Universidad Tecnológica de Panamá

Facultad de Ingeniería en Sistemas Computacionales

Licenciatura en Ciberseguridad

Estudiantes:

Cristian Jiménez 8-1011-665

Lía Reyes 8-1035-1875

Cristian Guevara 3-760-2077

Jassier Hernandez 8-1046-17

Documento de Diseño de Interfaz y Validaciones - Avance #3

Sistema de Gestión de Alquiler de Propiedades

Profesor:

Víctor Sarmiento

Fecha:

Viernes 12 de Diciembre de 2025

Introducción

El presente documento describe el **diseño de la interfaz de usuario**, las **validaciones implementadas** para cada entrada de datos y las **actualizaciones realizadas** al proyecto "Sistema de Gestión de Alquiler de Propiedades", con base en la retroalimentación de los Avances #1 y #2, y el código final del proyecto (Avance #3).

El sistema fue desarrollado en Java aplicando **Programación Orientada a Objetos (POO)**, siguiendo principios de:

- Encapsulamiento y visibilidad adecuada de atributos.
- Herencia mediante una clase abstracta Usuario y sus subclases (Propietario, Inquilino, Tecnico, Administrador).
- Composición entre clases de dominio (por ejemplo, Propiedad contiene una Direccion, un Propietario y una lista de ServicioIncluido).
- Manejo de errores mediante una excepción personalizada ValidacionException y validaciones centralizadas en la clase de utilidades IOUtils.

Este documento servirá como **guía técnica** para la presentación final del sistema y como evidencia de la aplicación de las recomendaciones del profesor.

1. Diseño de Interfaz de Usuario (UI/UX)

1.1 Descripción general de la interacción

El sistema ofrece dos modalidades principales de interacción:

1. **Interfaz gráfica** mediante cuadros de diálogo con JOptionPane, para:
    - Menú principal.
    - Formularios de registro (inquilinos, propiedades, contratos, pagos, solicitudes de mantenimiento).
    - Mensajes de error, confirmación y reportes.
2. **Interfaz en modo consola**, utilizada principalmente para:
    - Reemplazar a JOptionPane en caso de que esta llegase a fallar.
    - Mostrar listados de objetos (propiedades, inquilinos, contratos, pagos, tickets).
    - Ver reportes generados por GestorReportes.
    - Ver mensajes de estado durante pruebas.

El usuario navega a través de un **menú principal**, implementado en la clase Main, que invoca métodos estáticos en las clases de dominio (Inquilino, Propiedad, ContratoAlquiler, Pago, SolicitudMantenimiento) para ejecutar cada opción.

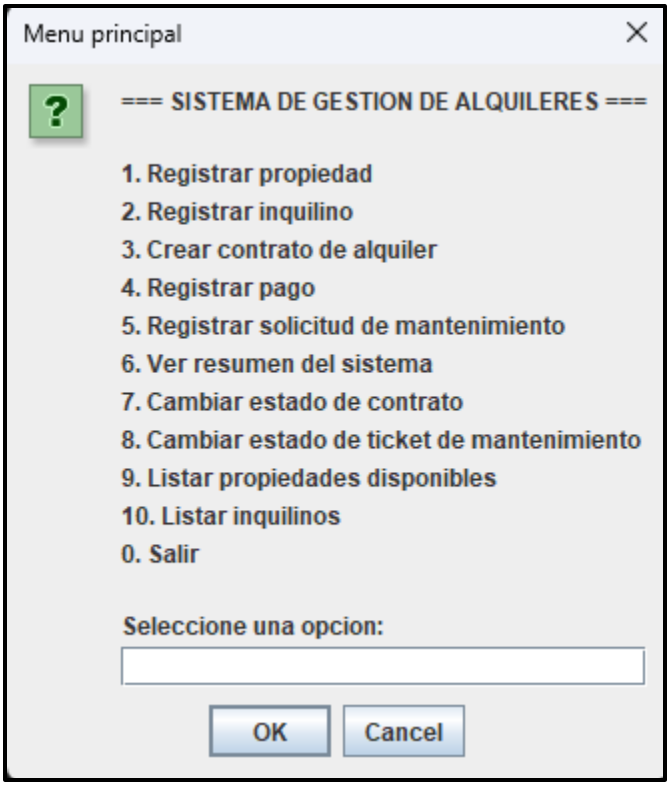
1.2 Prototipos de pantallas con JOptionPane

