejercicio == "4":
    st.header("Ejercicio 4")

if ejercicio == "5":
    st.header("Ejercicio 5")

if ejercicio == "6":
    st.header("Ejercicio 6")

if ejercicio == "7":
    st.header("Ejercicio 7")
```

Ejercicio 01: Geoposicionamiento IP

Cómo convertir un script de Python en una aplicación Streamlit:

```
import requests

#Obten tu propia dirección IP
direccion_ip = requests.get("http://ifconfig.me")
print(f"Dirección IP: {direccion_ip.text}")

#Obten detalles de tu dirección IP detectada en formato JSON
loc = requests.get(f"http://ip-api.com/json/{direccion_ip.text}")
print(loc.json())

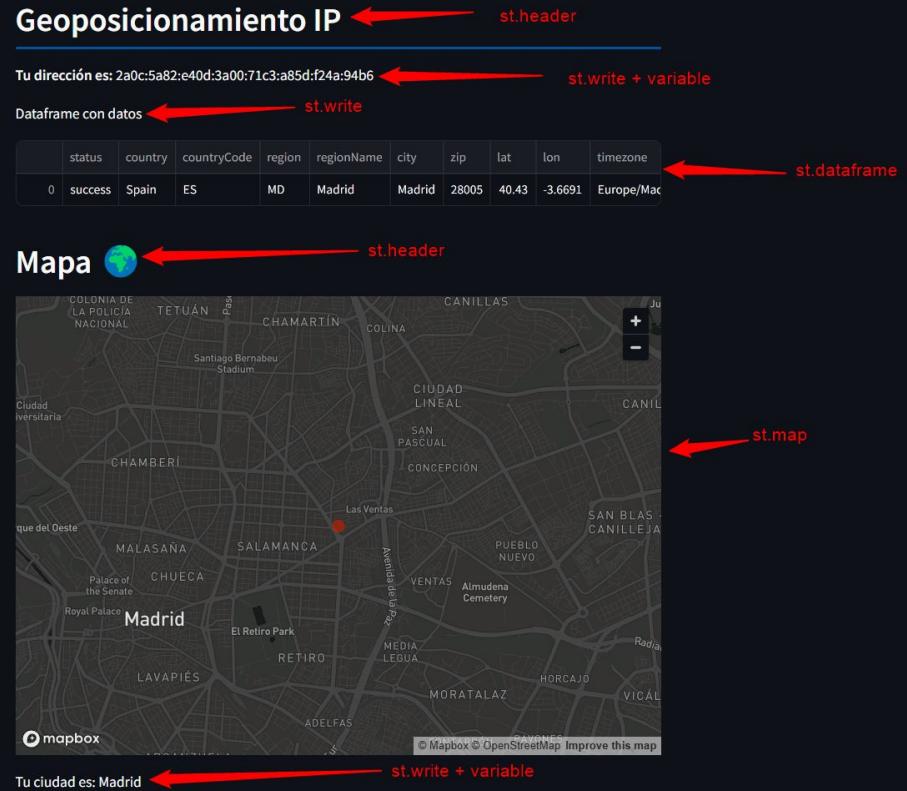
# Imprime solo latitud y longitud
lat = loc.json()['lat']
lon = loc.json()['lon']
print(f"Latitud: {lat}, Longitud: {lon}")
```

Pista para mapa: Funciona mejor con dataframe

```
df = pd.DataFrame({"lat": [lat], "lon": [lon]})  
st.map(df)
```

Bonus: Añade pestañas mostrando los datos en JSON y en tabla

