
TRABAJO GRUPAL (SEMANA 17)

TEMA: Sistema de depreciación de
Activo fijo para la Universidad
Tecnológica del Perú

Integrantes:
Injoque Guadaña, Fernando (25%)
Quispe Valqui, Rosa María (25%)
Ramirez Varas, Constantino (25%)
Ramirez Zuñiga, Axel Israel (25%)

Docente: Christian Eduardo Espinoza
Paredes

Salón: 22252

Historial de Versiones

VERSIÓN	PARTES QUE CAMBIAN	DESCRIPCIÓN DEL CAMBIO	FECHA DE CAMBIO	MODIFICADO POR	APROBADO POR
0.1.0	Carátula, Cronograma de trabajo inicial, Requerimientos funcionales	<ul style="list-style-type: none"> - Modificada la empresa objetivo. - Ordenados los integrantes. - Creada la primera versión del cronograma (cronograma de trabajo inicial). - Creación inicial de los títulos de los requerimientos REQMS-001 a REQMS-005. - Modificación de roles. 	22/08/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
0.2.0	Requerimientos funcionales	<ul style="list-style-type: none"> - Integración de las descripciones de los requerimientos REQMS-001 a REQMS-005 elaboradas por cada integrante. 	24/08/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
0.3.0	Requerimientos funcionales	<ul style="list-style-type: none"> - Creación de un documento integrado en Word Online para mejorar la gestión de versiones. - Expansión de los títulos de requerimientos de REQMS-005 a REQMS-024. 	25/08/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
0.3.1	Cronograma de trabajo final	<ul style="list-style-type: none"> - Actualización del cronograma de trabajo final. 	25/08/2024	Ramirez Varas, Constantino	Injoque Guadaña, Fernando
0.4.0	Requerimientos funcionales	<ul style="list-style-type: none"> - Añadida la descripción detallada de los requerimientos REQMS-013 a REQMS-018. 	26/08/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
0.4.1	Requerimientos funcionales	<ul style="list-style-type: none"> - Añadida la descripción detallada de los requerimientos REQMS-001 a REQMS-006. 	26/08/2024	Ramirez Zuñiga, Axel Israel	Injoque Guadaña, Fernando
0.5.0	Requerimientos funcionales	<ul style="list-style-type: none"> - Añadida la descripción detallada de los requerimientos REQMS-007 a REQMS-012. 	28/08/2024	Quispe Valqui, Rosa Maria	Injoque Guadaña, Fernando
0.5.1	Requerimientos funcionales	<ul style="list-style-type: none"> - Añadida la descripción detallada de los requerimientos REQMS-019 a REQMS-024. 	28/08/2024	Ramirez Varas, Constantino	Injoque Guadaña, Fernando
0.5.2	Requerimientos funcionales, Anexo	<ul style="list-style-type: none"> - Mejoradas las descripciones de los requerimientos REQMS-001 a REQMS-024. - Creación de nuevo requerimiento REQMS-025 y REQMS-026. - Creación del cuestionario de preguntas dentro del Anexo. 	28/08/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
0.5.3	Cronograma de trabajo final	<ul style="list-style-type: none"> - Actualización del cronograma de trabajo final. 	28/08/2024	Ramirez Varas, Constantino	Ramirez Zuñiga, Axel Israel
0.6.0	Introducción, Índice, Alcance, Actores del sistema, Requerimientos de alto nivel, Requerimientos funcionales	<ul style="list-style-type: none"> - Redacción de la introducción. -Delimitación del alcance. -Descripción de los actores del sistema. - Modificación de los requerimientos de alto nivel. - Mejorado el formato de la tabla de requerimientos funcionales. -Actualización del índice. 	29/08/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
0.6.1	Cronograma de trabajo final	<ul style="list-style-type: none"> - Actualización del cronograma de trabajo final. 	29/08/2024	Ramirez Varas, Constantino	Injoque Guadaña, Fernando

1.0.0	Carátula	-Añadido el porcentaje de participación de cada integrante en el proyecto.	30/08/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
1.0.1	Cronograma de trabajo inicial, Cronograma de trabajo final	-Expandido el cronograma de trabajo inicial a 18 semanas. -Actualizado el cronograma de trabajo final.	31/08/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
1.1.0	Requerimientos funcionales	-Modificado de los títulos de requerimientos a su forma infinitiva. -Mejorada la descripción de los requerimientos funcionales. -Eliminado los REQMS-021 (Generación de Estados Financieros) y REQMS-022 (Reporte Anual a la Administración).	21/09/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
1.2.0	Requerimientos funcionales	-Eliminado REQMS-020 (Generar reporte valorizado al área de contabilidad). -Incluida la funcionalidad del REQMS-009 (Generar reporte de depreciación anual) dentro del REQMS-010. -Añadido los REQMS-005 (Dar de baja activo fijo) y REQMS-007 (Gestionar el estado de un activo fijo) -Modificado título de REQMS-015 (Generar reporte de valor residual) a "Calcular valor residual"	28/09/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
1.3.0	Actores del sistema, Requerimientos de alto nivel, Requerimientos funcionales	-Eliminado "Usuario General", "Contador" y "Auditor" como actores del sistema. -Actualizado Requerimientos de alto nivel. -Actualizadas las descripciones de los requerimientos funcionales.	30/09/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
1.4.0	Apéndice	-Creación de Apéndice B con el código fuente y sus diagramas UML.	1/10/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
1.5.0	Principales pantallas	-Añadidas principales pantallas.	2/10/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
1.6.0	Cronograma inicial, Cronograma final, Índice	-Actualizados los estados del cronograma inicial a "Cumplida". -Actualizado el cronograma final. -Actualizado el Índice.	3/10/2024	Injoque Guadaña, Fernando	Ramirez Varas, Constantino
2.0.0	Carátula	-Actualizado el porcentaje de participación de cada integrante en el proyecto.	4/10/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
2.1.0	Requerimientos funcionales	-Eliminados los requerimientos de Auditar activos, Generar informes de auditoría, Planificar activos futuros, Realizar backup y restauración del sistema, y Proteger datos sensibles. -Añadidos los requerimientos de Generar depreciación acumulada mensual, Generar depreciación acumulada anual, Ordenar activos fijos por estado o categoría, Filtrar activos fijos, Buscar activo fijo, y Vender activo fijo.	12/10/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel

