**Código base:**

"""Modulo principal de la app de gestión de inventario: Registro de Ventas."""

*# Importaciones estándar de Python*

*import* os

*from* datetime *import* date

*# Importaciones de librerías externas*

*import* streamlit *as* st

*import* psycopg2

*import* pandas *as* pd

*from* dotenv *import* load\_dotenv

*# Ruta absoluta al archivo .env*

dotenv\_path *=* r"C:\Users\Robert\Desktop\INTERESES\PROGRAMA PYTHON\APP\_INVENTORY\Generalidades\.env"   *# pylint: disable=invalid-name*

load\_dotenv(dotenv\_path)

*# Conexión a la base de datos Neon*

conn *=* psycopg2.connect(

    host*=*os.getenv("DB\_HOST"),

    dbname*=*os.getenv("DB\_NAME"),

    user*=*os.getenv("DB\_USER"),

    password*=*os.getenv("DB\_PASSWORD"),

    port*=*os.getenv("DB\_PORT")

)

cur *=* conn.cursor()

st.title("🛒 Registro de Ventas")

col1, col2 *=* st.columns([1, 2])

*# Keys a limpiar*

keys\_a\_limpiar *=* ["sucursal", "almacen", "producto", "descripcion",

                  "cantidad", "stock", "usuario", "fecha\_vt"]

*# Limpieza controlada con flag*

*if* st.session\_state.get("limpiar\_form", False):

*for* k *in* keys\_a\_limpiar:

*if* k *==* "cantidad":

            st.session\_state[k] *=* 1

*elif* k *==* "stock":

            st.session\_state[k] *=* 0

*elif* k *==* "fecha\_vt":

            st.session\_state[k] *=* date.today()

*else*:

            st.session\_state[k] *=* ""

*del* st.session\_state["limpiar\_form"]

*# --- Columna 1: Formulario ---*

*with* col1:

*with* st.form("venta\_formulario"):

        st.subheader("Nueva Venta")

        sucursal *=* st.text\_input("Sucursal")

        almacen *=* st.text\_input("Almacén")

        producto *=* st.text\_input("Producto")

        descripcion *=* st.text\_input("Descripción")

        cantidad *=* st.number\_input("Cantidad Vendida", min\_value*=*1, step*=*1)

        stock *=* st.number\_input("Stock Actual (opcional)", min\_value*=*0, step*=*1)

        usuario *=* st.text\_input("Usuario que registra la venta")

        fecha\_vt *=* st.date\_input("Fecha de la venta", value*=*date.today())

        submitted *=* st.form\_submit\_button("💾 Guardar Venta")

*if* submitted:

*try*:

                cur.execute("""

                    INSERT INTO ventas (sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt)

                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

                """, (sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt))

                conn.commit()

                st.success("✅ Venta registrada correctamente.")

*# Limpieza del formulario al guardar exitosamente*

                st.rerun()

*except* Exception *as* e:

                st.error(f"❌ Error al guardar: {e}")

*# --- Columna 2: Visualización de la tabla ---*

*with* col2:

    st.subheader("📋 Registro actual de ventas")

*try*:

*# Cargar solo las columnas relevantes*

*# pylint: disable=invalid-name*

        query *=* """

            SELECT sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt

            FROM ventas

            ORDER BY fecha\_vt DESC

        """

        df *=* pd.read\_sql\_query(query, conn)

*# Mostrar sin índice, usando todo el ancho*

        st.dataframe(df.reset\_index(drop*=*True), use\_container\_width*=*True)

*except* Exception *as* e:

        st.error(f"Error al cargar la tabla: {e}")

**Modificación, última actualización: REFRESCA EL FORMULARIO – PROBLEMAS CON GUARDAR LA INFORMACION**

"""Modulo principal de la app de gestión de inventario: Registro de Ventas."""

*# Importaciones estándar de Python*

*import* os

*from* datetime *import* date

*# Importaciones de librerías externas*

*import* streamlit *as* st

*import* psycopg2

*import* pandas *as* pd

*from* dotenv *import* load\_dotenv

*# Ruta absoluta al archivo .env*

*# pylint: disable=invalid-name*

dotenv\_path *=* r"C:\Users\Robert\Desktop\INTERESES\PROGRAMA PYTHON\APP\_INVENTORY\Generalidades\.env"

load\_dotenv(dotenv\_path)

*# Función para obtener conexión a la base de datos*

def get\_db\_connection():

*return* psycopg2.connect(

        host*=*os.getenv("DB\_HOST"),

        dbname*=*os.getenv("DB\_NAME"),

        user*=*os.getenv("DB\_USER"),

        password*=*os.getenv("DB\_PASSWORD"),

        port*=*os.getenv("DB\_PORT")

    )

st.title("🛒 Registro de Ventas")

col1, col2 *=* st.columns([1, 2])