```
tab1, tab2 = st.tabs(["RAW", "Tabla"])
with tab1:
    st.write(loc.json())
with tab2:
    st.table(loc.json())
```



Ejercicio 02: Pokédex

Crea una pokédex con los recursos que hemos visto.

La página debe contener lo siguiente:

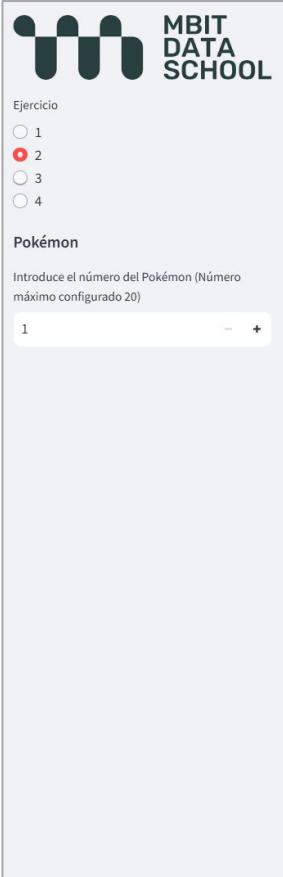
- Un título "Pokédex".
- Una imagen de Pokemon
- Un input para seleccionar el id del pokemon, en la barra lateral.

Después, tres columnas que contengan:

- La primera: Datos como nombre, altura, peso y experiencia base
- La segunda: La imagen del pokemon y su sonido característico.
- En la tercera: Las habilidades principales asociadas al pokémon.

Por último dos pestañas:

- En la primera, que aparezca una tabla con todos los pokémon y tres columnas: ID, nombre, Vista previa (imagen) y un check (True/False)
- En la segunda pestaña, un formulario para subir un csv y restaurar nuestro pokédex (tabla anterior)



Ejercicio 2

Pokédex


Pokémon

Datos	Aspecto	Habilidades
Nombre: bulbasaur	 0:00 / 0:00	overgrow chlorophyll
Altura: 7		
Peso: 6.9 kg		
Experiencia base: 64		

[Todos los Pokémon](#) | [Restaurar copia de seguridad](#)

Todos los pokémon

id	nombre	Vista previa	Visto
1	bulbasaur		<input checked="" type="checkbox"/>
2	ivysaur		<input type="checkbox"/>
3	venusaur		<input type="checkbox"/>

Ejercicio 03: Explorador de países

La página web <https://restcountries.com> contiene información acerca de los países.

Examina su API y realiza los siguientes ejercicios.

1. Muestra en un dataframe la información obtenida en el endpoint:
<https://restcountries.com/v3.1/all> (No funciona!
 Busca en la documentación por qué)
2. Añade 2 filtros en sidebar que permitan filtrar por número de población mínima y población máxima.
3. Por último, añade un st.selectbox que muestre el nombre de lista de países. Cuando se seleccione un país, que muestre la información básica del país ("Región", "Población", "Capital", "Idiomas", "Area") y la imagen de la bandera del país.



Ejercicio 3

Explorador de Países con REST Countries API

Esta aplicación interactiva utiliza la API pública [REST Countries](#) para mostrar información sobre países.
 Usa los filtros para explorar los datos.

Resultados

Mostrando 250 país(es)

	Nombre	Región	Población	Capital	Idiomas	Área (km²)	Bandera
0	South Georgia	Antarctic	30	King Edward Point	English	3,903	https://flagcdn.c
1	Grenada	Americas	112,519	St. George's	English	344	https://flagcdn.c
2	Switzerland	Europe	8,654,622	Bern	French, Swi	41,284	https://flagcdn.c
3	Sierra Leone	Africa	7,976,985	Freetown	English	71,740	https://flagcdn.c
4	Hungary	Europe	9,749,763	Budapest	Hungarian	93,026	https://flagcdn.c
5	Taiwan	Asia	23,503,349	Taipei	Chinese	36,193	https://flagcdn.c
6	Wallis and Futuna	Oceania	11,750	Mata-Utu	French	142	https://flagcdn.c
7	Barbados	Americas	287,371	Bridgetown	English	430	https://flagcdn.c
8	Pitcairn Islands	Oceania	56	Adamstown	English	47	https://flagcdn.c
9	Ivory Coast	Africa	26,378,275	Yamoussoukro	French	322,463	https://flagcdn.c

Selecciona un país para ver más detalles:

Detalles de Spain

Región: Europe

Población: 47351567

Capital: Madrid

Idiomas: Spanish, Catalan, Basque, Galician

Área: 505992.0 km²



77 | Streamlit



Ejercicio 04: Titanic

El dataset de Titanic ha sido ampliamente utilizado en proyectos de analítica de datos. Éste se encuentra en :
<https://www.kaggle.com/c/titanic/data>

Acceso al CSV: <https://raw.githubusercontent.com/datasets/titanic/master/titanic.csv>

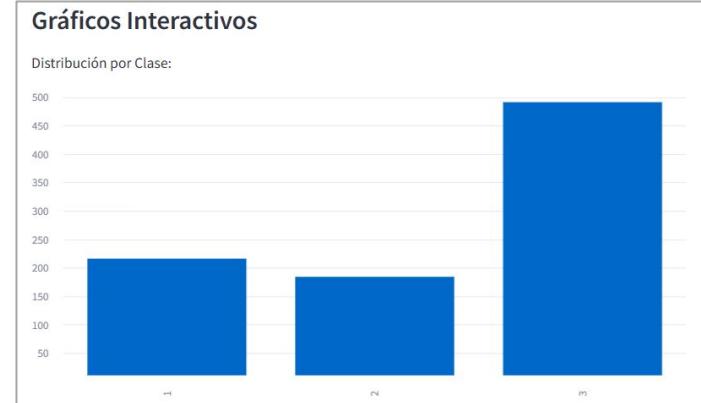
1. Carga el dataset en un dataframe y muestra dos dataframes, uno mostrando los datos de la tabla, y otro con la función `df.describe()` para ver la información básica.
2. Crea filtros en el sidebar que permitan filtrar por elementos como la clase (categoría) de los pasajeros: 1, 2 y 3
3. Crea un diagrama de barras que muestra el número de supervivientes por cada clase

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891	891	891	714	891	891	891
mean	446	0.3838	2.3086	29.6991	0.523	0.3816	32.2042
std	257.3538	0.4866	0.8361	14.5265	1.1027	0.8061	49.6934
min	1	0	1	0.42	0	0	0
25%	223.5	0	2	20.125	0	0	7.9104
50%	446	0	3	28	0	0	14.4542
75%	668.5	1	3	38	1	0	31
max	891	1	3	80	8	6	512.3292

Filtros

Selecciona clases:

1 ×
2 ×
3 ×



Ejercicio 05: Chatbot

Interacciona con Google AI Studio:

Obten tu API Token con tu cuenta Google.

Crea el formulario de forma que:

- Valide que tienes un token escrito, ¡tipo password!
- Tengas contenido en tu mensaje.
- Desaparezca los mensajes de informacion