		-Renumerado y reordenado los requerimientos funcionales.			
2.2.0	Introducción, Alcance, Requerimientos de Alto Nivel	-Actualizada la introducción. --Actualizado el alcance. --Actualizada la descripción de los requerimientos funcionales.	3/11/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
2.3.0	Principales pantallas, Conclusiones	-Añadidas principales pantallas. -Añadidas las conclusiones.	4/11/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
2.4.0	Apéndice, Índice	-Actualización de Apéndice B con el código fuente. -Actualizado el Índice.	5/11/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
2.5.0	Cronograma de trabajo final	-Actualizado el cronograma de trabajo final.	6/11/2024	Injoque Guadaña, Fernando	Ramirez Varas, Constantino
3.0.0	Carátula	-Actualizado el porcentaje de participación de cada integrante en el proyecto.	8/11/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
3.1.0	Requerimientos funcionales	-Modificados los requerimientos de Ordenar activos fijos por estado o categoría, Planificar financieramente y Generar presupuestos a Ordenar activos fijos, Ubicar activos y Analizar activos en baja, respectivamente.	15/11/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
3.2.0	Alcance, Requerimientos de Alto Nivel	-Actualizado el alcance. --Actualizada la descripción de los requerimientos funcionales.	20/11/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
3.3.0	Principales pantallas	-Actualizadas principales pantallas.	29/11/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel
3.4.0	Apéndice, Índice	-Actualización de Apéndice B con el código fuente. -Actualizado el Índice.	1/12/2024	Injoque Guadaña, Fernando	Injoque Guadaña, Fernando
3.5.0	Cronograma de trabajo final	-Actualizado el cronograma de trabajo final.	3/12/2024	Injoque Guadaña, Fernando	Ramirez Varas, Constantino
4.0.0	Carátula	-Actualizado el porcentaje de participación de cada integrante en el proyecto.	6/12/2024	Injoque Guadaña, Fernando	Ramirez Zuñiga, Axel Israel

Índice

HISTORIAL DE VERSIONES	2
ÍNDICE	5
1 INTRODUCCIÓN.....	6
2 ALCANCE	7
3 CRONOGRAMA DE TRABAJO INICIAL	8
4 CRONOGRAMA DE TRABAJO FINAL.....	9
5 ACTORES DEL SISTEMA	10
6 REQUERIMIENTOS DE ALTO NIVEL	11
7 REQUERIMIENTOS FUNCIONALES.....	11
8 PRINCIPALES PANTALLAS	40
9 CONCLUSIONES.....	52
10 APÉNDICE	53

1 INTRODUCCIÓN

En el presente informe, se detalla el desarrollo de un sistema de depreciación de activos fijos para la Universidad Tecnológica del Perú, diseñado con el propósito de gestionar de manera eficiente los activos físicos de la institución a lo largo de su ciclo de vida. Este sistema permitirá el registro, control, depreciación y generación de reportes sobre los activos fijos, asegurando una administración precisa y conforme a las normativas contables y financieras vigentes.

Nuestro equipo está compuesto por estudiantes de Ingeniería de Software y Sistemas e Informática, cada uno con habilidades y conocimientos específicos que complementan el desarrollo del proyecto:

- **Fernando Injoque:** Con experiencia en gestión de proyectos, se desempeña como el Delegado del grupo, encargado de gestionar las tareas, elaborar el plan de trabajo, y asegurar el cumplimiento de los objetivos. Además, asume el rol de Integrador, responsable de unificar y verificar que todos los componentes del sistema funcionen de manera coherente antes de cada entrega.
- **Axel Ramirez:** Como Primer Oficial (PMO), tiene la responsabilidad de monitorear el cumplimiento del plan de trabajo, brindar soporte en la coordinación de tareas y servir de contrapeso en la evaluación del progreso del proyecto.
- **Constantino Ramirez:** Con habilidades en redacción y documentación, asume el rol de Documentador, encargado de elaborar las actas de las reuniones, registrar los acuerdos y pendientes, y mantener un registro preciso de las exposiciones grupales.
- **Rosa Quispe:** Con experiencia en diseño y comunicación visual, se encarga de la Imagen del grupo, planificando la presentación en el curso, definiendo tiempos y temas para destacar el trabajo realizado.

El nombre de nuestro sistema, **Valorium**, se origina en el concepto de “valor”, ya que simboliza nuestro compromiso de optimizar la gestión de activos fijos y controlar su depreciación de manera efectiva. Elegimos este nombre para reflejar nuestro objetivo de ofrecer una solución que integre tecnología y educación, atendiendo así las necesidades específicas de instituciones como la Universidad Tecnológica del Perú.

