Repositorio:

https://github.com/lucianoilujan/MonorepositorioProgramacion

Tema: Gestión de restaurante

Descripción:

Soy el gerente de un restaurante y me enfrento a diario al desafío de gestionar de manera eficiente las compras de inventario, la relación con los proveedores y el control de las ventas. Para optimizar estos procesos, he decidido desarrollar una aplicación que me permita centralizar toda la información en un solo lugar.

Funcionalidades:

- Administración de proveedores: La aplicación me permitirá crear un registro de proveedores, incluyendo información de contacto, datos fiscales, historial de compras y productos ofrecidos.
- Gestión de inventario: Podré registrar la entrada y salida de productos, controlar el stock disponible y generar alertas de stock bajo.
- Control de compras: La aplicación me permitirá realizar pedidos a los proveedores, generar órdenes de compra y monitorear el estado de los pedidos.
- Registro de ventas: Podré registrar las ventas realizadas, incluyendo detalles de los productos vendidos, precios, cantidades y formas de pago.
- Análisis de datos: La aplicación me proporcionará informes y estadísticas sobre ventas, compras, stock e inventario, lo que me permitirá tomar decisiones más informadas sobre el negocio.

Beneficios:

Esta aplicación me permitirá:

- Ahorrar tiempo: Podré optimizar los procesos de compras, inventario y ventas, lo que me permitirá ahorrar tiempo y esfuerzo.
- Mejorar la eficiencia: La aplicación me permitirá tener un mayor control sobre el inventario, las compras y las ventas, lo que me permitirá mejorar la eficiencia del negocio.
- Tomar mejores decisiones: Los informes y estadísticas proporcionados por la aplicación me permitirán tomar decisiones más informadas sobre el negocio, como por ejemplo, qué productos comprar, a qué proveedores comprar y cómo optimizar las estrategias de ventas.
- Mejorar la relación con los proveedores: La aplicación me permitirá mantener una mejor comunicación con los proveedores, lo que me permitirá obtener mejores precios y condiciones de compra.

Consideraciones adicionales:

- La aplicación podría incluir una función de gestión de recetas que me permita crear y administrar las recetas de los platos que se sirven en el restaurante.
- La aplicación podría integrarse con un sistema de punto de venta (POS) para que pueda registrar las ventas directamente desde el restaurante.

- La aplicación podría incluir una función de gestión de empleados que me permita controlar los horarios de trabajo, las nóminas y otras tareas relacionadas con el personal.

Estoy seguro de que esta aplicación sería una herramienta invaluable para cualquier gerente de restaurante. Me permitiría optimizar los procesos de gestión, mejorar la eficiencia del negocio y tomar mejores decisiones para el éxito del restaurante.

Modelo Entidad-Relación (MER) con Usuarios para Gestión de Restaurante

Entidades:

1. Proveedor:
   - proveedor_id (ID único del proveedor)
   - nombre (Nombre del proveedor)
   - direccion (Dirección del proveedor)
   - telefono (Teléfono del proveedor)
   - correo_electronico (Correo electrónico del proveedor)
   - datos_fiscales (Información fiscal del proveedor)

2. Producto:

   - producto_id (ID único del producto)
   - nombre (Nombre del producto)
   - descripcion (Descripción del producto)
   - precio_costo (Precio de costo del producto)
   - precio_venta (Precio de venta del producto)
   - categoria (Categoría del producto)
   - proveedor_id (ID del proveedor que suministra el producto) (Clave foránea a la tabla Proveedor)

3. Compra:

   - compra_id (ID único de la compra)
   - fecha_compra (Fecha de la compra)
   - proveedor_id (ID del proveedor al que se le realizó la compra) (Clave foránea a la tabla Proveedor)
   - estado (Estado de la compra: "En curso", "Completada", "Cancelada")
   - usuario_id (ID del usuario que realizó la compra) (Clave foránea a la tabla Usuario)

4. DetalleCompra:

- detalle_compra_id (ID único del detalle de la compra)
- compra_id (ID de la compra a la que pertenece el detalle) (Clave foránea a la tabla Compra)
- producto_id (ID del producto comprado) (Clave foránea a la tabla Producto)
- cantidad (Cantidad del producto comprado)
- precio_compra (Precio de compra del producto en la compra)

5. Venta:

- venta_id (ID único de la venta)
- fecha_venta (Fecha de la venta)
- estado (Estado de la venta: "En curso", "Completada", "Cancelada")
- usuario_id (ID del usuario que realizó la venta) (Clave foránea a la tabla Usuario)