A continuación, se describen los prototipos de las pantallas más importantes

Menú principal – JOptionPane

Esta ventana muestra las opciones principales del sistema, por ejemplo:

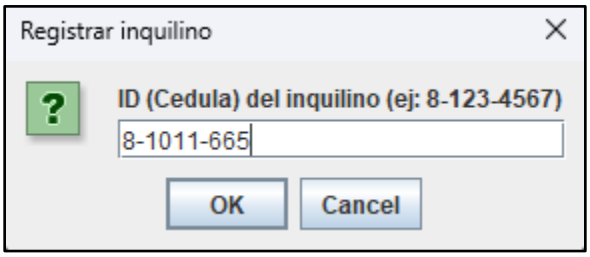
1. Registrar propiedad
2. Registrar Inquilino
3. Crear contrato de alquiler
4. Registrar pago
5. Registrar solicitud de mantenimiento
6. Ver Resumen del sistema
7. Cambiar estado de contrato
8. Cambiar estado de solicitud de mantenimiento
9. Listar propiedades disponibles
10. Listar Inquilinos
0. Salir

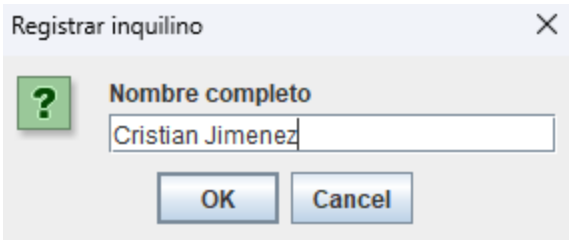


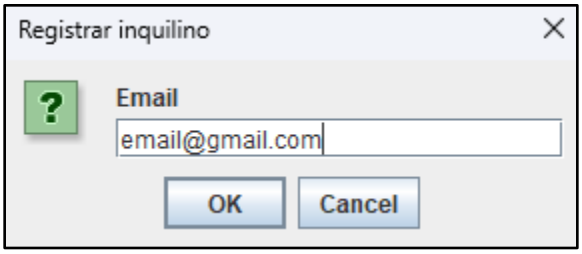
El usuario ingresa un número y el sistema llama al flujo correspondiente.

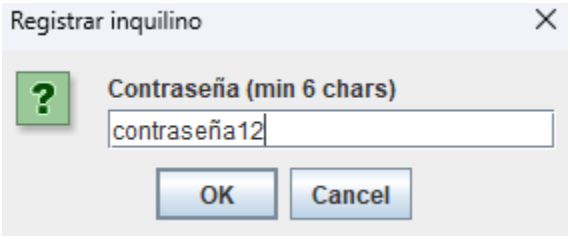
El menú principal permitirá al usuario seleccionar entre las opciones básicas del sistema, como registrar inquilinos, registrar propiedades, crear contratos de alquiler, listar datos y salir de la aplicación.