*# --- Columna 1: Formulario ---*

*with* col1:

    st.subheader("Nueva Venta")

*with* st.form(key*=*"venta\_form", clear\_on\_submit*=*True):

*# Campos del formulario (TODOS obligatorios)*

        sucursal *=* st.text\_input("Sucursal\*", key*=*"sucursal\_input")

        almacen *=* st.text\_input("Almacén\*", key*=*"almacen\_input")

        producto *=* st.text\_input("Producto\*", key*=*"producto\_input")

        descripcion *=* st.text\_input("Descripción\*", key*=*"descripcion\_input")

        cantidad *=* st.number\_input("Cantidad Vendida\*", min\_value*=*1,

                                   step*=*1, value*=*1, key*=*"cantidad\_input")

        stock *=* st.number\_input("Stock Actual\*", min\_value*=*0, step*=*1, value*=*0, key*=*"stock\_input")

        usuario *=* st.text\_input("Usuario que registra la venta\*", key*=*"usuario\_input")

        fecha\_vt *=* st.date\_input("Fecha de la venta\*", value*=*date.today(), key*=*"fecha\_input")

        submitted *=* st.form\_submit\_button("💾 Guardar Venta")

*if* submitted:

*# Validar TODOS los campos (todos son obligatorios)*

            campos\_validar *=* {

                "Sucursal": sucursal,

                "Almacén": almacen,

                "Producto": producto,

                "Descripción": descripcion,

                "Cantidad": cantidad,

                "Stock": stock,

                "Usuario": usuario,

                "Fecha": fecha\_vt

            }

            campos\_faltantes *=* [campo *for* campo, valor *in* campos\_validar.items()

*if* *not* valor *and* str(valor) *!=* "0"]

*if* campos\_faltantes:

                st.error(f"ERROR: Complete todos los campos obligatorios. Faltan: {', '.join(campos\_faltantes)}")

*else*:

*# Aquí iría la lógica para guardar la venta en la base de datos*

*pass*  *# <- Luego podés completar este bloque*

*# --- Carga de datos (fuera del formulario) ---*

conn *=* None *# pylint: disable=invalid-name*

*try*:

    conn *=* get\_db\_connection()

    cursor *=* conn.cursor()

    cursor.execute("SELECT \* FROM ventas ORDER BY fecha\_vt DESC")

    rows *=* cursor.fetchall()

    columns *=* [desc[0] *for* desc *in* cursor.description]

    data *=* pd.DataFrame(rows, columns*=*columns)

*except* Exception *as* e:

    st.error(f"Error al cargar los datos: {e}")

*finally*:

*if* conn:

        conn.close()

*# --- Columna 2: Visualización de la tabla ---*

*with* col2:

    st.subheader("📋 Registro actual de ventas")

*try*:

        conn *=* get\_db\_connection()

*with* conn.cursor() *as* cur:

            cur.execute("""

                SELECT sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt

                FROM ventas

                ORDER BY fecha\_vt DESC

            """)

            df *=* pd.DataFrame(

                cur.fetchall(),

                columns*=*["Sucursal", "Almacén", "Producto", "Descripción",

                         "Cantidad", "Stock", "Usuario", "Fecha Venta"]

            )

            st.dataframe(df, use\_container\_width*=*True)

*except* Exception *as* e:

        st.error(f"Error al cargar la tabla: {str(e)}")

*finally*:

*if* conn:

            conn.close()

*# Nota al pie sobre campos obligatorios*

st.markdown("""

<style>

.obligatorio {

    font-size: 0.8rem;

    color: #ff4b4b;

    margin-top: -10px;

    margin-bottom: 10px;

    font-weight: bold;

}

</style>

<div class="obligatorio">\* Todos los campos son obligatorios</div>

""", unsafe\_allow\_html*=*True)

**CASI PERFECTO, última actualización: REFRESCA EL FORMULARIO – PROBLEMAS CON GUARDAR LA INFORMACION**

"""Modulo principal de la app de gestión de inventario: Registro de Ventas."""

*# Importaciones estándar de Python*

*import* os

*from* datetime *import* date

*# Importaciones de librerías externas*

*import* streamlit *as* st

*import* psycopg2

*import* pandas *as* pd

*from* dotenv *import* load\_dotenv

*# Ruta absoluta al archivo .env*

*# pylint: disable=invalid-name*

dotenv\_path *=* r"C:\Users\Robert\Desktop\INTERESES\PROGRAMA PYTHON\APP\_INVENTORY\Generalidades\.env"

load\_dotenv(dotenv\_path)

*# Función para obtener conexión a la base de datos*

def get\_db\_connection():