A través de este trabajo grupal, buscamos adquirir habilidades prácticas en el desarrollo de sistemas de información complejos, mejorar nuestra capacidad de trabajar en equipo y aprender a gestionar proyectos desde la fase de planificación hasta la implementación final. Además, esperamos fortalecer nuestras competencias en la integración de diferentes módulos del sistema y en la presentación eficaz de nuestro trabajo.

El sistema de depreciación de activos fijos que hemos desarrollado está diseñado para asistir a la Universidad Tecnológica del Perú en la gestión integral de sus activos físicos y su ciclo de vida, asegurando una depreciación precisa conforme a los estándares contables.

2 ALCANCE

El proyecto se enfoca en el desarrollo de un sistema de depreciación de activos fijos para la Universidad Tecnológica del Perú, destinado a optimizar la gestión de los activos físicos de la institución a lo largo de su ciclo de vida. Este sistema permitirá una administración eficiente mediante la integración de funciones clave como el registro, depreciación y generación de reportes sobre los activos fijos.

Contenido:

- **Registro de Activos:** El sistema permitirá registrar diferentes tipos de activos fijos, incluyendo equipos de cómputo, mobiliario, edificios y vehículos. Se incluirán validaciones para asegurar la precisión en el ingreso de datos.
- **Cálculo de Depreciación:** Se implementará el método de depreciación de línea recta para calcular la depreciación mensual y anual de los activos, facilitando la proyección de depreciación futura.
- **Generación de Reportes:** Se proporcionará reportes detallados sobre el valor de los activos, la depreciación acumulada, el valor residual y el inventario general de los activos.
- **Gestión de Revaluaciones y Renovaciones:** Se incluirán módulos para la gestión de revaluaciones de activos y la auditoría de activos renovados.
- **Historial de Movimientos:** Se mantendrá un historial de movimientos, registrando operaciones de alta, baja, modificación y venta de activos.

Supuestos:

- El sistema está diseñado específicamente para la Universidad Tecnológica del Perú y se ajusta a sus necesidades de gestión de activos fijos.
- El método de depreciación utilizado será exclusivamente el método de línea recta.
- El sistema se implementará en un entorno tecnológico compatible con sistemas operativos Windows.
- El lenguaje de programación utilizado será Java, y el desarrollo se realizará con la versión de Apache NetBeans 22.
- El sistema se conecta a una base de datos en MySQL, permitiendo almacenar y recuperar la información de los activos y sus movimientos.
- Los reportes y funcionalidades se generarán en formatos estándar y serán accesibles a los usuarios con permisos adecuados.

Limitaciones:

- El sistema no incluirá módulos para la capacitación de usuarios finales, por lo que se asume que el personal encargado tendrá conocimientos previos o recibirá formación externa.
- El sistema estará diseñado para funcionar en un entorno Windows; no se contemplan versiones para otros sistemas operativos.
- Las funcionalidades de integración con otros sistemas universitarios están fuera del alcance de este proyecto, por lo que se centrará únicamente en las capacidades internas del sistema de depreciación.
- No se considerará la gestión de activos que no estén bajo el control directo de la universidad, como los activos arrendados o subcontratados.

3 CRONOGRAMA DE TRABAJO INICIAL

El cronograma inicial se planteó en cuatro fases. La primera, antes del primer avance, se enfocó en establecer la comunicación y definir 24 requerimientos iniciales, alcanzando un avance del 30%. En la segunda fase, hasta la semana 8, se desarrolló la arquitectura del sistema y se implementaron ocho funcionalidades clave, logrando un avance del 50%. La tercera fase, hasta la semana 13, incluyó la implementación de módulos de reportes y conexión a la base de datos, completando el 70%. Finalmente, en la fase de cierre, se finalizó la implementación de notificaciones, orden, filtrado y búsqueda de activos, asegurando la entrega del proyecto al 100% en la semana 17.

Fecha	Semana	Actividad	Estado	Responsable
17/08/2024	1	Definición de canales de comunicación	Completada	Delegado
19/08/2024	2	Definición de días de reunión y horarios de comunicación	Completada	Primer Oficial
20/08/2024	2	Estructuración del cronograma de trabajo inicial	Completada	Delegado
22/08/2024	2	Reunión: Elección de roles y distribución de requerimientos	Completada	Todo el equipo
24/08/2024	2	Reunión: Revisión de requerimientos finales	Completada	Todo el equipo
26/08/2024	3	Reunión: Elaboración del cuestionario de preguntas	Completada	Todo el equipo
28/08/2024	3	Reunión: Revisión del informe final y preparación de la presentación en PPT	Completada	Todo el equipo
29/08/2024	3	Ajuste final de la presentación	Completada	Imagen
30/08/2024	3	Ensayo de la presentación	Completada	Todo el equipo
31/08/2024	3	Entrega final del avance 1 (30%) y presentación	Completada	Todo el equipo
20/09/2024	6	Reunión: Revisión y ajuste de los requerimientos	Completada	Todo el equipo
23/09/2024	7	Programación de los primeros requerimientos	Completada	Todo el equipo
27/09/2024	7	Reunión: Revisión del cumplimiento y funcionalidad de los requerimientos programados	Completada	Todo el equipo
28/09/2024	7	Integración de los requerimientos programados	Completada	Integrador
1/10/2024	8	Ajuste de la presentación	Completada	Imagen
3/10/2024	8	Ensayo de la presentación	Completada	Todo el equipo
5/10/2024	8	Entrega final del avance 2 (50%) y presentación	Completada	Todo el equipo
12/10/2024	9	Revisión y ajuste de los requerimientos anteriores	Completada	Todo el equipo
19/10/2024	10	Reunión: Distribución de los nuevos requerimientos	Completada	Todo el equipo
21/10/2024	10	Programación de los nuevos requerimientos	Completada	Todo el equipo
26/10/2024	11	Integración de los requerimientos programados	Completada	Integrador
28/10/2024	12	Conexión con la base de datos	Completada	Integrador
30/10/2024	13	Ajuste de la presentación	Completada	Imagen
2/11/2024	13	Ensayo de la presentación	Completada	Todo el equipo
9/11/2024	13	Entrega final del avance 3 (70%) y presentación	Completada	Todo el equipo