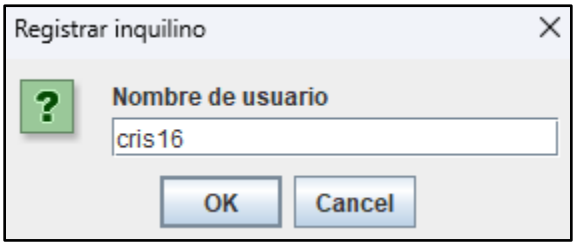
- **FORMULARIO DE REGISTRO DE INQUILINO – JOptionPane**

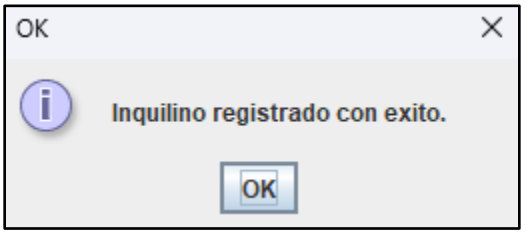








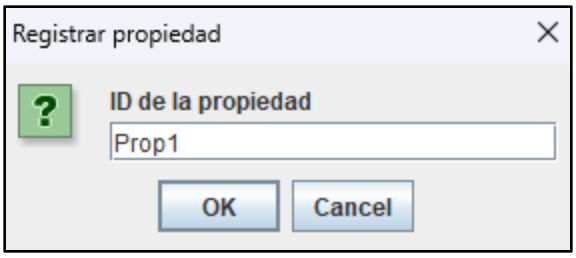


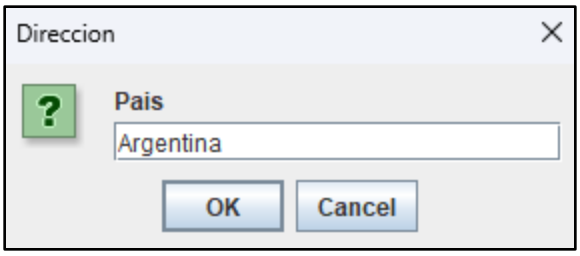


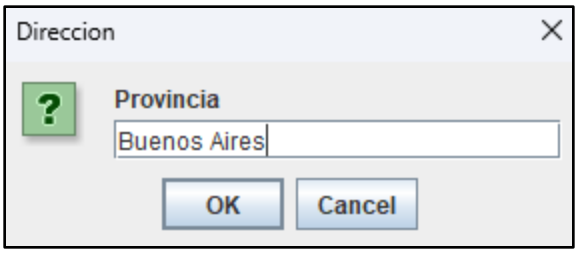
El formulario de registro de inquilino contendrá campos para nombre completo, cédula, teléfono y correo electrónico, así como los botones correspondientes para confirmar o cancelar el registro.

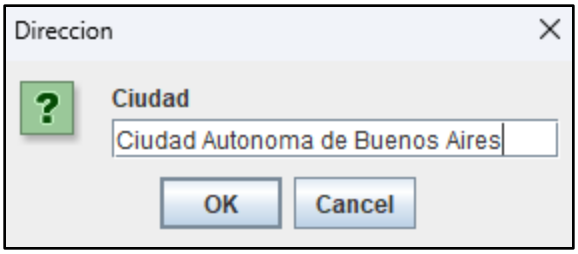
- **FORMULARIO DE REGISTRO DE PROPIEDAD Y PROPIETARIO– JOptionPane**

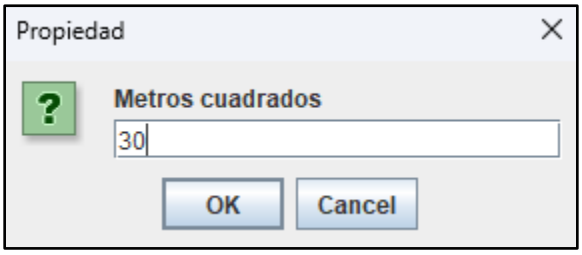
  - **Formulario de Registro de Propiedad:**