```
if st.button("Enviar") and google_api_key and user_message:
    with st.spinner("Pensando..."):
        # URL de la API de Google Gemini
        url =
            "https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash-latest:generateContent?key"
            =(google_api_key)

        payload = {
            "contents": [
                {
                    "parts": [
                        {
                            "text": user_message
                        }
                    ]
                },
                "generationConfig": {
                    "temperature": 0.7,
                    "maxOutputTokens": 100
                }
            ],
            "headers": {
                "Content-Type": "application/json"
            }
        }

        # Hacer la petición
        response = requests.post(url, json=payload, headers=headers)

        if response.status_code == 200:
            response.json()

        # Extraer la respuesta
        if "candidates" in result and len(result["candidates"]) > 0:
            candidate = result["candidates"][0]
            if "content" in candidate and "parts" in candidate["content"]:
                bot_response = candidate["content"]["parts"][0]["text"]

                st.success("Respuesta generada:")
                st.write("Tu mensaje:")
                st.write(f"--- Tu mensaje")
                st.write("Respuesta del bot:")
                st.write(f"--- {bot_response}")
                st.write(f"--- {bot_response}")

        else:
            st.error(f"Error {response.status_code}: {response.text}")
```

Haz tu pregunta al bot de Google AI Studio



Google AI Studio

introduce tu token

X O

Escribe tu mensaje:

Enviar
X

Necesitas una API key de Google AI Studio para continuar.

Ejercicio 06: Conversacional con memoria

Copia ejercicio anterior y aplica cambios:

```

if ejercicio == "6":
    st.header("Ejercicio 6")
    with st.expander("Session State"):
        st.session_state
    [...]
#+++++#
    if "chat_history" not in st.session_state:
        st.session_state["chat_history"] = []
#-----
    [...]
#+++++#
    # Mostrar historial de la conversación
    if st.session_state["chat_history"]:
        st.markdown("---")
        st.markdown("## Historial de la conversación:")
        for i, (role, msg) in enumerate(st.session_state["chat_history"]):
            if role == "user":
                st.write(f"👤 {msg}")
            else:
                st.write(f"🤖 {msg}")
        st.markdown("---")
    [...]
#+++++#
    # Construir el historial para el payload con roles correctos
    contents = []
    for role, msg in st.session_state["chat_history"]:
        if role == "user":
            contents.append({"role": "user", "parts": [{"text": msg}]})
        elif role == "bot":
            contents.append({"role": "model", "parts": [{"text": msg}]})
    # Añadir el mensaje actual del usuario
    contents.append({"role": "user", "parts": [{"text": user_message}]})
#-----
    [...]
#+SUSTITUCION DE CONTENTS+++++
    "contents": contents,
#-----
    [...]
#+++++#
    # Guardar en el historial
    st.session_state["chat_history"].append(("user", user_message))
    st.session_state["chat_history"].append(("bot", bot_response))
#-----
    [...]
#+ PARA HACER REFRESCO de
    st.rerun()
#-----

```

Ejercicio 6

Session State

Chatbot con Google AI Studio (Gemini) con memoria



Introduce tu token

.....



Escribe tu mensaje:

¿Qué cantidad de población tiene?

Historial de la conversación:

👤 ¿En qué continente se encuentra el país Kenia?

🤖 Kenia se encuentra en el continente de África.

👤 ¿Cuál es la capital de ese país?

🤖 La capital de Kenia es Nairobi.

👤 ¿Qué cantidad de población tiene?

🤖 La población de Kenia es difícil de dar con exactitud ya que está en constante cambio. Sin embargo, según estimaciones recientes, la población de Kenia ronda los 55 millones de habitantes. Es importante tener en cuenta que esta cifra es una aproximación y puede variar ligeramente dependiendo de la fuente y la fecha de la estimación.



Ejercicio 07: IA generativa con Replicate → Texto a Voz

- Vamos a generar e importar un fichero de audio desde Replicate.