11/11/2024	14	Reunión: Distribución de los nuevos requerimientos	Completada	Todo el equipo
18/11/2024	15	Programación de los nuevos requerimientos	Completada	Todo el equipo
25/11/2024	16	Integración de los requerimientos programados	Completada	Integrador
28/11/2024	16	Finalización de la documentación técnica del sistema	Completada	Integrador y Documentador
2/12/2024	17	Ensayo de la presentación para la entrega final	Completada	Todo el equipo
7/12/2024	17	Presentación y entrega final del trabajo	Completada	Todo el equipo

4 CRONOGRAMA DE TRABAJO FINAL

Fecha Inicial	Fecha Real	Actividad	Estado	Responsable
17/08/2024	17/08/2024	Definición de WhatsApp como canal de comunicación principal	Completada	Delegado
19/08/2024	19/08/2024	Definición de días de reunión y horarios de comunicación	Completada	Primer Oficial
20/08/2024	20/08/2024	Estructuración del cronograma de trabajo inicial	Completada	Delegado
22/08/2024	22/08/2024	Reunión: Elección de roles, definición de términos, y distribución de 5 requerimientos	Completada	Todo el equipo
24/08/2024	25/08/2024	Reunión: Establecimiento de 24 requerimientos, debate sobre términos, propuestas de mejoras, y preparación del formato Word Online	Completada	Todo el equipo
26/08/2024	28/08/2024	Reunión: Revisión de requerimientos finales, incorporación de capa de seguridad de accesos (REQMS-025 y REQMS-026), y elaboración del cuestionario de preguntas	Completada	Todo el equipo
28/08/2024	29/08/2024	Reunión: Revisión del informe final y preparación de la presentación en PPT	Completada	Todo el equipo
30/08/2024	30/08/2024	Ensayo y ajuste final de la presentación	Completada	Todo el equipo
31/08/2024	31/08/2024	Entrega final del avance 1 (30%)	Completada	Todo el equipo
20/09/2024	21/09/2024	Reunión: Revisión y ajuste de los requerimientos; elección y distribución de los 8 requerimientos a programar	Completada	Todo el equipo
23/09/2024	22/09/2024	Creación de código de los 8 requerimientos	Completada	Todo el Equipo
27/09/2024	28/09/2024	Reunión: Revisión del código elaborado, ajuste de los requerimientos funcionales y	Completada	Todo el equipo

		estructuración de Códigos Individuales		
28/09/2024	30/09/2024	Integración y documentación de los requerimientos programados en <i>javadocs</i> . Creación de Apéndice B con el código fuente y sus diagramas UML	Completada	Integrador
1/10/2024	2/10/2024	Ajuste de la presentación	Completada	Imagen
3/10/2024	3/10/2024	Ensayo de la presentación	Completada	Todo el equipo
5/10/2024	5/10/2024	Entrega final del avance 2 (50%) y presentación	Completada	Todo el equipo
12/10/2024	13/10/2024	Revisión y ajuste de los 8 requerimientos anteriores	Completada	Todo el equipo
19/10/2024	23/10/2024	Reunión: Distribución de los 7 nuevos requerimientos	Completada	Todo el equipo
21/10/2024	24/10/2024	Programación de los 7 nuevos requerimientos	Completada	Todo el equipo
26/10/2024	2/11/2024	Integración de los 7 requerimientos programados	Completada	Integrador
28/10/2024	3/11/2024	Conexión con la base de datos con <i>mySQL</i>	Completada	Integrador
30/10/2024	4/11/2024	Ajuste de la presentación	Completada	Imagen
2/11/2024	7/11/2024	Ensayo de la presentación	Completada	Todo el equipo
9/11/2024	9/11/2024	Entrega final del avance 3 (70%) y presentación	Completada	Todo el equipo
11/11/2024	15/11/2024	Reunión: Distribución de los 9 nuevos requerimientos	Completada	Todo el equipo
18/11/2024	22/11/2024	Programación de los 9 nuevos requerimientos	Completada	Todo el equipo
25/11/2024	29/11/2024	Integración de los 9 requerimientos programados	Completada	Integrador
28/11/2024	1/12/2024	Finalización de la documentación técnica del sistema	Completada	Integrador y Documentador
2/12/2024	5/12/2024	Ajuste y ensayo de la presentación para la entrega final	Completada	Todo el equipo
7/12/2024	7/12/2024	Entrega final (100%) y presentación	Completada	Todo el equipo

5 ACTORES DEL SISTEMA

Actores	Descripción
Administrador	<ul style="list-style-type: none"> • Es el responsable principal de la configuración del sistema, incluyendo la definición de roles, permisos, y la gestión general de los activos fijos. También supervisa la incorporación de nuevos activos y la renovación de activos existentes. Tiene acceso a todas las funcionalidades del sistema, incluyendo la generación de informes, auditorías, y la planificación financiera.

