UNIVERSIDAD CATÓLICA DE TEMUCO
Facultad de Ingeniería
Ingeniería Civil en Informática

Evaluación 3

Programación II

Sistema de Gestión de Clientes y Pedidos en un Restaurante

Profesores: Guido Mellado

Ayudantes: Fernando Valdés y Joaquín Cantero

Información General

Modalidad: Proyecto Grupal o Individual (Mínimo 2 - Máximo 5)
Duración: 2 semanas aproximadamente
Objetivo: Mejorar el sistema de restaurante utilizando programación orientada a objetos, manejo de bases de datos con ORM y diseño de interfaces gráficas en Python.

Herramientas a Utilizar

- **Lenguaje:** Python 3.12 o superior
- **Bibliotecas:**
  - Interfaz gráfica: `customtkinter`, `tkinter`
  - ORM y Base de Datos: `SQLAlchemy`
  - Gráficos: `matplotlib`

Objetivos del Proyecto

1. Aprender a programar usando programación orientada a objetos. Crear clases y métodos para manejar las entidades del proyecto: Clientes, Pedidos, Ingredientes y Menús
2. Crear una base de datos y sincronizarla con el ORM.
3. Hacer un CRUD (Crear, Leer, Actualizar, Eliminar) para cada entidad, usando el concepto de ORM con SQLAlchemy.
4. Diseñar una interfaz gráfica clara y fácil de usar con customtkinter para que los usuarios puedan interactuar con el sistema sin dificultad.
5. Usar expresiones lambda, `map`, `filter` y `reduce`.
6. Crear gráficos sencillos con estadísticas basadas en los datos del sistema.
7. Simular un proceso de compra para generar boletas y registrar pedidos.

Organización del Proyecto

```
ORM_clientes/
 app.py                # Archivo principal con la interfaz gráfica y pestañas
 database.py           # Configuración de la base de datos y ORM
 models.py             # Definición de los modelos ORM
 graficos.py           # Funciones para generación de gráficos
 main.py               # Script para inicializar las tablas
 crud/
    cliente_crud.py
    pedido_crud.py
    ingrediente_crud.py
    menu_crud.py
```

Módulos a Implementar

Módulo	Requisitos Mínimos
Base de Datos	Crear una base de datos para la gestión del restaurante. El acceso a la base de datos se realizará únicamente mediante el ORM. ▪ Validar la conexión al motor de base de datos antes de realizar operaciones. ▪ Evitar la inserción de registros duplicados (claves primarias o campos únicos). ▪ Implementar `rollback()` automático en caso de error en una transacción.
Gestión de Ingredientes	▪ Carga de datos desde un archivo CSV, guardados en la base de datos. Controlar errores en la carga (columnas faltantes o formato incorrecto). ▪ Operaciones CRUD: Crear, Leer, Actualizar y Eliminar. ▪ Verificar que el nombre del ingrediente no esté vacío ni duplicado. ▪ Validar que el stock sea un número positivo y mayor que cero. ▪ Al cargar stock desde CSV, debe actualizar la tabla de ingredientes y stock mediante el ORM. ▪ Uso de expresiones `lambda`, `map`, `filter`, `reduce`.

Gestión de Menús	▪ Visualización y edición de los menús disponibles. ▪ Creación de nuevos menús usando ingredientes existentes en la base de datos. ▪ Cada menú debe tener nombre, descripción y lista de ingredientes con cantidades requeridas. ▪ Evitar ingredientes duplicados o con cantidades inválidas (cero o negativas). ▪ Validar que los ingredientes seleccionados existan y tengan suficiente stock. ▪ Uso de operaciones `lambda`, `map`, `filter`, `reduce`. ▪ *Ejemplo:* Un "Completo" incluiría 1 vienesa, 1 pan, $\frac{1}{2}$ palta y $\frac{1}{2}$ tomate.
Gestión de Clientes	▪ Visualizar todos los clientes registrados. ▪ Asociar clientes con pedidos y compras en la base de datos (ORM). ▪ Agregar pestaña de gestión de clientes. ▪ Operaciones CRUD: Crear, Leer, Actualizar y Eliminar. ▪ Validar formato y unicidad del correo electrónico antes de guardar. ▪ Verificar que nombre y correo no estén vacíos. ▪ Impedir eliminar clientes que tengan pedidos asociados en la base de datos. ▪ Uso de `lambda`, `map`, `filter`, `reduce`.

Panel de Compra	▪ Permitir seleccionar un cliente registrado. ▪ Agregar productos del menú a una lista de compra. ▪ Generar boleta detallada y guardar el pedido en la base de datos. ▪ Validar que se haya seleccionado un cliente válido antes de procesar la compra. ▪ Verificar que se hayan agregado productos al pedido antes de generar la boleta. ▪ Controlar errores al guardar el pedido (por ejemplo, pérdida de conexión o datos incompletos). ▪ Uso de `lambda`, `map`, `filter`, `reduce`.
Gestión de Pedidos	▪ Mostrar todos los pedidos registrados y permitir su organización. ▪ CRUD: Visualizar pedidos por cliente específico. ▪ Cada pedido debe incluir descripción, total, fecha y cantidad de menús comprados. ▪ Asegurar que cada pedido tenga cliente, fecha y total válidos. ▪ Validar que los menús asociados existan y estén disponibles. ▪ Manejar errores al intentar acceder o eliminar pedidos inexistentes.

