```
import tkinter as tk
from tkinter import messagebox
import pymysql

# Función para conectar a la base de datos
def conectar_bd():
    return pymysql.connect(
        host='localhost',
        user='root',
        password='sachita1023977432',  # Cambia por tu contraseña
        database='proyecto_StoreGame'
    )

def iniciar_sesion_usuario(ventana_principal):
    def verificar_usuario():
        correo = entry_correo.get()
        contrasena = entry_contrasena.get()
        mostrar_menu_usuario(ventana_principal, correo)

        try:
            conexion = conectar_bd()
            cursor = conexion.cursor()

            # Verificar si el correo y la contraseña son correctos
            cursor.execute("SELECT * FROM Usuario WHERE correo=%s AND
contraseña=%s", (correo, contrasena))
            usuario = cursor.fetchone()

            if usuario:
                messagebox.showinfo("Inicio de sesión exitoso",
f"Bienvenido {correo}")
                ventana_usuario.withdraw()  # Cerrar ventana de inicio
de sesión
                mostrar_menu_usuario(ventana_principal, correo)  #
Mostrar el menú del usuario
            else:
                messagebox.showerror("Error", "Correo o contraseña
incorrectos")

            conexion.close()

        except Exception as e:
```

```
            messagebox.showerror("Error de conexión", f"No se pudo
conectar a la base de datos: {e}")

    # Crear ventana de inicio de sesión
    ventana_usuario = tk.Toplevel(ventana_principal)
    ventana_usuario.title("Iniciar sesión como usuario")
    ventana_usuario.geometry("400x300")

    tk.Label(ventana_usuario, text="Correo:").pack(pady=10)
    entry_correo = tk.Entry(ventana_usuario)
    entry_correo.pack(pady=5)

    tk.Label(ventana_usuario, text="Contraseña:").pack(pady=10)
    entry_contrasena = tk.Entry(ventana_usuario, show="*")
    entry_contrasena.pack(pady=5)

    tk.Button(ventana_usuario, text="Iniciar sesión",
command=verificar_usuario).pack(pady=20)
    tk.Button(ventana_usuario, text="Volver al menú", command=lambda:
ventana_usuario.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)

# Función para mostrar el menú del usuario
def mostrar_menu_usuario(ventana_principal, correo):
    # Crear ventana del menú del usuario
    ventana_usuario = tk.Toplevel(ventana_principal)
    ventana_usuario.title("Menú Usuario")
    ventana_usuario.geometry("400x400")

    tk.Label(ventana_usuario, text="Menú Usuario", font=("Arial",
14)).pack(pady=20)

    # Botones del menú del usuario
    tk.Button(ventana_usuario, text="Comprar Juegos", command=lambda:
comprar_juegos(ventana_principal, correo)).pack(pady=10)
    tk.Button(ventana_usuario, text="Añadir saldo", command=lambda:
agregar_saldo(correo,ventana_usuario)).pack(pady=10)
    tk.Button(ventana_usuario, text="Ver lista de juegos",
command=lambda: ver_lista_juegos(ventana_usuario)).pack(pady=10)
    tk.Button(ventana_usuario, text="Volver al menú principal",
command=lambda: ventana_usuario.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)
```

```
# Función para iniciar sesión como administrador
def iniciar_sesion_admin(ventana_principal):
    def verificar_admin():
        contrasena_admin = entry_contrasena_admin.get()

        if contrasena_admin == "admin123":  # Cambia esto por la
contraseña del administrador
            messagebox.showinfo("Inicio de sesión exitoso",
"Bienvenido, Administrador")
            ventana_admin.withdraw()  # Cerrar ventana de inicio de
sesión
            mostrar_menu_admin(ventana_principal)  # Mostrar el menú
del administrador