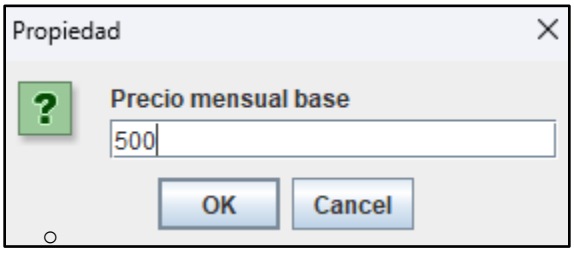




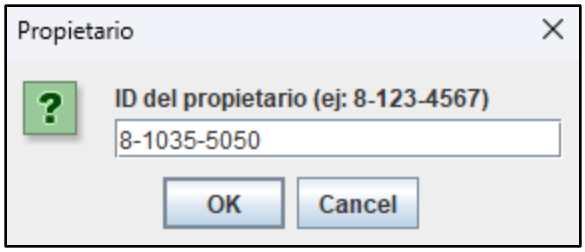


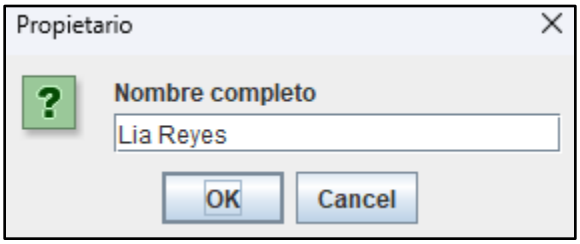


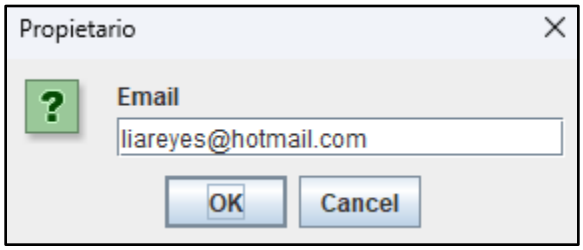


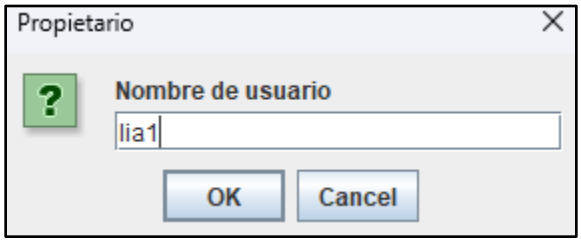


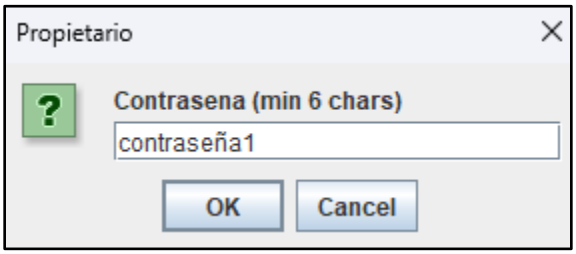
- **Formulario de Registro de Propietario:**

