INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE AGOSTO-DICIEMBRE 2025

CARRERA
Ingeniería en Sistemas Computacionales

MATERIA Y SERIE
Patrones de diseño.

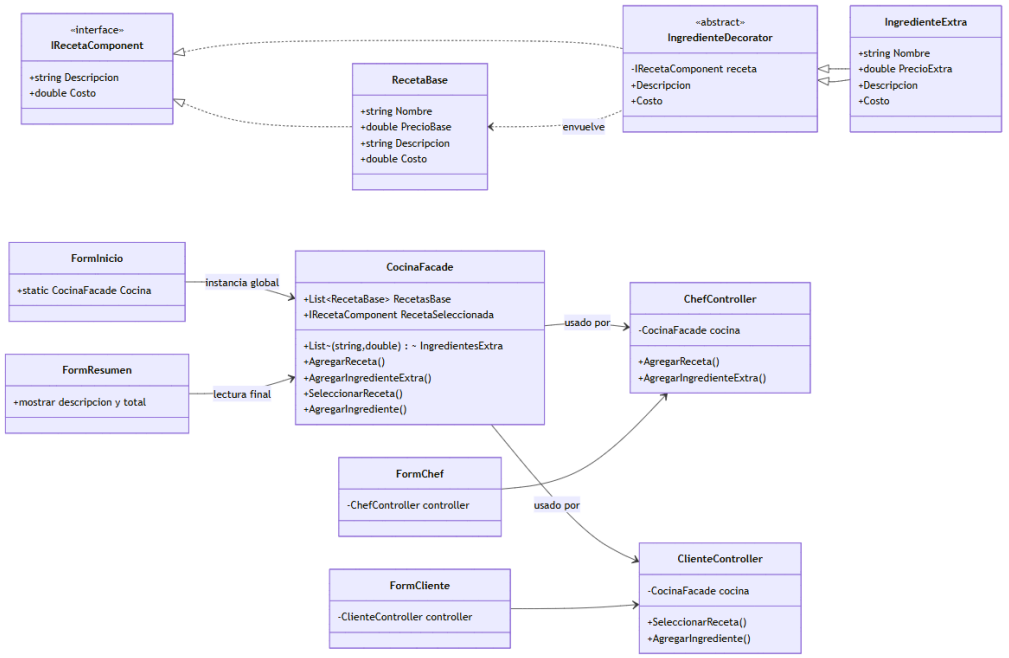
TÍTULO
Examen Final

UNIDAD A EVALUAR
Unidad 4 y 5

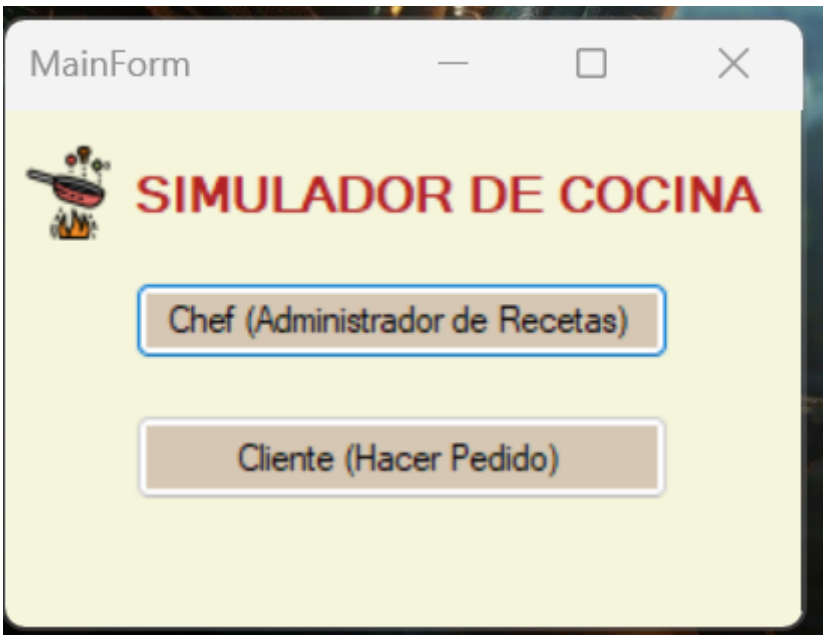
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO
Perez Villa Belén, 21212579