        else:
            messagebox.showerror("Error", "Contraseña incorrecta")

    # Crear ventana de inicio de sesión
    ventana_admin = tk.Toplevel(ventana_principal)
    ventana_admin.title("Iniciar sesión como administrador")
    ventana_admin.geometry("400x300")

    tk.Label(ventana_admin, text="Contraseña de
administrador:").pack(pady=10)
    entry_contrasena_admin = tk.Entry(ventana_admin, show="*")
    entry_contrasena_admin.pack(pady=5)

    tk.Button(ventana_admin, text="Iniciar sesión",
command=verificar_admin).pack(pady=20)
    tk.Button(ventana_admin, text="Volver al menú", command=lambda:
ventana_admin.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)

# Función para mostrar el menú del administrador
def mostrar_menu_admin(ventana_principal):
    # Crear ventana del menú del administrador
    ventana_admin = tk.Toplevel(ventana_principal)
    ventana_admin.title("Menú Administrador")
    ventana_admin.geometry("400x400")

    tk.Label(ventana_admin, text="Menú Administrador", font=("Arial",
14)).pack(pady=20)

    # Botones del menú del administrador
```

```
    tk.Button(ventana_admin, text="Agregar nuevo juego",
command=lambda: agregar_juego(ventana_admin)).pack(pady=10)
    tk.Button(ventana_admin, text="Ver lista de juegos",
command=lambda: ver_lista_juegos(ventana_admin)).pack(pady=10)
    tk.Button(ventana_admin, text="Agregar unidades a un juego",
command=lambda: agregar_unidades_juego(ventana_admin)).pack(pady=10)
    tk.Button(ventana_admin, text="Ver recibos", command=lambda:
ver_lista_recibos(ventana_admin)).pack(pady=10)
    tk.Button(ventana_admin, text="Volver al menú principal",
command=lambda: ventana_admin.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)

# Función para agregar un nuevo juego
def agregar_juego(ventana_admin):
    def guardar_juego():
        nombre = entry_nombre.get()
        desarrollador = entry_desarrollador.get()
        tamano = entry_tamano.get()
        precio = entry_precio.get()
        unidadesD = entry_unidadesD.get()
        unidadesP = entry_unidadesP.get()
        fechal = entry_fechal.get()
        categoria = entry_categoria.get()

        try:
            conexion = conectar_bd()
            cursor = conexion.cursor()

            # Insertar el nuevo juego en la base de datos
            cursor.execute("INSERT INTO Juego (nombre, desarrollador,
tamaño, precio, unidades_disponibles, unudades_vendidas,
fecha_de_lanzamiento, gam_categoria) VALUES (%s, %s, %s, %s, %s, %s,
%s, %s)", (nombre, desarrollador, tamano, precio, unidadesD, unidadesP,
fechal, categoria))
            conexion.commit()

            messagebox.showinfo("Éxito", "Juego agregado exitosamente")
            conexion.close()
            ventana_agregar_juego.withdraw()  # Cerrar ventana de
agregar juego
            mostrar_menu_admin(ventana_admin)  # Volver al menú del
administrador
```

```
        except Exception as e:
            messagebox.showerror("Error de conexión", f"No se pudo
conectar a la base de datos: {e}")

    # Crear ventana para agregar un nuevo juego
    ventana_agregar_juego = tk.Toplevel(ventana_admin)
    ventana_agregar_juego.title("Agregar nuevo juego")
    ventana_agregar_juego.geometry("400x400")

    tk.Label(ventana_agregar_juego, text="Nombre del
juego:").pack(pady=10)
    entry_nombre = tk.Entry(ventana_agregar_juego)
    entry_nombre.pack(pady=5)

    tk.Label(ventana_agregar_juego,
text="Desarrollador:").pack(pady=10)
    entry_desarrollador = tk.Entry(ventana_agregar_juego)
    entry_desarrollador.pack(pady=5)

