Informe de Proyecto Final

Sistema de Gestión de Restaurante "Cangri & Sons"

Asignatura: Programación II

Autor: [Tu Nombre Completo]
Docente: Guido Mellado
Ayudantes: Fernando Valdés y Joaquín Cantero

21 de noviembre de 2025

Índice

1. Introducción

El presente informe detalla el desarrollo e implementación del "Sistema de Gestión para el Restaurante Cangri & Sons". El objetivo principal de este proyecto es digitalizar y optimizar los procesos operativos del restaurante, abarcando la gestión de clientes, control de inventario, administración del menú y procesamiento de pedidos.

Para lograr esto, se ha utilizado el lenguaje de programación **Python**, integrando tecnologías modernas como **SQLAlchemy** para el mapeo objeto-relacional (ORM) y **CustomTkinter** para el diseño de una interfaz gráfica moderna y amigable.

El sistema resuelve problemáticas críticas como:

- Inconsistencia en el stock de ingredientes.
- Dificultad para calcular totales y gestionar pedidos.
- Falta de persistencia de datos estructurada.

2. Arquitectura del Sistema

El proyecto sigue una arquitectura modular que separa la lógica de acceso a datos, la lógica de negocio y la interfaz de usuario.

2.1. Estructura de Archivos

La organización del proyecto se distribuye de la siguiente manera:

```
ORM_clientes/
|-- database.py         # Configuración de conexión y sesión
|-- models.py           # Definición de clases (Tablas)
|-- main.py             # Script de inicialización y sembrado de datos
|-- app.py              # Interfaz Gráfica (Frontend)
|-- crud/               # Lógica de Negocio (Backend)
|   |-- cliente_crud.py
|   |-- menu_crud.py
|   |-- pedido_crud.py
|   |-- ingrediente_crud.py
```

2.2. Tecnologías Utilizadas

- **Python 3.12+**: Lenguaje base.
- **SQLAlchemy**: ORM para interactuar con SQLite sin escribir SQL crudo.
- **CustomTkinter**: Biblioteca de UI basada en Tkinter.
- **Matplotlib**: Para la generación de gráficos estadísticos.

3. Diseño de Base de Datos (ORM)

Se implementó un modelo relacional utilizando clases de Python que heredan de `DeclarativeBase`.

3.1. Entidades Principales

1. **Cliente**: Almacena información personal (Nombre, Email). Relación 1 a N con Pedidos.
2. **Menu**: Representa los platos disponibles. Relación N a M con Ingredientes.
3. **Ingrediente**: Controla el stock y unidad de medida.
4. **Pedido**: Registra la transacción, fecha y total.

3.2. Relaciones Complejas

Uno de los desafíos fue la relación Muchos a Muchos entre *Menu* e *Ingrediente*, ya que un plato tiene muchos ingredientes, pero cada ingrediente requiere una cantidad específica para ese plato. Esto se resolvió con una tabla de asociación:

```
menu_ingrediente = Table(
    'menu_ingrediente', Base.metadata,
    Column('menu_id', Integer, ForeignKey('menus.id')),
    Column('ingrediente_id', Integer, ForeignKey('ingredientes.id')),
    Column('cantidad_requerida', Float, nullable=False)
)
```

Listing 1: Definición de Tabla Intermedia

4. Lógica de Negocio y Validaciones

4.1. Control de Stock Automático

La característica más robusta del sistema es el descuento automático de inventario al realizar un pedido. El algoritmo sigue estos pasos: 1. Recibe el pedido. 2. Consulta la receta de cada plato. 3. Calcula el total de ingredientes necesarios (Receta $\times$ Cantidad Pedida). 4. Verifica si existe stock suficiente. 5. Si hay stock, descuenta y confirma. Si falta aunque sea un ingrediente, realiza un **Rollback** y cancela la operación.

```
# Fragmento de lógica de validación
if ing_obj.stock_actual < total_a_descontar:
    raise ValueError(
        f"Stock insuficiente de '{ing_obj.nombre}'."
    )
# Descontar si todo está correcto
ing_obj.stock_actual -= total_a_descontar
```

Listing 2: Lógica de Validación de Stock en pedido_crud.py

4.2. Carga Masiva de Datos

Se implementó una funcionalidad para cargar ingredientes desde archivos CSV y un sistema de "sembrado.ªutomático que carga el menú completo (Chorrillanas, Empanadas, etc.) la primera vez que se ejecuta el programa, evitando una base de datos vacía.

5. Interfaz Gráfica de Usuario

La interfaz se desarrolló utilizando `CustomTkinter` para ofrecer una experiencia moderna (Modo Oscuro/Claro). Se organiza en pestañas funcionales:

- **Pestaña Clientes:** Formulario de registro y tabla de visualización.

- **Pestaña Menú:** Visualización de precios y platos cargados desde el catálogo.
- **Pestaña Ingredientes:** Monitor de stock en tiempo real y botón para carga de CSV.
- **Pestaña Pedidos:** Sistema de Çarrito de compras"que permite agregar múltiples platos antes de confirmar la venta.
- **Pestaña Estadísticas:** Integración con *Matplotlib* para visualizar los platos más vendidos.

6. Conclusiones

El desarrollo de este sistema permitió consolidar conocimientos clave en la ingeniería de software:

1. **Uso de ORM:** Se evidenció cómo SQLAlchemy facilita la manipulación de datos y protege contra inyecciones SQL básicas, además de gestionar relaciones complejas de manera transparente.
2. **Integridad de Datos:** La implementación de transacciones atómicas (commit/rollback) asegura que el inventario nunca quede en estados inconsistentes.
3. **Escalabilidad:** Gracias a la estructura modular (separando Modelos, Vistas y Controladores/CRUD), el sistema es fácilmente escalable para agregar nuevas funcionalidades como facturación electrónica o gestión de mesas.

El sistema cumple con todos los requisitos de la evaluación, proporcionando una herramienta robusta para la administración de Çangri & Sons".