### ****Configuración de la página****

python

st.set\_page\_config(layout="wide")

* **Descripción**: Configura la página de la aplicación en modo de ancho completo para mejorar la visualización.

### ****Creación de un objeto de sesión****

python

session\_state = st.session\_state

* **Descripción**: Se crea un objeto de sesión session\_state que permite almacenar y mantener valores durante la ejecución de la aplicación.

### ****Título de la página****

python

st.markdown("<h1 style='text-align: center;margin-top: -7px; margin-left: -940px'>Productos Almar</h1>", unsafe\_allow\_html=True)

* **Descripción**: Se utiliza st.markdown con HTML para mostrar un título estilizado al centro de la página.

### ****Conexión a Google Sheets y lectura de datos****

python

conn = st.connection("gsheets", type=GSheetsConnection)

df = conn.read(worksheet='Productos')

* **Descripción**: Se establece una conexión con Google Sheets utilizando streamlit\_gsheets y se lee el contenido de la hoja de cálculo llamada **Productos**, almacenándola en un DataFrame de Pandas.

### ****Manejo de la columna "Categoria Producto"****

1. **Extracción de letras del código:**

python

df['Categoria Producto'] = df.iloc[:, 0].str.extract(r'^([a-zA-Z]\*)', expand=False)

1. **Eliminación de la columna de su posición actual:**

python

categoria\_producto = df.pop('Categoria Producto')

1. **Reinserción de la columna en la posición deseada:**

python

df.insert(1, 'Categoria Producto', categoria\_producto)

* **Descripción**: Se agrega una columna extraída del código de productos, se manipula su posición dentro del DataFrame.

### ****Botón para resetear filtros****

python

if st.button('Resetear filtros'):

st.session\_state.reset\_filtros = True

st.session\_state.codigo\_seleccionado = 'Todos'

...

df\_filtrado = df

* **Descripción**: Crea un botón que, al ser presionado, resetea los filtros y selecciona los valores predeterminados en todos los campos del DataFrame.

### ****Filtros por diferentes columnas****

1. **Filtro por código:**

python

with col1:

codigo\_seleccionado = st.selectbox('Filtrar por código', ['Todos'] + df['Codigo'].dropna().unique().tolist(), key='codigo\_seleccionado')

if codigo\_seleccionado == 'Todos':

df\_codigo = df

else:

df\_codigo = df.loc[df['Codigo'] == codigo\_seleccionado]

1. **Filtro por categoría:**

python

with col2:

categoria\_seleccionada = st.selectbox('Filtrar por categoría', ['Todos'] + df\_codigo['Categoria Producto'].dropna().unique().tolist(), key='categoria\_seleccionada')

* **Descripción**: Crea selectores (selectbox) para filtrar el DataFrame según diferentes categorías: código, categoría, tela madre, corte, ancho, peso, color, etc. Los filtros se almacenan en session\_state y se aplica el filtrado dinámicamente.

### ****Grilla interactiva con paginación****

python

gd = GridOptionsBuilder.from\_dataframe(df\_filtrado)

gd.configure\_pagination(enabled=True)

gd.configure\_default\_column(editable=False, groupable=True)

gridoptions = gd.build()

grid\_table = AgGrid(df\_filtrado, gridOptions=gridoptions, selected\_rows=filas\_seleccionadas)

* **Descripción**: Utiliza st\_aggrid para mostrar una tabla interactiva con paginación, donde los datos pueden ser agrupados o editados según las configuraciones aplicadas.

### ****Actualización de filas seleccionadas****

python

if grid\_table['selected\_rows'] is not None:

selected\_rows = grid\_table['selected\_rows']

filas\_seleccionadas = [fila[0] for fila in selected\_rows]

st.session\_state.filas\_seleccionadas = filas\_seleccionadas

* **Descripción**: Almacena las filas seleccionadas de la tabla en session\_state para persistir la selección de filas incluso después de aplicar filtros.

### ****Gestión del carrito de compras****

1. **Creación de un DataFrame para el carrito:**

python

carrito\_df = pd.DataFrame(articulo\_seleccionado\_df[['Articulo','Peso', 'Precio/USD', 'PrecioKg/USD']])

carrito\_df['Cantidad'] = 1

1. **Configuración del carrito con AgGrid:**

python

carrito\_go = GridOptionsBuilder.from\_dataframe(carrito\_df)

carrito\_go.configure\_default\_column(editable=False)

carrito\_go.configure\_column("Cantidad", editable=True)

carrito\_lindo = AgGrid(carrito\_df, gridOptions=carrito\_go.build(), update\_mode=GridUpdateMode.MODEL\_CHANGED)

* **Descripción**: Se crea un DataFrame para representar el carrito de compras, permitiendo que se edite la cantidad de productos seleccionados en la tabla interactiva. La cantidad de artículos es editable directamente desde la interfaz.

### ****Importaciones utilizadas****

python

import streamlit as st

import pandas as pd

from streamlit\_gsheets import GSheetsConnection

from st\_aggrid import AgGrid, GridOptionsBuilder, GridUpdateMode

* **Descripción**: Se importan librerías necesarias como streamlit para la interfaz web, pandas para el manejo de datos, streamlit\_gsheets para la conexión con Google Sheets, y st\_aggrid para las tablas interactivas.

### ****Función para actualizar el carrito****

python

def update\_carrito\_df(grid\_table):

carrito\_lindo = carrito\_df.drop(columns=['Codigo', 'Tela Madre', 'Tela','Corte','Ancho','Peso','Color'])

carrito\_lindo.loc[:, 'Cantidad'] = grid\_table['data']['Cantidad']

* **Descripción**: Función que actualiza el DataFrame del carrito de compras cuando se realizan cambios en la cantidad de productos seleccionados.

### ****Resumen****

Este código implementa una interfaz de usuario en Streamlit que se conecta a una hoja de cálculo de Google Sheets, muestra una tabla interactiva con filtros dinámicos y permite gestionar un carrito de compras. Las filas seleccionadas se almacenan en una variable de sesión, y se puede editar la cantidad de artículos directamente desde la tabla.