    tk.Label(ventana_agregar_juego, text="Tamaño (GB):").pack(pady=10)
    entry_tamano = tk.Entry(ventana_agregar_juego)
    entry_tamano.pack(pady=5)

    tk.Label(ventana_agregar_juego, text="Precio (USD):").pack(pady=10)
    entry_precio = tk.Entry(ventana_agregar_juego)
    entry_precio.pack(pady=5)

    tk.Label(ventana_agregar_juego, text="Stock:").pack(pady=10)
    entry_unidadesD = tk.Entry(ventana_agregar_juego)
    entry_unidadesD.pack(pady=5)

    tk.Label(ventana_agregar_juego, text="Ventas:").pack(pady=10)
    entry_unidadesP = tk.Entry(ventana_agregar_juego)
    entry_unidadesP.pack(pady=5)

    tk.Label(ventana_agregar_juego, text="Fecha:").pack(pady=10)
    entry_fechal = tk.Entry(ventana_agregar_juego)
    entry_fechal.pack(pady=5)

    tk.Label(ventana_agregar_juego, text="Categoria:").pack(pady=10)
    entry_categoria = tk.Entry(ventana_agregar_juego)
    entry_categoria.pack(pady=5)
```

```
    tk.Button(ventana_agregar_juego, text="Guardar juego",
command=guardar_juego).pack(pady=20)
    tk.Button(ventana_agregar_juego, text="Volver al menú",
command=lambda: ventana_agregar_juego.withdraw() or
mostrar_menu_admin(ventana_admin)).pack(pady=10)

# Función para agregar unidades a un juego
def agregar_unidades_juego(ventana_admin):
    def agregar_unidades():
        juego_nombre = entry_juego_nombre.get()
        unidades = int(entry_unidades.get())

        try:

            conexion = conectar_bd()
            cursor = conexion.cursor()

            # Actualizar las unidades de un juego
            cursor.execute("UPDATE Juego SET unidades_disponibles =
unidades_disponibles + %s WHERE nombre = %s", (unidades, juego_nombre))
            conexion.commit()

            messagebox.showinfo("Éxito", f"Unidades de {juego_nombre}
agregadas exitosamente")
            conexion.close()
            ventana_agregar_unidades.withdraw()
            mostrar_menu_admin(ventana_admin)

        except Exception as e:
            messagebox.showerror("Error de conexión", f"No se pudo
conectar a la base de datos: {e}")

    # Crear ventana para agregar unidades
    ventana_agregar_unidades = tk.Toplevel(ventana_admin)
    ventana_agregar_unidades.title("Agregar unidades a un juego")
    ventana_agregar_unidades.geometry("400x400")

    tk.Label(ventana_agregar_unidades, text="Nombre del
juego:").pack(pady=10)
    entry_juego_nombre = tk.Entry(ventana_agregar_unidades)
    entry_juego_nombre.pack(pady=5)
```

```
    tk.Label(ventana_agregar_unidades, text="Cantidad de unidades a
agregar:").pack(pady=10)
    entry_unidades = tk.Entry(ventana_agregar_unidades)
    entry_unidades.pack(pady=5)

    tk.Button(ventana_agregar_unidades, text="Agregar unidades",
command=agregar_unidades).pack(pady=20)
    tk.Button(ventana_agregar_unidades, text="Volver al menú",
command=lambda: ventana_agregar_unidades.withdraw() or
mostrar_menu_admin(ventana_admin)).pack(pady=10)

# Función para ver las facturas (recibos)
def ver_lista_recibos(ventana_principal):
    try:
        conexion = conectar_bd()
        cursor = conexion.cursor()

        # Obtener todos los recibos
        cursor.execute("SELECT * FROM Recibo")
        recibos = cursor.fetchall()

        # Crear ventana de lista de recibos
        ventana_recibos = tk.Toplevel(ventana_principal)
        ventana_recibos.title("Lista de Recibos")
        ventana_recibos.geometry("600x400")

        tk.Label(ventana_recibos, text="Lista de Recibos",
font=("Arial", 14)).pack(pady=10)

        # Mostrar los recibos en la ventana
        for recibo in recibos:
