Gestor de Usuarios — MVC con Patrones de Diseño

Universidad de Cordoba

Asignatura: Programacion III

Proyecto: Gestor de Usuarios — MVC con Patrones de Diseño

Integrantes: Saúl Pérez

Docente: Alberto Paternina

Fecha: [23/10/2025]

1. Descripción del Proyecto

El presente proyecto implementa un Gestor de Usuarios desarrollado en Java, estructurado bajo el patrón arquitectónico Modelo–Vista–Controlador (MVC) y complementado con tres patrones de diseño provenientes del catálogo de Refactoring.Guru.

El sistema permite registrar, visualizar y eliminar usuarios según su tipo (Administrador, Cliente o Invitado). Incluye validaciones para evitar registros incompletos o con correos electrónicos inválidos, asegurando un flujo de uso controlado y coherente.

2. Patrones de Diseño Utilizados

2.1 Patrón Creacional — Factory Method (Modelo)

Ubicación: model/UsuarioFactory.java

Se utiliza para crear distintos tipos de usuarios sin exponer la lógica de instanciación al controlador. Esto permite extender el sistema fácilmente si se agregan nuevos tipos de usuario.

Ventaja: separación de la creación de objetos de su uso.

2.2 Patrón Estructural — Decorator (Vista)

Ubicación: view/VistaDecorator.java y view/VistaDetallada.java

Permite ampliar la funcionalidad de la vista sin modificar su estructura base, facilitando la personalización visual o funcional.

Ventaja: promueve la reutilización de componentes gráficos.

2.3 Patrón de Comportamiento — Command (Controlador)

Ubicación: controller/AgregarUsuarioCommand.java, EliminarUsuarioCommand.java

Encapsula cada acción (agregar, eliminar) en un objeto comando, desacoplando la lógica de eventos del controlador principal.

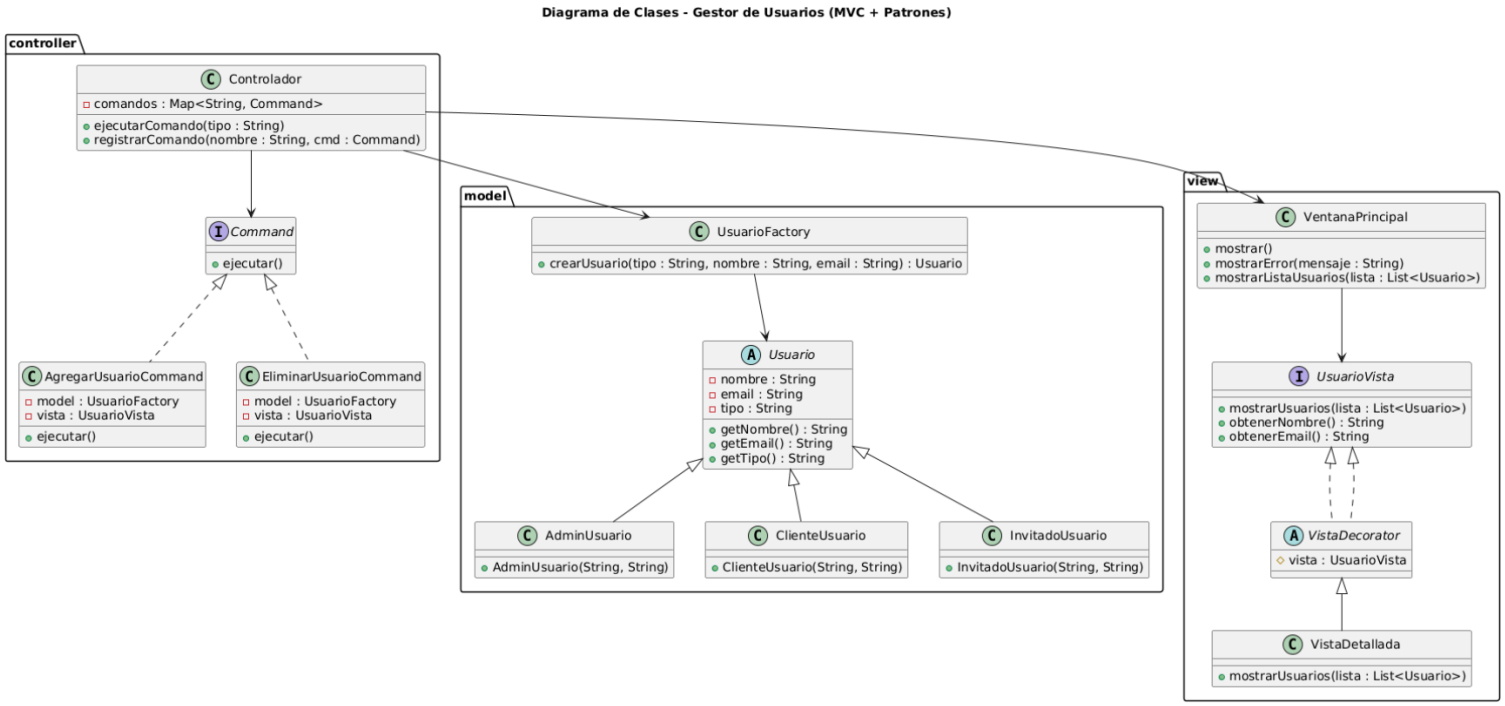
Ventaja: permite agregar o modificar acciones sin alterar el resto del sistema.