```
if st.button("Generar Audio") and replicate_api_key and user_message:
    with st.spinner("Generando audio..."):
        try:
            url = "https://api.replicate.com/v1/models/resemble-ai/chatterbox/predictions"
            headers = {
                "Authorization": f"Token {replicate_api_key}",
                "Content-Type": "application/json",
                "Prefer": "wait"
            }
            input_data = {
                "seed": 0,
                "prompt": user_message,
                "cfg_weight": 0.5,
                "temperature": 0.8,
                "exaggeration": 0.5
            }
            payload = {
                "input": input_data
            }
            response = requests.post(url, headers=headers, json=payload)
            if response.status_code in [200, 201]:
                result = response.json() # El resultado debe tener una URL de audio en 'output' o 'audio'
                audio_url = result.get("output")
                if not audio_url and "audio" in result:
                    audio_url = result["audio"]
                if audio_url:
                    st.success("Audio generado correctamente:")
                    st.audio(audio_url)
                    st.markdown(f"[Descargar audio]({audio_url})")
                else:
                    st.error("URL no válida")
            else:
                st.error(f"Error {response.status_code}: {response.text}")
        except Exception as e:
            st.error(f"Error: {e}")

```

Ejercicio 5

Text-to-Speech con Replicate



Genera un archivo de audio a partir de texto. Modelo 'resemble-ai/chatterbox'.

Ingrésa tu API Key de Replicate:

Escribe el texto que quieres convertir a voz:



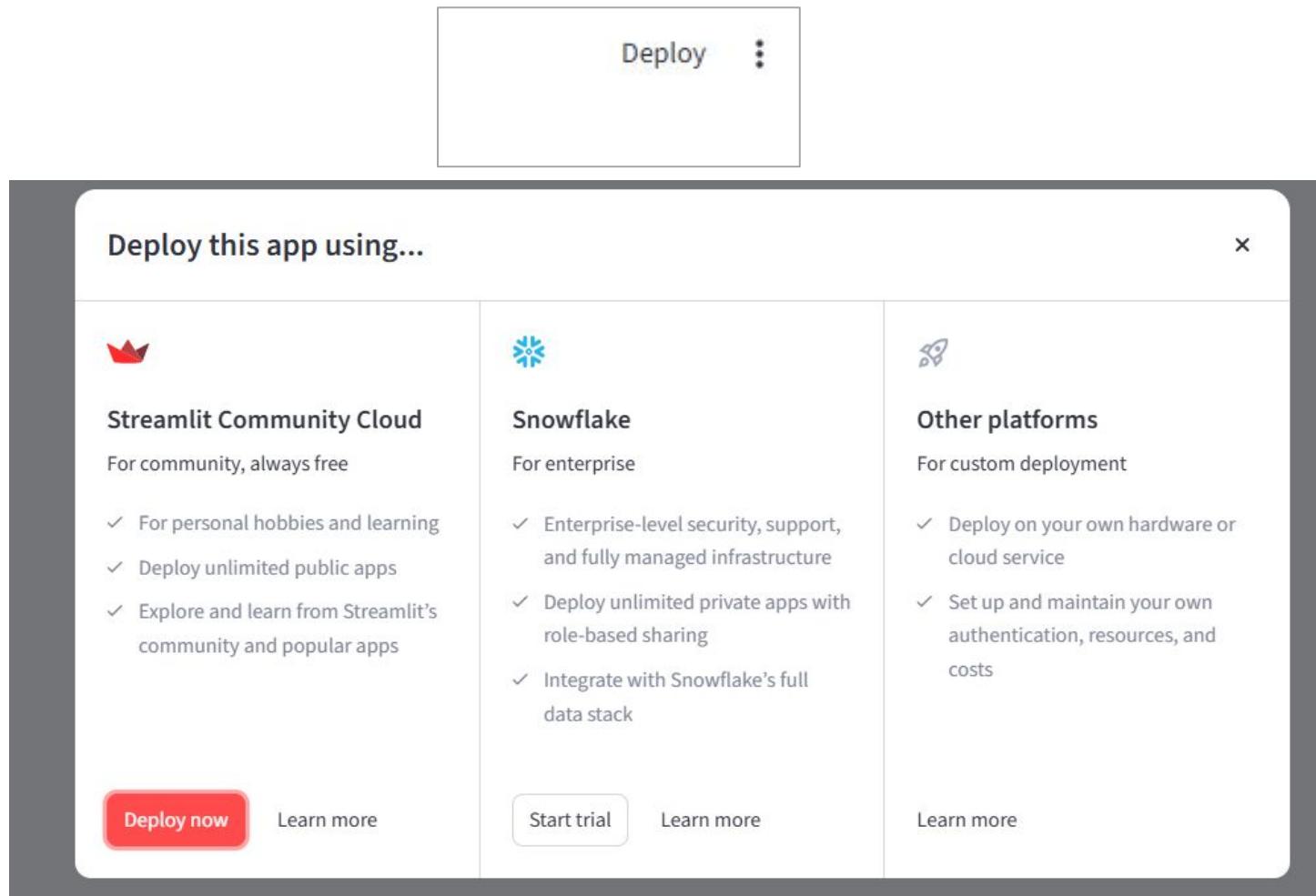
 Necesitas una API key de Replicate para continuar.