Gráficos Estadísticos	▪ Permitir seleccionar tipo de gráfico mediante un menú desplegable. ▪ Tipos: • Ventas por fecha (diarias, semanales, mensuales, anuales). • Distribución de menús más comprados. • Uso de ingredientes en todos los pedidos realizados. ▪ Verificar que existan datos suficientes antes de generar gráficos. ▪ Manejar errores de formato en los datos (fechas, totales, etc.). ▪ Mostrar un mensaje de "No hay datos disponibles" cuando no existan registros para graficar.

Nota: Estos son los elementos mínimos; si en el desarrollo puedes considerar agregar otra funcionalidad que sea necesaria para su correcto funcionamiento.

Uso de Patrón de diseño Factory Method (0.5 puntos extras)

- Los estudiantes pueden integrar el patrón de diseño Factory Method en su proyecto para obtener un punto adicional en la nota final.
- Se debe incluir una explicación clara del uso del patrón en la presentación final del proyecto, mostrando cómo ayuda a mejorar la estructura y flexibilidad del código.

Entregables

Código

- Diagrama de clases y MER en PDF
- Presentación del proyecto en latex igual que la entrega anterior
- Enlace al repositorio de GitHub

Presentación

- Presentación de un PPT en grupos de 2-4 personas para el miercoles 19/11/2025 (seccion 1) y 21/11/2025 (seccion 2). se subira un excel para agendar la presentacion
- debe contener título del problema, nombre de los integrantes, fecha y logo del departamento
- Presentación de las mejoras del proyecto con respecto al mostrado en la ev2
- El diagrama de clases y MER con la solución propuesta del problema.
- Presentación de la solución en código explicando el flujo principal de la solución.
- manejo de ORM y funciones Lambda
- La presentación de la interfaz visual del problema.
- La demostración de la funcionalidad total del programa.
- Duración: 10 minutos + 5 minutos de preguntas.

NOTA: cualquiera de los items esperados, o reglas planteadas para la presentación, pueden tener que ser cambiados o ser mostrados al ayudante de forma diferida para su puntuación el día de presentación.

Rúbrica de Evaluación

Criterio	Excelente (10)	Bueno (8)	Satisfactorio (5)	Insuficiente (0)
ORM y base de datos (15 %)	Define modelos y relaciones correctas. Uso de ORM en todo el proyecto.	Modelos y relaciones mayormente correctos; usa el ORM en la mayoría de las consultas	Uso superficial del ORM; relaciones	No usa ORM o lo usa solo como mapeo sin relaciones.
Map, filter, reduce (10 %)	Implementa correctamente las operaciones MAP, filter, reduce para todas las entidades. Sin errores	Implementa correctamente las operaciones MAP, filter, reduce con algunos errores.	Implementa correctamente las operaciones MAP, filter, con errores evidentes o algunas partes faltantes.	No presenta lo requerido de forma mínima
Funcionalidad del CRUD (15 %)	Implementa correctamente las operaciones CRUD para todas las entidades. Sin errores	Implementa las operaciones CRUD con pequeños errores en algunas entidades.	CRUD parcialmente implementado, con errores evidentes o algunas partes faltantes.	CRUD incompleto y con errores graves que afectan el funcionamiento del sistema.
Interfaz Gráfica (5 %)	GUI es clara, intuitiva y está bien diseñada. Cumple con todos los requisitos del proyecto.	GUI está bien diseñada pero presenta pequeñas dificultades de navegación o uso.	GUI es funcional, pero difícil de navegar o con diseño poco intuitivo	GUI incompleta o difícil de usar, afectando la experiencia del usuario.
Uso de LaTeX (10 %)	Documento perfectamente formateado en LaTeX: estructura clara, tablas, figuras y secciones bien organizadas.	Documento en LaTeX con pequeños errores de formato o estilo.	Uso parcial de LaTeX o con errores evidentes de compilación o formato.	Documento no presentado en LaTeX o con formato ilegible.

Uso de GitHub (5 %)	Repositorio en GitHub completo, bien estructurado y actualizado. Uso adecuado de ramas, *Pull Requests* y *merge*. Commits descriptivos siguiendo buenas prácticas como *Conventional Commits*.	Repositorio funcional con buena organización general, aunque algunos commits no son claros o faltan PR documentados.	Repositorio incompleto o con historial de commits desordenado. No se evidencia un flujo de trabajo colaborativo claro.	Sin repositorio funcional o con escasa evidencia de control de versiones o colaboración.
Gráficos Estadísticos (10 %)	Los gráficos están bien implementados, son claros y muestran correctamente las estadísticas solicitadas.	Los gráficos están bien implementados, pero algunos carecen de claridad o de etiquetas adecuadas.	Los gráficos están implementados, pero muestran información incompleta o confusa.	Gráficos ausentes o inútilemtes implementados, sin proporcionar información clara.
Validación de Errores (15 %)	El sistema valida correctamente todas las entradas del usuario y maneja los errores mediante excepciones controladas, mostrando mensajes claros y evitando interrupciones del programa.	El sistema valida la mayoría de las entradas y maneja errores comunes, aunque algunos casos límite pueden generar comportamientos inesperados.	La validación de datos es parcial o inconsistente; algunos errores no se manejan correctamente y pueden provocar cierres inesperados.	No existe validación de errores; el programa falla ante entradas incorrectas o situaciones no previstas.
Proceso de Compra (15 %)	Proceso de Compra Simulado Proceso de compra completo y fácil de seguir. La generación de boleta es clara y se guarda en la base de datos sin errores.	Proceso de compra implementado, pero con pequeñas dificultades o errores menores.	Proceso de compra incompleto o difícil de seguir. Generación de boleta con errores.	Proceso de compra ausente o con errores graves que impiden su uso