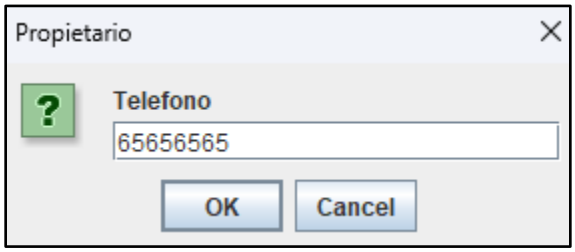




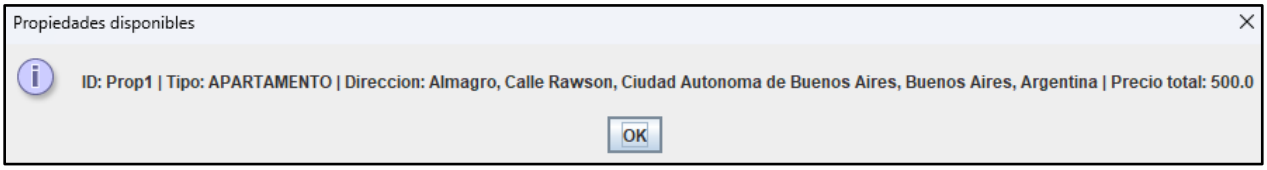






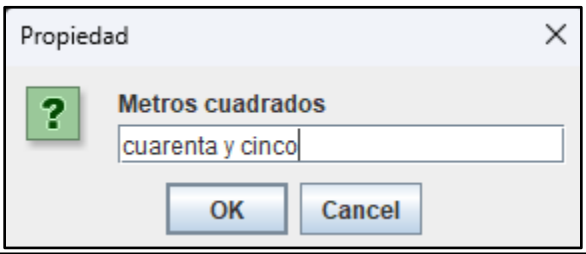


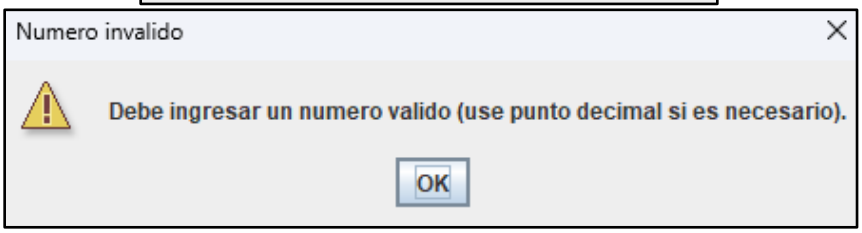
El formulario de registro de propiedad incluirá campos para dirección, tipo de propiedad (casa, apartamento, local, etc.), precio de alquiler mensual y la creación de un nuevo propietario. Al finalizar el registro de la nueva propiedad, la misma se marca en estado “disponible”:

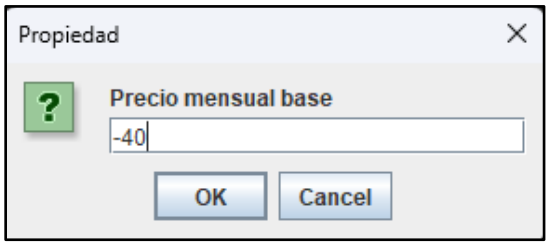


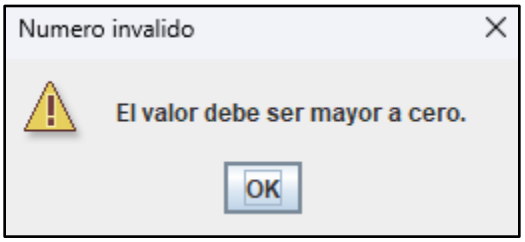
- **MENSAJES DE ERROR / VALIDACIÓN – JOptionPane**
    - **El campo de tipo de propiedad solo acepta una de las 4 opciones brindadas:**

    - Estos campos solo esperan de entrada números enteros, no cadena de textos:





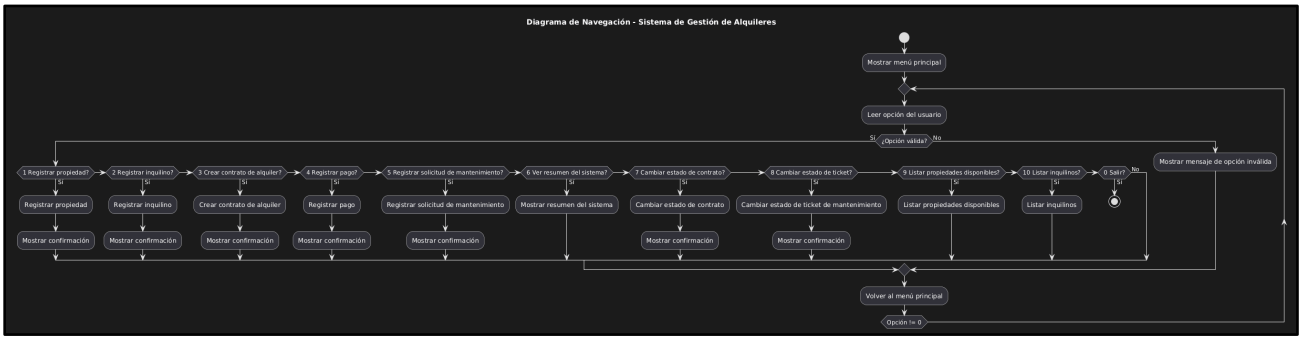




Los mensajes de error mostrarán al usuario cuándo una validación ha fallado, indicando de forma clara qué dato es inválido y cómo debe corregirse (por ejemplo, “El teléfono debe contener solo números” o “El precio mensual debe ser mayor a 0”).

1.3 Flujo de navegación entre opciones

El flujo de navegación describe cómo el usuario se desplaza entre las diferentes opciones del sistema, desde el inicio hasta el cierre de la aplicación.



De forma general, el flujo sigue la siguiente estructura:

1. Inicio → Mostrar menú principal.
2. El usuario selecciona una opción del menú.
3. Según la opción, se muestra el formulario o submenú correspondiente (registro de inquilino, registro de propiedad, creación de contrato, listados, etc.).
4. Se validan las entradas de datos y se muestran mensajes de éxito o error.
5. Tras completar la operación, se regresa al menú principal o se permite salir del sistema.

2. Validaciones por cada entrada de usuario

2.1 Resumen de validaciones

En esta sección se describen las validaciones que se aplican a cada entrada de usuario en el sistema. Para cada campo se especifica el tipo de dato esperado, los rangos o formatos válidos, los mensajes de error personalizados y las excepciones que pueden generarse y cómo serán manejadas.

Las validaciones se centralizan en:

- La clase IOUtils (métodos askNonEmpty, askSoloLetras, askSoloDigitos, askEmail, askDouble, etc.).
- Reglas de negocio dentro de las clases de dominio (Direccion, Propiedad, Usuario, ContratoAlquiler, Pago, SolicitudMantenimiento).
- Excepción personalizada ValidacionException que se lanza cuando un campo obligatorio no cumple las reglas establecidas.

2.2 Menú principal

Entrada: opción del menú principal
Tipo de dato esperado: int (o convertible a entero)
Rango válido: depende de la cantidad de opciones (por ejemplo, 1–9)

La siguiente tabla resume las validaciones del menú principal del sistema:

Entrada	Tipo de dato esperado	Rangos o formatos válidos	Mensaje de error personalizado	Excepciones manejadas
Opción del menú principal	int	Valores entre 0 y 11	“Seleccione una opción válida.”	NumberFormatException, ValidacionException

2.3 Registro de inquilino

Validaciones aplicadas a los campos del formulario de registro de inquilino:

Campo	Tipo de dato esperado	Rangos o formatos válidos	Mensaje de error personalizado	Excepciones manejadas
Nombre completo	String	No vacío, solo letras y espacios	"El nombre no puede estar vacío ni contener caracteres inválidos."	ValidacionException
Cédula	String	No vacío. Formato sugerido: 0-0000-0000	"Formato de cédula inválido. Ejemplo: 8-1234-5678."	ValidacionException
Teléfono	int	Solo números, longitud mínima 8 dígitos	"El teléfono debe contener solo números y tener al menos 8 dígitos."	NumberFormatException, ValidacionException
Correo electrónico	String	Debe contener "@" y dominio	"Correo electrónico inválido. Verifique el formato."	ValidacionException
Usuario	String	No vacío, único en el sistema	"El nombre de usuario no puede estar vacío o ya está en uso."	ValidacionException
Contraseña	String	Longitud	"La contraseña	ValidacionException

		mínima de 6 caracteres	debe tener al menos 6 caracteres.”	