*return* psycopg2.connect(

        host*=*os.getenv("DB\_HOST"),

        dbname*=*os.getenv("DB\_NAME"),

        user*=*os.getenv("DB\_USER"),

        password*=*os.getenv("DB\_PASSWORD"),

        port*=*os.getenv("DB\_PORT")

    )

st.title("🛒 Registro de Ventas")

col1, col2 *=* st.columns([1, 2])

*# --- Columna 1: Formulario ---*

*with* col1:

    st.subheader("Nueva Venta")

*with* st.form(key*=*"venta\_form", clear\_on\_submit*=*True):

        sucursal *=* st.text\_input("Sucursal\*", key*=*"sucursal\_input")

        almacen *=* st.text\_input("Almacén\*", key*=*"almacen\_input")

        producto *=* st.text\_input("Producto\*", key*=*"producto\_input")

        descripcion *=* st.text\_input("Descripción\*", key*=*"descripcion\_input")

        cantidad *=* st.number\_input("Cantidad Vendida\*", min\_value*=*1, step*=*1, value*=*1, key*=*"cantidad\_input")

        stock *=* st.number\_input("Stock Actual\*", min\_value*=*0, step*=*1, value*=*0, key*=*"stock\_input")

        usuario *=* st.text\_input("Usuario que registra la venta\*", key*=*"usuario\_input")

        fecha\_vt *=* st.date\_input("Fecha de la venta\*", value*=*date.today(), key*=*"fecha\_input")

        submitted *=* st.form\_submit\_button("💾 Guardar Venta")

*if* submitted:

*# Validar campos obligatorios*

            campos\_validar *=* {

                "Sucursal": sucursal,

                "Almacén": almacen,

                "Producto": producto,

                "Descripción": descripcion,

                "Cantidad": cantidad,

                "Stock": stock,

                "Usuario": usuario,

                "Fecha": fecha\_vt

            }

            campos\_faltantes *=* [campo *for* campo, valor *in* campos\_validar.items()

*if* (valor *==* "" *or* valor *is* None) *and* str(valor) *!=* "0"]

*if* campos\_faltantes:

                st.error(f"ERROR: Complete todos los campos obligatorios. Faltan: {', '.join(campos\_faltantes)}")

*else*:

                conn *=* None

*try*:

                    conn *=* get\_db\_connection()

*with* conn.cursor() *as* cur:

                        cur.execute("""

                            INSERT INTO ventas

                            (sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt)

                            VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

                        """, (sucursal, almacen, producto, descripcion,

                              cantidad, stock, usuario, fecha\_vt))

                        conn.commit()

                    st.success("✅ Venta registrada correctamente.")

*except* Exception *as* e:

                    st.error(f"❌ Error al guardar: {str(e)}")

*if* conn:

                        conn.rollback()

*finally*:

*if* conn:

                        conn.close()

*# --- Carga de datos (fuera del formulario) ---*

conn *=* None

*try*:

    conn *=* get\_db\_connection()

    cursor *=* conn.cursor()

    cursor.execute("""

        SELECT sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt

        FROM ventas

        ORDER BY fecha\_vt DESC

    """)

    rows *=* cursor.fetchall()

    columns *=* [desc[0] *for* desc *in* cursor.description]

    data *=* pd.DataFrame(rows, columns*=*columns)

*except* Exception *as* e:

    st.error(f"Error al cargar los datos: {str(e)}")

*finally*:

*if* conn:

        conn.close()

*# --- Columna 2: Visualización de la tabla ---*

*with* col2:

    st.subheader("📋 Registro actual de ventas")

    st.dataframe(data, use\_container\_width*=*True)

*# Nota al pie sobre campos obligatorios*

st.markdown("""

<style>

.obligatorio {

    font-size: 0.8rem;

    color: #ff4b4b;

    margin-top: -10px;

    margin-bottom: 10px;

    font-weight: bold;

}

</style>

<div class="obligatorio">\* Todos los campos son obligatorios</div>

""", unsafe\_allow\_html*=*True)

**CASI CASI PERFECTO, última actualización: REFRESCA EL FORMULARIO – PROBLEMAS CON GUARDAR LA INFORMACION, Si no se modifica cantidades no se guarda la información, es decir tengo que modificar todos los datos de columna 1:**

* **Hay que revisar, funciona, pero a veces se traba me parece. Veremos (no hay problema aún soy novato, pero vamos avanzando)**

"""Módulo principal de la app de gestión de inventario: Registro de Ventas."""

*# Importaciones estándar y externas*

*import* os

*from* datetime *import* date

*import* streamlit *as* st

*import* psycopg2

*import* pandas *as* pd

*from* dotenv *import* load\_dotenv

*# Ruta al archivo .env*

dotenv\_path *=* r"C:\Users\Robert\Desktop\INTERESES\PROGRAMA PYTHON\APP\_INVENTORY\Generalidades\.env" *# pylint: disable=invalid-name*

load\_dotenv(dotenv\_path)

*# Conexión a la base de datos*

def get\_db\_connection():

    """

    Establece y retorna una conexión a la base de datos PostgreSQL

    usando las variables de entorno configuradas en .env.