            recibo_info = f"Recibo ID: {recibo[0]}, Fecha: {recibo[1]},
Usuario ID: {recibo[2]}, Monto: ${recibo[6]}"
            tk.Label(ventana_recibos, text=recibo_info).pack(pady=5)

        tk.Button(ventana_recibos, text="Volver al menú",
command=lambda: ventana_recibos.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)
        conexion.close()

    except Exception as e:
        messagebox.showerror("Error de conexión", f"No se pudo obtener
la lista de recibos: {e}")
```

```
# Función para mostrar la lista de juegos
def ver_lista_juegos(ventana_principal):
    try:
        conexion = conectar_bd()
        cursor = conexion.cursor()

        # Obtener todos los juegos
        cursor.execute("SELECT * FROM Juego")
        juegos = cursor.fetchall()

        # Crear ventana de lista de juegos
        ventana_juegos = tk.Toplevel(ventana_principal)
        ventana_juegos.title("Lista de juegos")
        ventana_juegos.geometry("600x400")

        tk.Label(ventana_juegos, text="Lista de juegos", font=("Arial",
14)).pack(pady=10)

        # Mostrar los juegos en la ventana
        for juego in juegos:
            juego_info = f"Juego ID: {juego[0]}, Nombre: {juego[1]},
Desarrollador: {juego[2]}, Tamaño: {juego[3]}GB, Precio: ${juego[4]},
Unidades: {juego[5]}"
            tk.Label(ventana_juegos, text=juego_info).pack(pady=6)

        tk.Button(ventana_juegos, text="Volver al menú",
command=lambda: ventana_juegos.withdraw() or
mostrar_menu_admin(ventana_principal)).pack(pady=10)
        conexion.close()

    except Exception as e:
        messagebox.showerror("Error de conexión", f"No se pudo obtener
la lista de juegos: {e}")

# Función para crear nuevo usuario
def crear_nuevo_usuario(ventana_principal):
    def guardar_usuario():
        correo_nuevo = entry_correo_nuevo.get()
        contrasena_nueva = entry_contrasena_nueva.get()

        try:
            conexion = conectar_bd()
```

```
            cursor = conexion.cursor()

            # Insertar nuevo usuario en la base de datos
            cursor.execute("INSERT INTO Usuario (correo, contraseña)
VALUES (%s, %s)", (correo_nuevo, contrasena_nueva))
            conexion.commit()

            messagebox.showinfo("Éxito", "Usuario creado exitosamente")
            conexion.close()
            ventana_crear_usuario.withdraw()  # Cerrar ventana de crear
usuario
            mostrar_menu_principal(ventana_principal)  # Volver al menú
principal

        except Exception as e:
            messagebox.showerror("Error de conexión", f"No se pudo
conectar a la base de datos: {e}")

    # Crear ventana para ingresar datos del nuevo usuario
    ventana_crear_usuario = tk.Toplevel(ventana_principal)
    ventana_crear_usuario.title("Crear nuevo usuario")
    ventana_crear_usuario.geometry("400x300")

    tk.Label(ventana_crear_usuario, text="Correo:").pack(pady=10)
    entry_correo_nuevo = tk.Entry(ventana_crear_usuario)
    entry_correo_nuevo.pack(pady=5)

    tk.Label(ventana_crear_usuario, text="Contraseña:").pack(pady=10)
    entry_contrasena_nueva = tk.Entry(ventana_crear_usuario, show="*")
    entry_contrasena_nueva.pack(pady=5)

    tk.Button(ventana_crear_usuario, text="Crear usuario",
command=guardar_usuario).pack(pady=20)
    tk.Button(ventana_crear_usuario, text="Volver al menú",
command=lambda: ventana_crear_usuario.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)

# Función para mostrar la lista de juegos
def ver_lista_juegos(ventana_principal):
    try:
        conexion = conectar_bd()
        cursor = conexion.cursor()
```

```
        # Obtener todos los juegos