Muchas de estas reglas se implementan en las clases Usuario/Inquilino y se apoyan en los métodos de IOUtils**.**

2.4 Registro de propiedad

Validaciones aplicadas a los campos del formulario de registro de propiedad:

Campo	Tipo de dato esperado	Rangos o formatos válidos	Mensaje de error personalizado	Excepciones manejadas
País	String	No vacío	“El país no puede estar vacío”	ValidacionException
Provincia	String	No vacío	“La provincia no puede estar vacía.”	“La provincia no puede estar vacía.”
Dirección detallada	String	No vacío	“La dirección detallada no puede estar vacía.”	ValidacionException
Tipo de propiedad	Enum	Debe ser uno de TipoPropiedad (CASA, APARTAMENTO, OFICINA, LOCAL)	“Tipo de propiedad inválido.”	IllegalArgumentException, ValidacionException
Metros Cuadrados	Int, Double	Mayor a 0	“Los metros cuadrados deben ser mayores a	NumberFormatException, ValidacionException

			cero.”	
Precio mensual Base	Int, double	Mayor a 0	“El precio mensual debe ser mayor a 0.”	NumberFormatException, IllegalArgumentException
Propietario	Propietario	Debe existir en la lista de propietarios	“Debe seleccionar un propietario válido para la propiedad”	ValidacionException

2.5 Creación de contrato de alquiler

Validaciones aplicadas a los campos involucrados en la creación de un contrato de alquiler:

Campo	Tipo de dato esperado	Rangos o formatos válidos	Mensaje de error personalizado	Excepciones manejadas
ID de Propiedad	Propiedad	Debe existir y estar disponible	“Propiedad no encontrada o no disponible para alquiler.”	ValidacionException
ID de Inquilino	Inquilino	Debe existir en la lista de inquilinos	“Inquilino no encontrado. Verifique el ID o selección.”	ValidacionException
Fecha de inicio	String/LocalDate	Formato de fecha válido	“Fecha de inicio inválida.”	ValidacionException, DateTimeParseException
Fecha de fin	String/LocalDate	Debe ser posterior a la fecha de inicio	“La fecha de fin debe ser posterior a la fecha de inicio.”	ValidacionException, DateTimeParseException

Monto de depósito	Int, double	Mayor a 0 (normalmente calculado desde la propiedad)	"El monto mensual debe ser mayor a cero."	ValidacionException
Día de corte	Int	Entre 1 y 28	"El día de corte debe estar entre 1 y 28."	NumberFormatException, ValidacionException

La clase ContratoAlquiler también calcula el campo proximoVencimiento y determina si un contrato está vencido mediante estaVencido().

2.6 Registro de pago

Campo	Tipo esperado	Regla / formato válido	Mensaje de error	Excepciones manejadas
Contrato	ContratoAlquiler	Debe existir en la lista de contratos	"Contrato no encontrado. Verifique el ID ingresado."	ValidacionException
Fecha de pago	LocalDate	No nula	"La fecha de pago es obligatoria."	ValidacionException
Monto	Double	Mayor a 0	"El monto del pago debe ser mayor a cero."	NumberFormatException, ValidacionException
Método pago	String	No vacío	"El método de pago no puede estar vacío."	ValidacionException
Referencia	String	No vacía	"La referencia del pago no puede estar vacía."	ValidacionException

La propiedad aTiempo se calcula comparando fechaPago con contrato.getProximoVencimiento().

2.7 Solicitud de mantenimiento

Campo	Tipo esperado	Regla / formato válido	Mensaje de error	Excepciones Manejadas
Contrato	ContratoAlquiler	Debe existir	"El contrato asociado a la solicitud es obligatorio."	ValidacionException
Descripción problema	String	No vacía	"La descripción del problema no puede estar vacía."	ValidacionException
Técnico asignado	Tecnico (opcional)	Si se asigna, debe ser válido	"El técnico asignado no puede ser nulo."	ValidacionException
Estado	EstadoSolicitud (enum)	Debe ser uno de los valores definidos	"Estado de solicitud inválido."	ValidacionException

2.8 Búsquedas, listados y reportes

En los listados (por ejemplo, para mostrar propiedades, inquilinos o contratos), se valida:

- Que las listas no estén completamente vacías.
- En búsqueda por ID, que el criterio no sea vacío y exista un elemento con dicho ID.