6. DetalleVenta:

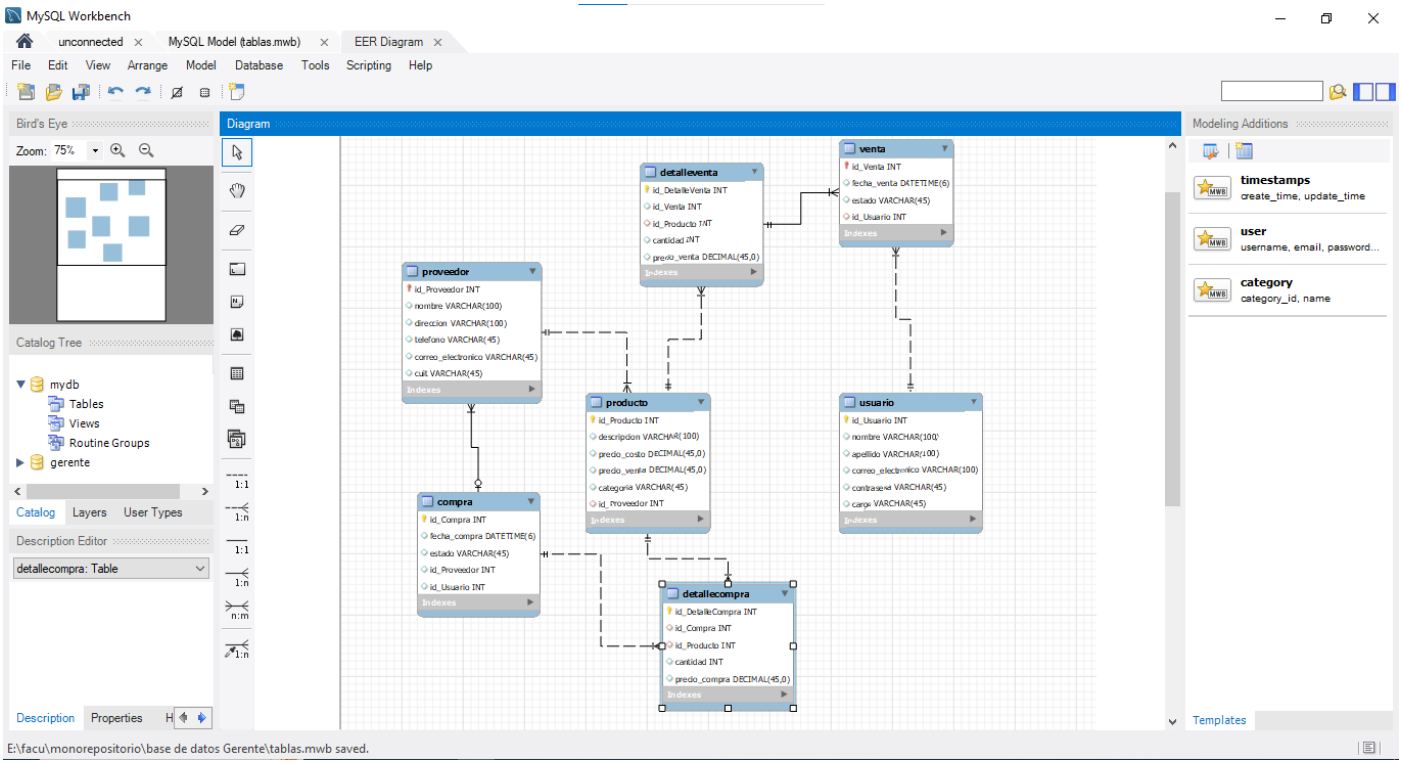
- detalle_venta_id (ID único del detalle de la venta)
- venta_id (ID de la venta a la que pertenece el detalle) (Clave foránea a la tabla Venta)
- producto_id (ID del producto vendido) (Clave foránea a la tabla Producto)
- cantidad (Cantidad del producto vendido)
- precio_venta (Precio de venta del producto en la venta)

7. Usuario:

- usuario_id (ID único del usuario)
- nombre (Nombre del usuario)
- apellido (Apellido del usuario)
- correo_electronico (Correo electrónico del usuario)
- contrasena (Contraseña del usuario)
- rol (Rol del usuario: "Administrador", "Empleado", "Cliente")

Relaciones:

- Un proveedor puede tener varios productos.
- Un producto pertenece a un solo proveedor.
- Una compra se realiza a un solo proveedor.
- Una compra se compone de varios detalles de compra.
- Un detalle de compra pertenece a una sola compra.
- Un detalle de compra se refiere a un solo producto.
- Una venta se realiza en una sola fecha.
- Una venta se compone de varios detalles de venta.
- Un detalle de venta pertenece a una sola venta.
- Un detalle de venta se refiere a un solo producto.
- Un usuario puede realizar varias ventas.
- Una venta está asociada a un solo usuario.
- Un usuario puede realizar varias compras.
- Una compra está asociada a un solo usuario.