NOMBRE DEL MAESTRO
Maribel Guerrero Luis

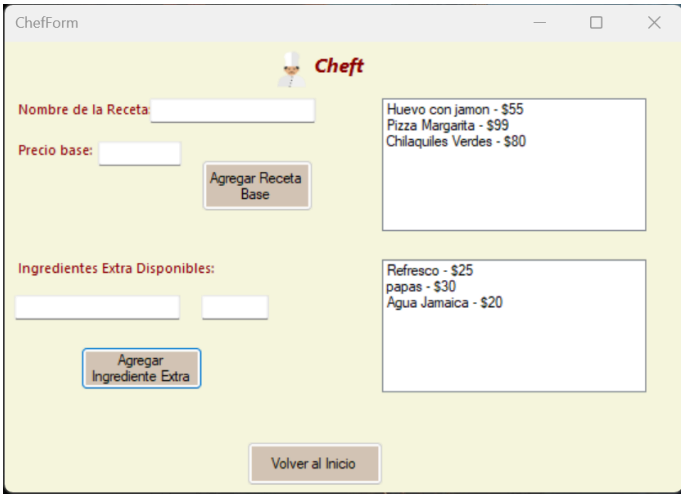
Diagrama UMl



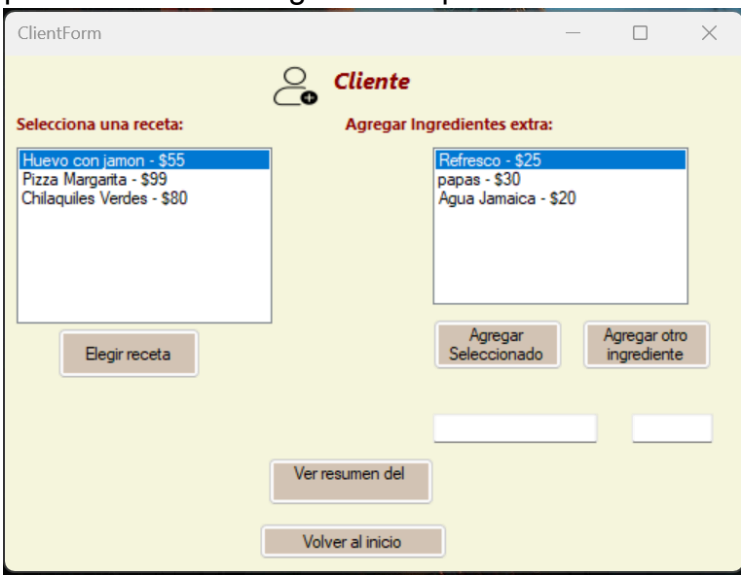
Captura de pantalla
Pantalla de inicio que puedes escoger 2 tipos de usuarios que puedes ser el Chef o el cliente.



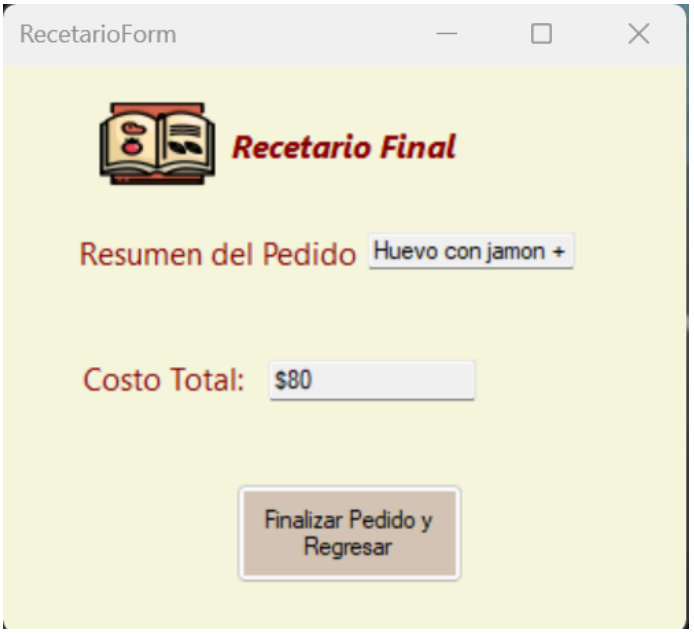
Pantalla del chef que puede agregar recetas base e ingredientes extras con su respectivos precios.



Pantalla del cliente que puede elegir las recetas y los ingredientes extra que quiera elegir y puede añadir otro ingrediente que no esté en la selección



Al final se muestra el resumen del pedido y con el costo total.



Conclusión

Como proyecto final, reutilizamos el examen de la unidad 3 del simulador de cocina lo que hicimos es mejorar el proyecto de una manera visual y con 2 diferentes tipos de usuarios “Cliente y Chef” donde el cliente puede seleccionar una receta base y personalizar agregandos los ingredientes adicionales de una forma más dinámica. El usuario de chef es encargado de registrar las nuevas recetas e ingredientes.

Reutilizamos los patrones utilizados antes en el examen de la unidad 3 que son el decorator y el fachada y agregamos otros 3 patrones que son el Singleton, el MVC Ligero y el command, cómo utilizamos estos patrones en el proyecto:

En el decorator utilizamos para la personalización dinámica de recetas y se diseñó una interfaz llamada IRecetaComponent, la clase base RecetaBase y un decorador abstracto IngredienteDecorator.

El Facade implementó la clase CocinaFacade que unifica y simplifica todo el proceso.

Uno de los nuevos patrones incorporados en este proyecto fue el singleton que fue implementado de la siguiente manera que en el MainForm contiene una instancia estática: public static CocinaFacade Cocina = new CocinaFacade();, que toda la aplicación sólo utiliza una instancia de la cocina que el chef y el cliente trabajan sobre los mismos datos y el fachada actúe como centro de operaciones del sistema.

Como arquitectura del proyecto se utilizó el MVC ligero que nos ayuda a organizar de una manera más simplificado:

- Forms = Vista
- Controllers = Controlador
- Clases de modelo = modelo
- Facade = servicio central.

Este patrón nos ayuda a tener una separación clara entre la interfaz y lógica, controladores independientes por usuario y un mejor entendimiento, reutilización y escalabilidad del código.

Como patrón de comportamiento utilizamos el Command, que cada accion del usuario en los forms (clics, selecciones, etc,) dispara eventos que pasan por el controlador, donde cada botón ejecuta una acción encapsulada.

- Mantiene la lógica fuera de la interfaz
- Controla el flujo entre vista, controlador y modelo
- permite añadir nuevos comportamientos fácilmente.