Extra: Vamos a incluir 3 columnas para personalizar sus campos

Despliegue en Streamlit Cloud

Despliegue de una aplicación

Podemos desplegar aplicaciones de forma gratuita si subimos el código fuente a un repositorio Git público.



The screenshot shows a modal window titled "Deploy this app using...". It lists three deployment options:

- Streamlit Community Cloud** (represented by a crown icon): For community, always free. Includes:
 - ✓ For personal hobbies and learning
 - ✓ Deploy unlimited public apps
 - ✓ Explore and learn from Streamlit's community and popular appsA red "Deploy now" button is at the bottom.
- Snowflake** (represented by a snowflake icon): For enterprise. Includes:
 - ✓ Enterprise-level security, support, and fully managed infrastructure
 - ✓ Deploy unlimited private apps with role-based sharing
 - ✓ Integrate with Snowflake's full data stackA "Start trial" button is at the bottom.
- Other platforms** (represented by a rocket icon): For custom deployment. Includes:
 - ✓ Deploy on your own hardware or cloud service
 - ✓ Set up and maintain your own authentication, resources, and costsA "Learn more" button is at the bottom.

Ejercicio para casa

Ejercicio Casa 1: Ficha de ciudadano

Construye una ficha de ciudadano a partir de la api NameFake (<https://api.namefake.com/>) una imagen a partir de ThisPersonDoesNotExist.

Habilita widgets que permita personalizar la API query:

*You can also specify gender and country for the generated name. For example:
[//api.name-fake.com/ukrainian-ukraine/male/](https://api.name-fake.com/ukrainian-ukraine/male/)*

La ficha debe contener al menos:

- Nombre
- Dirección:
- Número de Teléfono
- Correo electrónico
- Imagen (`st.image("https://thispersondoesnotexist.com")`)



Nombre: Juan Martinez