6 REQUERIMIENTOS DE ALTO NIVEL

Requerimiento de Alto Nivel	Descripción
REQMS-001	Controlar acceso basado en roles
REQMS-002	Registrar activos fijos
REQMS-003	Calcular la vida útil según categoría
REQMS-004	Aplicar el método de depreciación por línea recta
REQMS-005	Calcular depreciación mensual
REQMS-006	Generar depreciación acumulada mensual
REQMS-007	Generar depreciación acumulada anual
REQMS-008	Calcular valor residual
REQMS-009	Notificar para la revisión de activos
REQMS-010	Mostrar el estado de un activo fijo
REQMS-011	Modificar activo fijo
REQMS-012	Ver historial de cambios de activos fijos
REQMS-013	Ordenar activos fijos
REQMS-014	Filtrar activos fijos
REQMS-015	Buscar activo fijo
REQMS-016	Vender activo fijo
REQMS-017	Dar de baja activo fijo
REQMS-018	Exportar reporte de depreciación
REQMS-019	Proyectar depreciación futura
REQMS-020	Generar reporte de inventario
REQMS-021	Gestionar la revaluación de activos fijos
REQMS-022	Gestionar la renovación de activos
REQMS-023	Ubicar activos
REQMS-024	Analizar activos en baja

7 REQUERIMIENTOS FUNCIONALES

Código de Requerimiento	Descripción del Requerimiento Funcional	Propuesta de Solución
REQMS-001	Controlar acceso basado en roles	<p>1. El usuario ingresa sus credenciales desde una ventana:</p> <ul style="list-style-type: none"> El sistema presenta una ventana gráfica donde el usuario debe ingresar su nombre de usuario y contraseña. Dependiendo del rol asignado (Administrador), el sistema limita el acceso a funcionalidades específicas dentro de la interfaz gráfica principal. <p>2. Verificación de credenciales:</p> <ul style="list-style-type: none"> El sistema valida las credenciales ingresadas contra una lista de usuarios registrados. Si las credenciales son incorrectas, se muestra un cuadro de diálogo con el

		<p>mensaje "Usuario o contraseña incorrectos. Intentos restantes: X."</p> <p>3. Validaciones durante la autenticación:</p> <ul style="list-style-type: none"> Si el usuario excede el número permitido de intentos fallidos, la sesión se cierra automáticamente y el sistema finaliza su ejecución. <p>4. El sistema controla el acceso dependiendo del rol del usuario:</p> <ul style="list-style-type: none"> Administrador: acceso total para registrar, modificar y gestionar activos. <p>5. Notificaciones y mensajes:</p> <ul style="list-style-type: none"> Al autenticarse exitosamente, el sistema muestra un mensaje emergente: "Ingreso exitoso". En caso de fallar la autenticación, el sistema informa sobre los intentos restantes. <p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> Roles claramente definidos con permisos de acceso limitados. Seguridad en los intentos fallidos de inicio de sesión. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> No se contempla la creación de roles personalizados. No incluye autenticación multifactor o integración con servicios externos de autenticación. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> Normal: El usuario ingresa sus credenciales correctamente a través de la ventana y accede a las funcionalidades del sistema según su rol. Alternativo 1: El usuario ingresa credenciales incorrectas, se reduce el número de intentos restantes y se muestra un mensaje de error. Alternativo 2: El usuario excede los intentos de autenticación permitidos y el sistema cierra la sesión, finalizando su ejecución.
REQMS-002	Registrar activos fijos	<p>1. El Administrador utiliza la interfaz gráfica para registrar un activo:</p> <ul style="list-style-type: none"> El sistema presenta un formulario en la interfaz gráfica para que el usuario ingrese los datos del activo, como el nombre, categoría, valor inicial y fecha de adquisición.

		<p>2. El sistema verifica los datos ingresados:</p> <ul style="list-style-type: none"> • El nombre debe tener al menos 3 caracteres. • El valor inicial debe ser mayor a 0. • La fecha de adquisición debe ser válida y estar dentro de un rango razonable (entre 1997 y la fecha actual). <p>3. Validaciones durante el registro:</p> <ul style="list-style-type: none"> • Si los datos son incorrectos, el sistema muestra un mensaje que indica el error específico. <p>4. Confirmación del registro:</p> <ul style="list-style-type: none"> • Al finalizar el registro, el sistema muestra un cuadro de diálogo con los detalles del activo registrado: "Activo registrado correctamente: [detalle del activo]". <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Si hay errores en los datos ingresados, se muestra un mensaje de error específico sobre qué dato es inválido.
		<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • El sistema realiza validaciones específicas para el nombre, valor y fecha del activo. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No se contempla la integración automática de activos desde otros sistemas externos. • No se permiten registros de activos sin una fecha válida.
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El Administrador ingresa correctamente todos los datos de un nuevo activo, el sistema valida los datos y registra el activo, mostrando un mensaje de confirmación. • Alternativo 1: El usuario ingresa datos incorrectos (por ejemplo, un valor inicial inválido o una fecha fuera de rango). El sistema muestra un mensaje de error específico y solicita corregir los datos antes de permitir el registro.
REQMS-003	Calcular la vida útil según categoría	<p>1. El sistema calcula automáticamente la vida útil del activo según su categoría:</p> <ul style="list-style-type: none"> • Basándose en las reglas de la SUNAT, el sistema asigna una vida útil predefinida a cada categoría de activo (por ejemplo, 4 años para equipos de cómputo). <p>2. Integración con otras funcionalidades:</p>