        cursor.execute("SELECT * FROM Juego")
        juegos = cursor.fetchall()

        # Crear ventana de lista de juegos
        ventana_juegos = tk.Toplevel(ventana_principal)
        ventana_juegos.title("Lista de juegos")
        ventana_juegos.geometry("600x400")

        tk.Label(ventana_juegos, text="Lista de juegos", font=("Arial",
14)).pack(pady=10)

        # Mostrar los juegos en la ventana
        for juego in juegos:
            juego_info = f"Nombre: {juego[0]}, Desarrollador:
{juego[1]}, Tamaño: {juego[2]} GB, Precio: {juego[3]} USD, Unidades
{juego[4]}"
            tk.Label(ventana_juegos, text=juego_info).pack(pady=6)

        tk.Button(ventana_juegos, text="Volver al menú",
command=lambda: ventana_juegos.withdraw() or
mostrar_menu_principal(ventana_principal)).pack(pady=10)
        conexion.close()

    except Exception as e:
        messagebox.showerror("Error de conexión", f"No se pudo conectar
a la base de datos: {e}")

def agregar_saldo(correo, ventana_usuario):
    def guardar_saldo():
        try:
            saldo_a_agregar = float(entry_saldo.get())  # Obtener el
saldo a agregar desde el campo de texto

            if saldo_a_agregar <= 0:
                messagebox.showerror("Error", "El saldo debe ser un
valor positivo")
                return

            # Conectar a la base de datos
            conexion = conectar_bd()
            cursor = conexion.cursor()
```

```
            # Actualizar el saldo del usuario en la base de datos
            cursor.execute("UPDATE Usuario SET saldo = saldo + %s WHERE
correo = %s", (saldo_a_agregar, correo))
            conexion.commit()

            messagebox.showinfo("Éxito", f"Se agregaron
${saldo_a_agregar} al saldo de {correo}")

            # Cerrar la ventana de agregar saldo y volver al menú
            conexion.close()
            ventana_agregar_saldo.withdraw()
            mostrar_menu_usuario(ventana_usuario, correo)

        except ValueError:
            messagebox.showerror("Error", "Por favor ingresa un valor
numérico válido")

    # Crear ventana para agregar saldo
    ventana_agregar_saldo = tk.Toplevel(ventana_usuario)
    ventana_agregar_saldo.title("Agregar saldo")
    ventana_agregar_saldo.geometry("400x200")

    tk.Label(ventana_agregar_saldo, text="Cantidad de saldo a
agregar:").pack(pady=10)
    entry_saldo = tk.Entry(ventana_agregar_saldo)
    entry_saldo.pack(pady=5)

    tk.Button(ventana_agregar_saldo, text="Agregar saldo",
command=guardar_saldo).pack(pady=20)
    tk.Button(ventana_agregar_saldo, text="Volver al menú",
command=lambda: ventana_agregar_saldo.withdraw() or
mostrar_menu_usuario(ventana_usuario, correo)).pack(pady=10)

def comprar_juegos(ventana_principal, correo):
    def calcular_descuento(carrito):
        # Contar juegos por categoría
        conteo_categorias = {
            'ROMPECABEZA': 0,
            'DEPORTES': 0,
            'ACCIÓN': 0
        }

        for juego, cantidad, _, categoria in carrito:
```

```
            if categoria in conteo_categorias:
                conteo_categorias[categoria] += cantidad

        # Verificar condiciones de descuento
        porcentaje_descuento = 0
        if conteo_categorias['ROMPECABEZA'] >= 25:
            porcentaje_descuento = max(porcentaje_descuento, 20)
        if conteo_categorias['DEPORTES'] >= 20 and
conteo_categorias['ACCIÓN'] >= 15:
            porcentaje_descuento = max(porcentaje_descuento, 15)

        return porcentaje_descuento

    def realizar_compra():
        try:
            # Verificar si hay juegos seleccionados