Si no se encuentran registros, se muestra un mensaje:

"No se encontraron registros para el criterio ingresado."

Los reportes generados por GestorReportes usan estas listas para producir resúmenes textuales sobre:

- Contratos por estado (EstadoContrato).
- Pagos de un contrato.
- Solicitudes de mantenimiento por estado (EstadoSolicitud).

3. Actualización basada en retroalimentación

3.1 Comentarios del Avance #1

A continuación, se listan los comentarios recibidos en el Avance #1 y las acciones tomadas para corregirlos o mejorarlos:

- El diagrama UML inicial no incluía todas las relaciones entre clases. Se actualizó el diagrama para incluir asociaciones, multiplicidades y visibilidades adecuadas entre las clases Inquilino, Propiedad y Contrato.
- Se solicitó un mejor uso de colecciones para el almacenamiento de datos. Se implementaron listas (ArrayList) para gestionar los inquilinos, propiedades y contratos de alquiler.
- Se recomendó mejorar la separación de responsabilidades. Se refactorizó el código para que las clases de dominio contengan la lógica relacionada con su propio comportamiento (por ejemplo, métodos de validación y representación de datos).

3.2 Comentarios del Avance #2

En el Avance #2 se hicieron mejoras adicionales relacionadas con la implementación del código y la interacción con el usuario:

- Se renombraron métodos para que reflejen de forma clara su propósito (por ejemplo, registrarInquilino(), registrarPropiedad(), crearContrato()).
- Se señaló la falta de validaciones robustas para las entradas del usuario. Se añadieron validaciones de tipo, rango y formato, así como manejo de excepciones específicas (NumberFormatException, InputMismatchException, IllegalArgumentException, etc.).
- Se recomendó un mejor balance entre el uso de consola y JOptionPane. Se ajustó el diseño para que las operaciones principales utilicen JOptionPane para formularios y mensajes, mientras que la consola se mantiene para listados y pruebas.

3.3 Resumen de mejoras aplicadas en el Avance #3

Gracias a la retroalimentación:

- El modelo de dominio ahora es más claro, **cohesivo** y alineado con la realidad del problema.
- Las validaciones son más completas, con mensajes amigables al usuario y un manejo centralizado de errores.
- Se añadió GestorUsuarios para encapsular el registro y la autenticación de usuarios.
- GestorReportes proporciona reportes de alto nivel, facilitando la presentación de resultados al profesor.
- El sistema cuenta con un flujo de navegación más intuitivo y una documentación alineada con el código final.

4. Entrega técnica

Para cumplir con los requisitos de entrega del Avance #3, se seguirán las siguientes acciones:

- Exportar este documento en formato PDF y guardarlo en la carpeta /docs del repositorio de GitHub del proyecto.
- Adjuntar el mismo archivo PDF en la tarea correspondiente de la plataforma Teams.
- Comprimir todos los archivos fuente .java del proyecto en un archivo ZIP organizado (respetando la estructura de paquetes) y subirlo también a la tarea de Teams.

Estas acciones garantizan una entrega ordenada y alineada con los criterios establecidos por el docente para la evaluación técnica del avance.

5. Conclusión

El Documento de Diseño de Interfaz y Validaciones del Avance #3 consolida las decisiones de diseño tomadas para el Sistema de Gestión de Alquiler de Propiedades, detalla el comportamiento esperado de la interfaz de usuario con JOptionPane y especifica las validaciones y excepciones que garantizan la integridad de los datos ingresados.

Asimismo, se documentan las mejoras realizadas a partir de la retroalimentación recibida en los avances anteriores, evidenciando una evolución positiva del proyecto en términos de claridad, robustez y buenas prácticas de programación orientada a objetos.