Dirección: Calle Gran Vía
Teléfono: +34 611 22 33 44
Correo electrónico: juanma@rtinez.es

Ejercicio Casa 2: Buscador de productos en Ebay

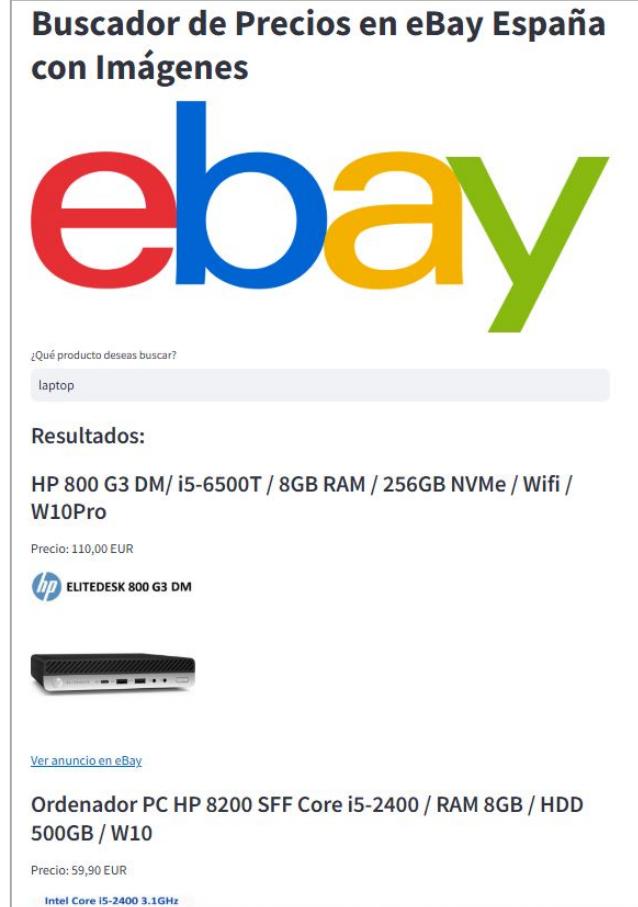
Combinando técnicas de Web Scraping con BeautifulSoup, vamos a crear un buscador rápido en eBay:

(Recuerda importar `from bs4 import BeautifulSoup`)

Con un término de búsqueda capture:

- El título (etiqueta `div.s-item__title`)
- El precio (etiqueta `span.s-item__price`)
- La imagen (etiqueta `img loading=eager`)
- El link ("`a.s-item__link`")

Buscador de Precios en eBay España con Imágenes



ebay

¿Qué producto deseas buscar?

laptop

Resultados:

HP 800 G3 DM/ i5-6500T / 8GB RAM / 256GB NVMe / Wifi / W10Pro

Precio: 110,00 EUR

 ELITEDSK 800 G3 DM



[Ver anuncio en eBay](#)

Ordenador PC HP 8200 SFF Core i5-2400 / RAM 8GB / HDD 500GB / W10

Precio: 59,90 EUR

Intel Core i5-2400 3.1GHz

Otros ejercicios propuestos

Ejercicio Extra 2: AirBnB

Para profundizar más, AirBnB ofrece una página con información relacionada a sus alquileres:
<https://insideairbnb.com/get-the-data/>

Inside Airbnb
Adding data to the debate

Data ▾ About Support ▾ Organise ▾ Donate!

Madrid, Comunidad de Madrid, Spain
11 September, 2024 (Explore)

Country/City	File Name	Description
Madrid	listings.csv.gz	Detailed Listings data
Madrid	calendar.csv.gz	Detailed Calendar Data
Madrid	reviews.csv.gz	Detailed Review Data
Madrid	listings.csv	Summary information and metrics for listings in Madrid (good for visualisations).
Madrid	reviews.csv	Summary Review data and Listing ID (to facilitate time based analytics and visualisations linked to a listing).
Madrid	neighbourhoods.csv	Neighbourhood list for geo filter. Sourced from city or open source GIS files.
Madrid	neighbourhoods.geojson	GeoJSON file of neighbourhoods of the city.