            if not lista_juegos.curselection():
                messagebox.showerror("Error", "Debe seleccionar al
menos un juego")
                return

            # Obtener la cantidad
            try:
                cantidad = int(entry_cantidad.get())
                if cantidad <= 0:
                    messagebox.showerror("Error", "La cantidad debe ser
mayor a 0")
                    return
            except ValueError:
                messagebox.showerror("Error", "Por favor ingrese un
número válido")
                return

            # Conectar a la base de datos
            conexion = conectar_bd()
            cursor = conexion.cursor()

            # Calcular costo total
            costo_total = 0
            carrito = []

            for item in lista_juegos.curselection():
                # Obtener nombre del juego de la selección
```

```
                juego_info = lista_juegos.get(item)
                juego = juego_info.split(" - ")[0]

                # Obtener precio, stock y categoría del juego
                cursor.execute("SELECT precio, unidades_disponibles,
gam_categoria FROM Juego WHERE nombre = %s", (juego,))
                resultado = cursor.fetchone()

                if not resultado:
                    messagebox.showerror("Error", f"No se encontró el
juego {juego}")
                    conexion.close()
                    return

                precio, stock_disponible, categoria = resultado

                # Verificar stock
                if cantidad > stock_disponible:
                    messagebox.showerror("Error", f"No hay suficientes
unidades de {juego}")
                    conexion.close()
                    return

                # Calcular subtotal para este juego
                subtotal = precio * cantidad
                costo_total += subtotal

                # Agregar al carrito (incluir categoría)
                carrito.append((juego, cantidad, precio, categoria))

            # Calcular descuento
            porcentaje_descuento = calcular_descuento(carrito)
            descuento = (costo_total * porcentaje_descuento) / 100
            costo_final = costo_total - descuento

            # Verificar saldo del usuario
            cursor.execute("SELECT saldo FROM Usuario WHERE correo =
%s", (correo,))
            resultado_saldo = cursor.fetchone()

            if not resultado_saldo:
                messagebox.showerror("Error", "No se encontró el
usuario")
```

```
                conexion.close()
                return

            saldo_usuario = resultado_saldo[0]

            if saldo_usuario < costo_final:
                messagebox.showerror("Error", "Saldo insuficiente")
                conexion.close()
                return

            # Realizar la compra
            for juego, cantidad, precio, _ in carrito:
                # Actualizar stock del juego
                cursor.execute("UPDATE Juego SET unidades_disponibles =
unidades_disponibles - %s WHERE nombre = %s",
                               (cantidad, juego))

                # Insertar recibo
                cursor.execute("""
                    INSERT INTO Recibo
                    (fecha, usuario_correo, juego_nombre, cantidad,
precio_unitario, total)
                    VALUES (NOW(), %s, %s, %s, %s, %s)
                """, (correo, juego, cantidad, precio, cantidad *
precio))

            # Descontar saldo del usuario
            cursor.execute("UPDATE Usuario SET saldo = saldo - %s WHERE
correo = %s",
                           (costo_final, correo))

            # Confirmar transacción