3. Integración del Patrón MVC

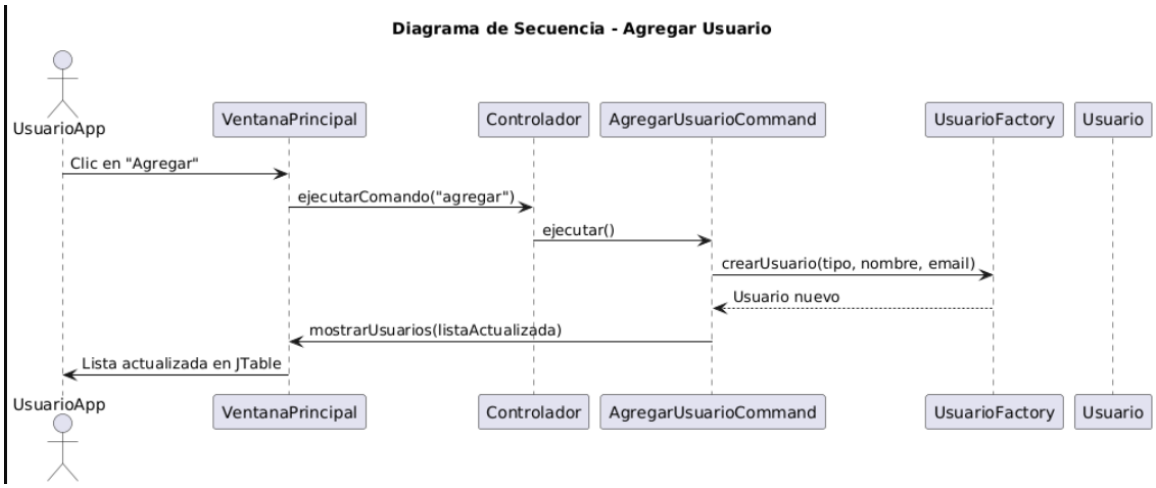
Componente	Rol	Ejemplo
Modelo (Model)	Lógica de negocio y creación de usuarios	UsuarioFactory, Usuario, AdminUsuario
Vista (View)	Interfaz gráfica (Swing)	VentanaPrincipal, VistaDetallada
Controlador (Controller)	Manejo de eventos y comandos	UsuarioControlador, AgregarUsuarioCommand