```
id,listing_url,scrape_id,last_scraped,source,name,description,neighborhood_overview,picture_url,host_id,host_url,host_name,host_since,host_location,ho
st_about,host_response_time,host_response_rate,host_acceptance_rate,host_is_superhost,host_thumbnail_url,host_picture_url,host_neighbourhood,host_list
ings_count,host_total_listings_count,host_verifications,host_has_profile_pic,host_identity_verified,neighbourhood,neighbourhood_cleansed,neighbourhood
_group_cleansed,latitude,longitude,property_type,accommodates,bathrooms,bathrooms_text,bedrooms,beds,amenities,price,minimum_nights,maximum_
nights,minimum_minimum_nights,maximum_minimum_nights,minimum_maximum_nights,maximum_maximum_nights,minimum_nights_avg_ntm,maximum_nights_avg_ntm,calen
dar_updated,has_availability,availability_30,availability_60,availability_90,availability_365,calendar_last_scraped,number_of_reviews,number_of_review
s_ltm,number_of_reviews_130d,first_review,last_review,review_scores_rating,review_scores_accuracy,review_scores_cleanliness,review_scores_checkin,revi
ew_scores_communication,review_scores_location,review_scores_value,license,instant_bookable,calculated_host_listings_count,calculated_host_listings_co
unt_entire_homes,calculated_host_listings_count_private_rooms,calculated_host_listings_count_shared_rooms,reviews_per_month
```



Visualización de datos: Streamlit

Alejandro Paredero

paredero@mbitschool.com

<https://es.linkedin.com/in/paredero>