            conexion.commit()

            # Mostrar mensaje con detalles del descuento
            if porcentaje_descuento > 0:
                mensaje = f"""
                Compra realizada exitosamente
                Subtotal: ${costo_total:.2f}
                Descuento ({porcentaje_descuento}%): ${descuento:.2f}
                Total final: ${costo_final:.2f}
                """
            else:
```

```
                mensaje = f"Compra realizada por ${costo_final:.2f}"

            messagebox.showinfo("Compra exitosa", mensaje)
            conexion.close()
            ventana_compra.withdraw()

        except Exception as e:
            messagebox.showerror("Error", f"Error al realizar la
compra: {e}")

    # Crear ventana de compra
    ventana_compra = tk.Toplevel(ventana_principal)
    ventana_compra.title("Comprar Juegos")
    ventana_compra.geometry("500x600")

    # Mostrar información de descuentos disponibles
    info_descuentos = """
    Descuentos disponibles:
    - 20% al comprar 25 o más juegos de ROMPECABEZA
    - 15% al comprar 20 o más juegos de DEPORTES y 15 o más de ACCIÓN
    """
    tk.Label(ventana_compra, text=info_descuentos,
justify=tk.LEFT).pack(pady=10)

    # Obtener lista de juegos disponibles
    try:
        conexion = conectar_bd()
        cursor = conexion.cursor()
        cursor.execute("SELECT nombre, precio, unidades_disponibles,
gam_categoria FROM Juego WHERE unidades_disponibles > 0")
        juegos = cursor.fetchall()
        conexion.close()
    except Exception as e:
        messagebox.showerror("Error", f"No se pudo obtener la lista de
juegos: {e}")
        return

    # Crear lista de juegos
    tk.Label(ventana_compra, text="Selecciona los juegos a
comprar:").pack(pady=10)
    lista_juegos = tk.Listbox(ventana_compra, selectmode=tk.MULTIPLE,
width=50)
    for juego in juegos:
```

```
        lista_juegos.insert(tk.END, f"{juego[0]} - ${juego[1]} (Stock:
{juego[2]}) - {juego[3]}")
    lista_juegos.pack(pady=10)

    # Entrada de cantidad
    tk.Label(ventana_compra, text="Cantidad de cada juego
seleccionado:").pack(pady=5)
    entry_cantidad = tk.Entry(ventana_compra)
    entry_cantidad.pack(pady=5)

    # Botón de compra
    tk.Button(ventana_compra, text="Realizar Compra",
command=realizar_compra).pack(pady=10)

# Función para mostrar el menú principal
def mostrar_menu_principal(ventana_principal=None):
    if ventana_principal is None:
        ventana_principal = tk.Tk()
        ventana_principal.title("Menú Principal")
        ventana_principal.geometry("400x400")

    tk.Label(ventana_principal, text="Menú Principal", font=("Arial",
14)).pack(pady=20)

    # Botones del menú principal
    tk.Button(ventana_principal, text="Iniciar sesión como usuario",
command=lambda:
iniciar_sesion_usuario(ventana_principal)).pack(pady=10)
    tk.Button(ventana_principal, text="Iniciar sesión como
administrador", command=lambda:
iniciar_sesion_admin(ventana_principal)).pack(pady=10)
    tk.Button(ventana_principal, text="Crear nuevo usuario",
command=lambda: crear_nuevo_usuario(ventana_principal)).pack(pady=10)
    tk.Button(ventana_principal, text="Ver lista de juegos",
command=lambda: ver_lista_juegos(ventana_principal)).pack(pady=10)

    ventana_principal.mainloop()

# Llamamos a la función para mostrar el menú principal al iniciar
mostrar_menu_principal()
```

```
CREATE DATABASE proyecto_StoreGame;

USE  proyecto_StoreGame;

CREATE TABLE  Usuario(
	id_usuario INT PRIMARY KEY NOT NULL,
	correo VARCHAR(50) UNIQUE NOT NULL,
	contraseña VARCHAR(15) NOT NULL
);