		<ul style="list-style-type: none"> La vida útil calculada se utiliza para los cálculos de depreciación mensual y acumulada, y se almacena como parte de los detalles del activo. <p>3. Validaciones:</p> <ul style="list-style-type: none"> El sistema asegura que los activos tengan una vida útil válida, y que categorías como "Terreno" no tengan vida útil asignada, ya que no se deprecian. <table border="1"> <tr><th>Actores</th></tr> <tr><td>Administrador</td></tr> <tr><th>Criterios de Cumplimiento</th></tr> <tr><td>Necesario</td></tr> <tr><th>Prioridad</th></tr> <tr><td>Alta</td></tr> <tr><th>Consideraciones</th></tr> <tr> <td> <ul style="list-style-type: none"> El cálculo de vida útil sigue estrictamente las reglas predefinidas internas del sistema basadas en la SUNAT. </td></tr> <tr><th>Alcance No Contemplado</th></tr> <tr> <td> <ul style="list-style-type: none"> No se contempla calcular la vida útil basada en datos externos o en términos de meses individuales. </td></tr> </table> <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> Normal: El sistema asigna automáticamente la vida útil estimada del activo al momento de su registro o modificación, basándose en las reglas predefinidas por la SUNAT según la categoría del activo. Alternativo 1: El usuario modifica la categoría del activo, y el sistema actualiza la vida útil estimada automáticamente. Si la nueva categoría es "Terreno", el sistema no asigna vida útil ni aplica depreciación. 	Actores	Administrador	Criterios de Cumplimiento	Necesario	Prioridad	Alta	Consideraciones	<ul style="list-style-type: none"> El cálculo de vida útil sigue estrictamente las reglas predefinidas internas del sistema basadas en la SUNAT. 	Alcance No Contemplado	<ul style="list-style-type: none"> No se contempla calcular la vida útil basada en datos externos o en términos de meses individuales.
Actores												
Administrador												
Criterios de Cumplimiento												
Necesario												
Prioridad												
Alta												
Consideraciones												
<ul style="list-style-type: none"> El cálculo de vida útil sigue estrictamente las reglas predefinidas internas del sistema basadas en la SUNAT. 												
Alcance No Contemplado												
<ul style="list-style-type: none"> No se contempla calcular la vida útil basada en datos externos o en términos de meses individuales. 												
REQMS-004	Aplicar el método de depreciación por línea recta	<p>1. El sistema aplica el método de depreciación por línea recta:</p> <ul style="list-style-type: none"> El sistema utiliza por defecto el método de depreciación por línea recta para todos los activos que no sean terrenos (que no se deprecian). Este método se aplica automáticamente en el registro y modificación de activos dentro del sistema. <p>2. Integración con otras funcionalidades:</p> <ul style="list-style-type: none"> El método de depreciación por línea recta se emplea para calcular la depreciación mensual y anual de cada activo, lo cual influye en los reportes generados y en el valor neto contable. El cálculo de depreciación se realiza en función del valor inicial y la vida útil del activo. 										

		<p>3. Validaciones:</p> <ul style="list-style-type: none"> • El sistema valida que los activos sujetos a depreciación tengan una vida útil asignada y que el valor inicial sea mayor a 0. • Los terrenos no se deprecian, y el sistema asigna automáticamente una depreciación de 0 para estos. <p>4. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Al visualizar los reportes de depreciación, el sistema informa sobre el método utilizado (línea recta) en los cálculos de depreciación mensual y acumulada. <table border="1"> <tr><th>Actores</th></tr> <tr><td>Administrador</td></tr> <tr><th>Criterios de Cumplimiento</th></tr> <tr><td>Necesario</td></tr> <tr><th>Prioridad</th></tr> <tr><td>Alta</td></tr> <tr><th>Consideraciones</th></tr> <tr> <td> <ul style="list-style-type: none"> • La depreciación se realiza únicamente usando el método de línea recta y se aplica automáticamente según la vida útil y categoría del activo. </td></tr> <tr><th>Alcance No Contemplado</th></tr> <tr> <td> <ul style="list-style-type: none"> • No se incluye la opción de cambiar a otros métodos de depreciación distintos al de línea recta. </td></tr> <tr><th>Escenarios del Negocio</th></tr> <tr> <td> <ul style="list-style-type: none"> • Normal: El sistema aplica el método de depreciación por línea recta de forma predeterminada a todos los activos que lo requieran, calculando automáticamente la depreciación mensual y acumulada en función de la vida útil. • Alternativo 1: El sistema detecta que el activo es un terreno, lo que impide que se aplique cualquier método de depreciación, asignando una depreciación de 0. </td></tr> </table>	Actores	Administrador	Criterios de Cumplimiento	Necesario	Prioridad	Alta	Consideraciones	<ul style="list-style-type: none"> • La depreciación se realiza únicamente usando el método de línea recta y se aplica automáticamente según la vida útil y categoría del activo. 	Alcance No Contemplado	<ul style="list-style-type: none"> • No se incluye la opción de cambiar a otros métodos de depreciación distintos al de línea recta. 	Escenarios del Negocio	<ul style="list-style-type: none"> • Normal: El sistema aplica el método de depreciación por línea recta de forma predeterminada a todos los activos que lo requieran, calculando automáticamente la depreciación mensual y acumulada en función de la vida útil. • Alternativo 1: El sistema detecta que el activo es un terreno, lo que impide que se aplique cualquier método de depreciación, asignando una depreciación de 0.
Actores														
Administrador														
Criterios de Cumplimiento														
Necesario														
Prioridad														
Alta														
Consideraciones														
<ul style="list-style-type: none"> • La depreciación se realiza únicamente usando el método de línea recta y se aplica automáticamente según la vida útil y categoría del activo. 														
Alcance No Contemplado														
<ul style="list-style-type: none"> • No se incluye la opción de cambiar a otros métodos de depreciación distintos al de línea recta. 														
Escenarios del Negocio														
<ul style="list-style-type: none"> • Normal: El sistema aplica el método de depreciación por línea recta de forma predeterminada a todos los activos que lo requieran, calculando automáticamente la depreciación mensual y acumulada en función de la vida útil. • Alternativo 1: El sistema detecta que el activo es un terreno, lo que impide que se aplique cualquier método de depreciación, asignando una depreciación de 0. 														
REQMS-005	Calcular depreciación mensual	<p>1. El sistema calcula automáticamente la depreciación mensual de los activos:</p> <ul style="list-style-type: none"> • Utilizando el método de depreciación por línea recta, el sistema calcula la depreciación mensual de cada activo en función de su vida útil y valor inicial. • Este cálculo se realiza automáticamente cuando el activo es registrado o modificado. <p>2. Integración con otras funcionalidades:</p> <ul style="list-style-type: none"> • El cálculo de depreciación mensual afecta directamente el valor neto contable y se refleja en los reportes generados por el sistema. • Los activos con estado "Baja" o aquellos sin vida útil (como terrenos) no se deprecian. 												