    """

*return* psycopg2.connect(

        host*=*os.getenv("DB\_HOST"),

        dbname*=*os.getenv("DB\_NAME"),

        user*=*os.getenv("DB\_USER"),

        password*=*os.getenv("DB\_PASSWORD"),

        port*=*os.getenv("DB\_PORT")

    )

*# Inicializar estado para controlar refresco*

*if* 'venta\_guardada' *not* *in* st.session\_state:

    st.session\_state.venta\_guardada *=* False

*# Título*

st.title("🛒 Registro de Ventas")

col1, col2 *=* st.columns([1, 2])

*# --- Columna 2: Visualización de la tabla ---*

*with* col2:

    st.subheader("📋 Registro actual de ventas")

*try*:

        conn *=* get\_db\_connection()

*with* conn.cursor() *as* cur:

            cur.execute("""

                SELECT sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt

                FROM ventas

                ORDER BY fecha\_vt DESC

            """)

            rows *=* cur.fetchall()

            columns *=* [desc[0] *for* desc *in* cur.description]

            data *=* pd.DataFrame(rows, columns*=*columns)

            st.dataframe(data, use\_container\_width*=*True)

*except* Exception *as* e:

        st.error(f"Error al cargar los datos: {str(e)}")

*finally*:

*if* conn:

            conn.close()

*# --- Columna 1: Formulario ---*

*with* col1:

    st.subheader("Nueva Venta")

*# Usar un form key único basado en el estado*

    form\_key *=* "venta\_form" *if* *not* st.session\_state.venta\_guardada *else* "venta\_form\_refresh"  *# pylint: disable=invalid-name*

*with* st.form(key*=*form\_key):

*# Campos del formulario con valores iniciales*

        sucursal *=* st.text\_input("Sucursal\*", value*=*"")

        almacen *=* st.text\_input("Almacén\*", value*=*"")

        producto *=* st.text\_input("Producto\*", value*=*"")

        descripcion *=* st.text\_input("Descripción\*", value*=*"")

        cantidad *=* st.number\_input("Cantidad Vendida\*", min\_value*=*1, step*=*1, value*=*1)

        stock *=* st.number\_input("Stock Actual\*", min\_value*=*0, step*=*1, value*=*0)

        usuario *=* st.text\_input("Usuario que registra la venta\*", value*=*"")

        fecha\_vt *=* st.date\_input("Fecha de la venta\*", value*=*date.today())

        submitted *=* st.form\_submit\_button("💾 Guardar Venta")

*if* submitted:

*# Validación de campos*

            campos\_validar *=* {

                "Sucursal": sucursal,

                "Almacén": almacen,

                "Producto": producto,

                "Descripción": descripcion,

                "Cantidad": cantidad,

                "Stock": stock,

                "Usuario": usuario,

                "Fecha": fecha\_vt

            }

            campos\_faltantes *=* [campo *for* campo, valor *in* campos\_validar.items()

*if* (valor *==* "" *or* valor *is* None) *and* str(valor) *!=* "0"]

*if* campos\_faltantes:

                st.error(f"ERROR: Complete todos los campos obligatorios. "

                         f"Faltan: {', '.join(campos\_faltantes)}")

*else*:

                conn *=* None *# pylint: disable=invalid-name*

*try*:

                    conn *=* get\_db\_connection()

*with* conn.cursor() *as* cur:

                        cur.execute("""

                            INSERT INTO ventas

                            (sucursal, almacen, producto, descripcion, cantidad, stock, usuario, fecha\_vt)

                            VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

                        """, (sucursal, almacen, producto, descripcion,

                              cantidad, stock, usuario, fecha\_vt))

                        conn.commit()

                    st.success("✅ Venta registrada correctamente.")

                    st.session\_state.venta\_guardada *=* True

                    st.rerun()  *# Forzar recarga completa*

*except* Exception *as* e:

                    st.error(f"❌ Error al guardar: {str(e)}")

*if* conn:

                        conn.rollback()

*finally*:

*if* conn:

                        conn.close()

*else*:

*# Resetear el estado si no se ha enviado el formulario*

            st.session\_state.venta\_guardada *=* False

*# Nota sobre campos obligatorios*

st.markdown("""

<style>

.obligatorio {

    font-size: 0.8rem;

    color: #ff4b4b;

    margin-top: -10px;

    margin-bottom: 10px;

    font-weight: bold;

}

</style>

<div class="obligatorio">\* Todos los campos son obligatorios</div>

""", unsafe\_allow\_html*=*True)