Programa:

```
import mysql.connector

from mysql.connector import Error

def connect_db():
    """Conéctrse a la base de datos MySQL."""
    try:
        conexion = mysql.connector.connect(
            host='localhost',
            port=3306,
            user='root',
            password='admin',
            db='gerente'
        )
        print("Conexión exitosa a la base de datos 'gerente'")
        return conexion
    except Error as ex:
        print("Error en la conexión", ex)
        return None

def menu_principal():
    """Mostrar el menú principal."""
    while True:
        print("\nMenú principal:")
        print("1. Administración de proveedores")
        print("2. Gestión de inventario")
        print("3. Control de compras")
        print("4. Registro de ventas")
        print("5. Análisis de datos")
        print("6. Salir")

        opcion = input("Ingrese la opción deseada: ")

        if opcion == "1":
            menu_administracion_proveedores()
        elif opcion == "2":
            menu_gestion_inventario()
        elif opcion == "3":
            menu_control_compras()
        elif opcion == "4":
            menu_registro_ventas()
        elif opcion == "5":
            menu_analisis_datos()
        elif opcion == "6":
            print("Saliendo del programa...")
            break
```

```
        else:
            print("Opción inválida. Intente nuevamente.")

def actualizar_venta(id_venta, nuevo_estado):
    """Actualizar un registro de venta."""
    db = connect_db()
    if not db:
        return

    sql = """
        UPDATE venta
        SET estado = %s
        WHERE id_Venta = %s
    """

    try:
        cursor = db.cursor()
        cursor.execute(sql, (nuevo_estado, id_venta))
        db.commit()
        print(f"Venta con ID {id_venta} actualizada a estado '{nuevo_estado}'")
    except Error as e:
        print(f"Error al actualizar venta: {e}")
    finally:
        if db:
            db.close()

def menu_gestion_inventario():
    """Mostrar el menú de gestión de inventario."""
    while True:
        print("\nMenú de gestión de inventario:")
        print("1. Agregar producto al inventario")
        print("2. Actualizar información de producto")
        print("3. Eliminar producto del inventario")
        print("4. Ver inventario")
        print("5. Volver al menú principal")

        opcion = input("Ingrese la opción deseada: ")

        if opcion == "1":
            agregar_producto_inventario()
        elif opcion == "2":
            actualizar_informacion_producto()
        elif opcion == "3":
            eliminar_producto_inventario()
        elif opcion == "4":
            ver_inventario()
        elif opcion == "5":
            break
        else:
```

```
            print("Opción inválida. Intente nuevamente.")

def menu_administracion_proveedores():
    """Mostrar el menú de administración de proveedores."""
    while True:
        print("\nMenú de administración de proveedores:")
        print("1. Registrar nuevo proveedor")
        print("2. Consultar proveedores")
        print("3. Actualizar información de proveedor")
        print("4. Eliminar proveedor")
        print("5. Volver al menú principal")

        opcion = input("Ingrese la opción deseada: ")

        if opcion == "1":
            registrar_proveedor()
        elif opcion == "2":
            consultar_proveedores()
        elif opcion == "3":
            actualizar_proveedor()
        elif opcion == "4":
            eliminar_proveedor()
        elif opcion == "5":
            break
        else:
            print("Opción inválida. Intente nuevamente.")

def menu_control_compras():
    """Mostrar el menú de control de compras."""
    while True:
        print("\nMenú de control de compras:")
        print("1. Actualizar estado de compra")
        print("2. Ver historial de compras")
        print("3. Gestionar proveedores")
        print("4. Volver al menú principal")

        opcion = input("Ingrese la opción deseada: ")

        if opcion == "1":
            actualizar_estado_compra()
        elif opcion == "2":
            ver_historial_compras()
        elif opcion == "3":
            menu_gestion_proveedores()
        elif opcion == "4":
            break
        else:
            print("Opción inválida. Intente nuevamente.")
```

```
def ver_historial_compras():
    """Ver historial de compras."""
    db = connect_db()
    if not db:
        return

    sql = "SELECT * FROM historial_compras"

    try:
        cursor = db.cursor()
        cursor.execute(sql)
        compras = cursor.fetchall()

        print("\nHistorial de Compras:")
        for compra in compras:
            print(f"ID: {compra[0]}, Fecha: {compra[1]}, Producto: {compra[2]}, Cantidad:
{compra[3]}, Precio: {compra[4]}, Estado: {compra[5]}")
    except Error as e:
        print(f"Error al ver historial de compras: {e}")
    finally:
        if db:
            db.close()