4. Diagramas UML

4.1 Diagrama de Clases —

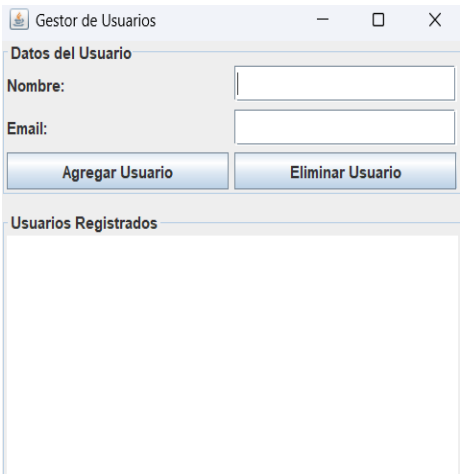


4.2 Diagrama de Secuencia —

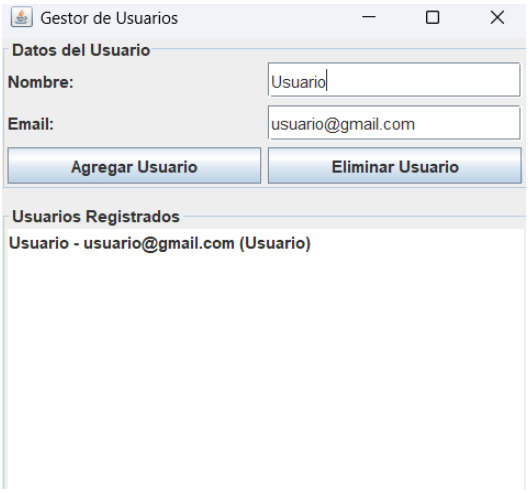


5. Capturas de Ejecución

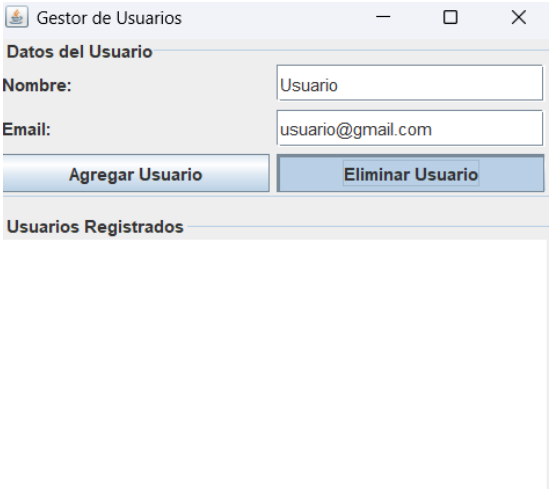
- Interfaz principal.
- Registro exitoso de usuario.
- Validación de email incorrecto.
- Eliminación de usuario.



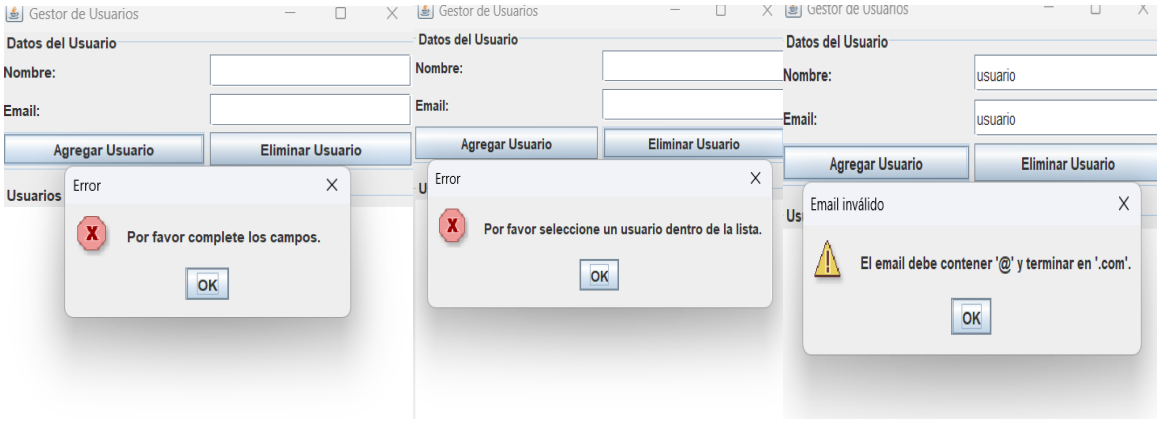
Interfaz: Contiene la base del proyecto y es bastante amigable con el usuario demostrándole la función de cada botón sin tanta complicación.



Agregar usuario: una vez agregado un usuario se muestra en la tabla de usuarios registrados con sus respectivos datos.



Eliminar Usuario: Una vez presionado este botón el usuario es inmediatamente removido del sistema.



Validaciones: Aquí salen los distintos mensajes si el usuario no completa una validación requerida.

6. Reflexión y Conclusiones

Durante el desarrollo del proyecto se logró aplicar de manera conjunta los patrones de diseño Factory Method, Decorator y Command, integrándolos en una arquitectura MVC coherente.

Esta práctica permitió comprender cómo los patrones pueden colaborar para mejorar la organización, flexibilidad y mantenibilidad del código.

Ventajas observadas:
- Separación clara de responsabilidades.
- Código modular y extensible.
- Facilidad para añadir nuevas funciones o tipos de usuario.

Limitaciones:
- Falta persistencia de datos (usuarios no se guardan al cerrar la aplicación).
- El diseño visual es básico y podría mejorarse.

Posibles mejoras:
- Integrar una base de datos (SQLite o MySQL).
- Añadir sistema de login.
- Incorporar filtros y búsquedas avanzadas.

7. Referencias

- Refactoring.Guru — Catálogo de Patrones de Diseño
- Oracle Docs — Java Swing Documentation