		<p>3. Validaciones:</p> <ul style="list-style-type: none"> • El sistema asegura que los activos tengan una vida útil válida y que no se calculen depreciaciones para aquellos en estado "Baja" o "Depreciado". <p>4. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Al generar reportes, se muestra la depreciación mensual calculada para cada activo junto con los detalles del activo, como la categoría y el valor inicial.
		<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • Los activos que han alcanzado su vida útil se marcan automáticamente como "Depreciados" y dejan de generar depreciación mensual. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No incluye otros métodos de depreciación diferentes a la línea recta. • No se considera el ajuste de la depreciación mensual según las fluctuaciones de mercado.
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El sistema calcula automáticamente la depreciación mensual de los activos según su vida útil y valor inicial, reflejando el resultado en la interfaz gráfica. • Alternativo 1: El sistema detecta que un activo ha alcanzado el final de su vida útil, lo marca como "Depreciado", y deja de calcular su depreciación mensual. • Alternativo 2: El sistema intenta calcular la depreciación mensual de un activo con estado "Baja" o "Depreciado", y no realiza ningún cálculo, mostrando un mensaje que indica que el activo ya no se deprecia.
REQMS-006	Generar depreciación acumulada mensual	<p>1. El Administrador utiliza la interfaz gráfica para generar la depreciación acumulada mensual de un activo:</p> <ul style="list-style-type: none"> • El sistema muestra un formulario donde el administrador selecciona el activo deseado. • Una vez seleccionado, el sistema calcula automáticamente la depreciación mensual basada en el costo inicial y la vida útil del activo. <p>2. El sistema verifica los datos del activo seleccionado:</p> <ul style="list-style-type: none"> • Verifica que el activo sea depreciable (excluyendo activos como "Terreno").

		<ul style="list-style-type: none"> Calcula la depreciación mensual dividiendo el costo inicial entre la vida útil en meses. <p>3. Validaciones durante el cálculo:</p> <ul style="list-style-type: none"> Si el activo ha alcanzado su vida útil, el sistema marca el activo como "Depreciado" y no incrementa más la depreciación acumulada. Si el activo no cumple con los requisitos (por ejemplo, categoría no depreciable), se muestra un mensaje de error. <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> Al finalizar el cálculo, el sistema despliega en pantalla el valor de la depreciación acumulada mensual actualizada del activo. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Si el activo no es depreciable o ya está depreciado, el sistema muestra un mensaje de advertencia indicando que no es posible calcular más depreciación para este activo.
		Actores
		Administrador
		Criterios de Cumplimiento
		Necesario
		Prioridad
		Alta
		Consideraciones
		<ul style="list-style-type: none"> El sistema debe verificar que solo los activos depreciables puedan calcular la depreciación mensual. Debe realizarse el cálculo utilizando el método de depreciación de línea recta y mostrar resultados redondeados a dos decimales.
		Alcance No Contemplado
		<ul style="list-style-type: none"> No se contempla la inclusión de activos que no se deprecian (como terrenos). No incluye la depreciación de activos que han sido dados de baja o vendidos.
		Escenarios del Negocio
		<ul style="list-style-type: none"> Normal: El administrador selecciona un activo depreciable, y el sistema genera la depreciación acumulada mensual, mostrando el resultado sin problemas. Alternativo 1: El activo seleccionado ya ha alcanzado el final de su vida útil. El sistema muestra un mensaje indicando que el activo está completamente depreciado. Alternativo 2: El usuario intenta calcular la depreciación de un activo no depreciable (por ejemplo, "Terreno"). El sistema muestra un mensaje de error específico: "El activo seleccionado no es sujeto de depreciación".

REQMS-007	Generar depreciación acumulada anual	<p>1. El Administrador utiliza la interfaz gráfica para generar la depreciación acumulada anual de un activo:</p> <ul style="list-style-type: none"> • El sistema presenta una interfaz donde el administrador puede seleccionar el activo y el año específico para calcular la depreciación acumulada. • El sistema calcula la depreciación anual sumando las depreciaciones mensuales acumuladas hasta el fin del año fiscal o el final de la vida útil del activo, según corresponda. <p>2. El sistema verifica los datos del activo seleccionado:</p> <ul style="list-style-type: none"> • Verifica que el activo esté en estado "Alta" y que aún esté dentro de su vida útil. • Si el activo ha alcanzado el fin de su vida útil, se notifica que ya no acumula depreciación. <p>3. Validaciones durante el cálculo:</p> <ul style="list-style-type: none"> • La depreciación acumulada anual no debe exceder el valor inicial menos el valor residual del activo. • Si el activo no es depreciable, el sistema muestra un mensaje de error. <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> • Al finalizar, el sistema muestra en pantalla el monto de la depreciación acumulada anual y el valor neto actual del activo. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Si el cálculo no es posible porque el activo ya está completamente depreciado, el sistema muestra un mensaje: "El activo ya ha alcanzado su vida útil y no acumula más depreciación."
Actores	Administrador	
Criterios de Cumplimiento	Necesario	
Prioridad	Alta	
Consideraciones	<ul style="list-style-type: none"> • El sistema debe calcular la depreciación acumulada anual utilizando el método de depreciación de línea recta y verificar que el activo esté dentro de su vida útil. • La depreciación acumulada debe actualizarse cada fin de año o al llegar al final de la vida útil del activo. 	