CREATE TABLE Factura(
	id_factura INT PRIMARY KEY NOT NULL,
	fecha DATE NOT NULL,
	nombre_empresa VARCHAR(50) NOT NULL,
	nit VARCHAR(20) NOT NULL,
	cod_usuario INT,
	rec_correo VARCHAR(50),
	FOREIGN KEY(cod_usuario) REFERENCES Usuario(id_usuario),
	FOREIGN KEY(rec_correo) REFERENCES Usuario(correo)
);

CREATE TABLE TipoJuego(
	id_categoria INT PRIMARY KEY NOT NULL,
	categoria VARCHAR(20) UNIQUE NOT NULL
);

CREATE TABLE Juego (
	nombre VARCHAR(50) PRIMARY KEY NOT NULL,
	desarrollador VARCHAR(50) NOT NULL,
	tamaño FLOAT NOT NULL,
	precio FLOAT NOT NULL,
	unidades_disponibles SMALLINT NOT NULL,
	unudades_vendidas SMALLINT NOT NULL,
	fecha_de_lanzamiento DATE NOT NULL,
	gam_categoria VARCHAR (20),
	FOREIGN KEY (gam_categoria) REFERENCES TipoJuego(categoria)
);

SELECT*FROM Juego;
SELECT*FROM TipoJuego;
SELECT*FROM Usuario;
SELECT*FROM Recibo;


ALTER TABLE TipoJuego ADD INDEX buscar_categoria(categoria);
ALTER TABLE Juego ADD INDEX buscar_por_precio(precio);
```

```
 RENAME TABLE Factura TO Recibo;

INSERT INTO Usuario (id_usuario, correo, contraseña)
VALUES (2, 'usariozz2m@uan.edu.co', 45783), (3, 'dguayara32@uan.edu.co', 89890), (4,
'jcardoso44@uan.edu.co', 23451), (5, 'dzartha33@uan.edu.co', 90911), (6,
'arueda33@uan.edu.co', 31456), (7, 'mreyes22@uan.edu.co', 11111);

ALTER TABLE Juego
CHANGE unudades_vendidas unidades_vendidas INT;

INSERT INTO Juego (nombre, desarrollador, tamaño, precio, unidades_disponibles,
unidades_vendidas, fecha_de_lanzamiento)
Values ('The adventure of Zelda', 'Nintendo', 8.5, 30, 5, 30, '2017-03-3');


CREATE USER 'DavidZartha'@'3306' IDENTIFIED BY 'My.Territory/SQL';
GRANT SELECT, INSERT ON proyecto_storegame.Usuario TO 'DavidZartha'@'3306';
GRANT SELECT, INSERT ON proyecto_storegame.Juego TO 'DavidZartha'@'3306';
REVOKE SELECT, INSERT ON proyecto_storegame.Juego FROM 'DavidZartha'@'3306';

SHOW GRANTS FOR 'DavidZartha'@'3306';

CREATE VIEW Usuario_VISION AS
SELECT id_usuario, correo
FROM Usuario
WHERE id_usuario >= 4 and id_usuario <= 6;

SELECT*FROM Usuario_VISION;

SELECT id_usuario, correo FROM Usuario;

ALTER TABLE Recibo MODIFY id_factura INT AUTO_INCREMENT;
ALTER TABLE Usuario
ADD COLUMN saldo DECIMAL(10, 2) DEFAULT 0;



DESCRIBE Usuario;
DESCRIBE Juego;

-- 1. Eliminar la clave externa
ALTER TABLE Recibo DROP FOREIGN KEY recibo_ibfk_1;
drop table juego;
-- 2. Modificar la columna id_usuario en la tabla Usuario para que sea autoincrementable
ALTER TABLE Usuario MODIFY id_usuario INT AUTO_INCREMENT;

-- 3. Volver a agregar la clave externa en la tabla Recibo
```

```
ALTER TABLE Recibo ADD CONSTRAINT recibo_ibfk_1 FOREIGN KEY (cod_usuario)
REFERENCES Usuario(id_usuario);

ALTER TABLE Recibo ADD COLUMN PrecioT float;

UPDATE Recibo set PrecioT=20 where id_factura=1;


CREATE TABLE Juego (
        nombre VARCHAR(50) PRIMARY KEY NOT NULL,
        desarrollador VARCHAR(50) NOT NULL,
        tamaño FLOAT NOT NULL,
        precio FLOAT NOT NULL,
        unidades_disponibles SMALLINT NOT NULL,
        unudades_vendidas SMALLINT NOT NULL,
        fecha_de_lanzamiento DATE NOT NULL,
        gam_categoria VARCHAR(20) NOT NULL
);

drop table recibo;
CREATE TABLE Recibo (
        id INT AUTO_INCREMENT PRIMARY KEY,
        fecha DATETIME,
        usuario_correo VARCHAR(100),  -- Asegúrate de que este campo exista
        juego_nombre VARCHAR(100),        -- Nombre del juego comprado
        cantidad INT,             -- Cantidad de juegos comprados
        precio_unitario DECIMAL(10,2),-- Precio de cada juego
        total DECIMAL(10,2)             -- Precio total de la compra
);
```