def registrar_proveedor():
    """Registrar un nuevo proveedor."""
    db = connect_db()
    if not db:
        return

    nombre = input("Ingrese el nombre del proveedor: ")
    direccion = input("Ingrese la dirección del proveedor: ")
    telefono = input("Ingrese el teléfono del proveedor: ")
    email = input("Ingrese el correo electrónico del proveedor: ")
    cuit = input("Ingrese el CUIT del proveedor: ")

    sql = """
        INSERT INTO proveedores (nombre, direccion, telefono, email, cuit)
        VALUES (%s, %s, %s, %s, %s)
    """

    try:
        cursor = db.cursor()
        cursor.execute(sql, (nombre, direccion, telefono, email, cuit))
        db.commit()
        print("Proveedor registrado con éxito.")
    except Error as e:
        print(f"Error al registrar proveedor: {e}")
    finally:
        if db:
```

```
            db.close()

def menu_registro_ventas():
    """Mostrar el menú de registro de ventas."""
    while True:
        print("\nMenú de registro de ventas:")
        print("1. Registrar nueva venta")
        print("2. Actualizar estado de venta")
        print("3. Volver al menú principal")

        opcion = input("Ingrese la opción deseada: ")

        if opcion == "1":
            registrar_venta()
        elif opcion == "2":
            actualizar_estado_venta()
        elif opcion == "3":
            break
        else:
            print("Opción inválida. Intente nuevamente.")

def registrar_venta():
    """Registrar una nueva venta."""
    db = connect_db()
    if not db:
        return

    producto = input("Ingrese el nombre del producto: ")
    cantidad = int(input("Ingrese la cantidad vendida: "))
    precio = float(input("Ingrese el precio unitario: "))
    estado = "Pendiente"

    sql = """
        INSERT INTO venta (producto, cantidad, precio, estado)
        VALUES (%s, %s, %s, %s)
    """

    try:
        cursor = db.cursor()
        cursor.execute(sql, (producto, cantidad, precio, estado))
        db.commit()
        print("Venta registrada con éxito.")
    except Error as e:
        print(f"Error al registrar venta: {e}")
    finally:
        if db:
            db.close()

def actualizar_estado_venta():
```

```
    """Actualizar el estado de una venta."""
    id_venta = int(input("Ingrese el ID de la venta a actualizar: "))
    nuevo_estado = input("Ingrese el nuevo estado de la venta: ")

    actualizar_venta(id_venta, nuevo_estado)

def menu_analisis_datos():
    """Mostrar el menú de análisis de datos."""
    while True:
        print("\nMenú de análisis de datos:")
        print("1. Ver total de ventas")
        print("2. Ver ventas por período")
        print("3. Volver al menú principal")

        opcion = input("Ingrese la opción deseada: ")

        if opcion == "1":
            ver_total_ventas()
        elif opcion == "2":
            ver_ventas_por_periodo()
        elif opcion == "3":
            break
        else:
            print("Opción inválida. Intente nuevamente.")

def ver_total_ventas():
    """Ver las ventas totales."""
    db = connect_db()
    if not db:
        return

    sql = "SELECT SUM(cantidad*precio) FROM venta WHERE estado='Completada'"

    try:
        cursor = db.cursor()
        cursor.execute(sql)
        total_ventas = cursor.fetchone()[0]

        print(f"\nTotal de ventas: ${total_ventas}")
    except Error as e:
        print(f"Error al obtener el total de ventas: {e}")
    finally:
        if db:
            db.close()

def ver_ventas_por_periodo():
    """Ver ventas por período."""
    db = connect_db()
    if not db:
```

```
        return

    fecha_inicio = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
    fecha_fin = input("Ingrese la fecha de fin (YYYY-MM-DD): ")

    sql = """
        SELECT SUM(cantidad*precio)
        FROM venta
        WHERE estado='Completada' AND fecha BETWEEN %s AND %s
    """

    try:
        cursor = db.cursor()
        cursor.execute(sql, (fecha_inicio, fecha_fin))
        ventas_periodo = cursor.fetchone()[0]

        print(f"\nVentas entre {fecha_inicio} y {fecha_fin}: ${ventas_periodo}")
    except Error as e:
        print(f"Error al obtener las ventas por período: {e}")
    finally:
        if db:
            db.close()

def consultar_proveedores():
    """Consultar proveedores."""
    db = connect_db()
    if not db:
        return

    sql = "SELECT * FROM proveedores"

    try:
        cursor = db.cursor()
        cursor.execute(sql)
        proveedores = cursor.fetchall()

        print("\nListado de Proveedores:")
        for proveedor in proveedores:
            print(f"ID: {proveedor[0]}, Nombre: {proveedor[1]}, Dirección: {proveedor[2]},
Teléfono: {proveedor[3]}, Email: {proveedor[4]}, CUIT: {proveedor[5]}")
    except Error as e:
        print(f"Error al consultar proveedores: {e}")
    finally:
        if db:
            db.close()


if __name__ == "__main__":
    menu_principal()
```