		<p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No incluye la depreciación de activos no sujetos a depreciación. • No contempla la depreciación de activos que han sido dados de baja o vendidos. • No considera el cambio de método de depreciación durante el ciclo de vida del activo. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador selecciona un activo que aún se deprecia, y el sistema calcula la depreciación acumulada anual correctamente. • Alternativo 1: El activo ya está completamente depreciado. El sistema muestra un mensaje indicando que no es posible calcular más depreciación. • Alternativo 2: El administrador selecciona un activo no depreciable (por ejemplo, un terreno). El sistema muestra un mensaje de error indicando que el activo no es sujeto a depreciación.
REQMS-008	Calcular valor residual	<ol style="list-style-type: none"> 1. El Administrador utiliza la interfaz gráfica para calcular el valor residual de un activo: <ul style="list-style-type: none"> • El sistema permite que el administrador seleccione un activo específico y realice el cálculo del valor residual. • El valor residual se calcula restando la depreciación acumulada actual del valor inicial del activo. 2. El sistema verifica los datos del activo seleccionado: <ul style="list-style-type: none"> • El sistema verifica que el activo esté en estado "Alta" o "Depreciado" y que el cálculo del valor residual sea aplicable. • Para activos no depreciables (como terrenos), el sistema establece automáticamente el valor residual como el valor inicial. 3. Validaciones durante el cálculo: <ul style="list-style-type: none"> • Si el activo ha alcanzado el fin de su vida útil, el valor residual se fija en cero. • Para activos con un valor residual que resulte en negativo, el sistema ajusta el valor a cero, ya que no puede ser menor que este valor. 4. Confirmación de operación: <ul style="list-style-type: none"> • Una vez realizado el cálculo, el sistema muestra en pantalla el valor residual del activo. 5. Mensajes y notificaciones: <ul style="list-style-type: none"> • Si el activo no es susceptible a depreciación y, por lo tanto, no tiene un valor residual calculable, el sistema muestra un mensaje: "El activo no es

		<p>depreciable; el valor residual es igual al costo inicial."</p> <p>Actores Administrador</p> <p>Criterios de Cumplimiento Necesario</p> <p>Prioridad Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • El cálculo del valor residual debe tener en cuenta el estado y categoría del activo, utilizando el método de depreciación de línea recta cuando corresponda. • Si el activo alcanza un valor residual negativo, el sistema debe ajustar el valor a cero para evitar inconsistencias. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No se contempla el cálculo del valor residual para activos dados de baja o vendidos. • No considera el uso de métodos de depreciación diferentes al de línea recta. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador selecciona un activo con depreciación, y el sistema calcula el valor residual correctamente. • Alternativo 1: El activo seleccionado es no depreciable. El sistema muestra un mensaje indicando que el valor residual es igual al costo inicial. • Alternativo 2: El valor residual del activo resulta en negativo. El sistema ajusta el valor a cero y muestra el valor final en pantalla.
REQMS-009	Notificar para la revisión de activos	<p>1. El administrador interactúa con el sistema para recibir notificaciones de revisión:</p> <ul style="list-style-type: none"> • El sistema genera automáticamente notificaciones basadas en condiciones específicas: <ul style="list-style-type: none"> ○ Revisión periódica de activos (por ejemplo, trimestral, semestral o anual). ○ Alerta sobre activos cuya vida útil está próxima a completarse (menos de 6 meses restantes). • Las notificaciones se muestran en un panel dedicado o emergen como alertas en la interfaz. <p>2. El sistema verifica la información del activo para determinar la necesidad de revisión:</p> <ul style="list-style-type: none"> • Revisa las fechas relevantes, como: <ul style="list-style-type: none"> ○ Fecha de adquisición del activo. ○ Periodo de vida útil restante.

	<ul style="list-style-type: none"> Considera el estado del activo, enviando notificaciones únicamente para activos en "Alta" o "Depreciado". <p>3. Validaciones para la generación de notificaciones:</p> <ul style="list-style-type: none"> Evitar la duplicación de notificaciones dentro del mismo periodo. No generar alertas para activos en estados no aplicables, como "Baja" o "Baja por venta". Validar que los datos del activo (fechas y estado) sean consistentes antes de crear la notificación. <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> Una vez verificada la necesidad de revisión, el sistema muestra la notificación al administrador: <ul style="list-style-type: none"> Ejemplo: "El activo [nombre] requiere revisión." <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Si el activo no cumple los criterios para revisión, el sistema muestra un mensaje: "No hay activos pendientes de revisión en este periodo." Las notificaciones generadas se registran con detalles claros: <ul style="list-style-type: none"> Nombre del activo, estado actual, fecha de próxima revisión sugerida.
	<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Opcional</p> <p>Prioridad</p> <p>Media</p> <p>Consideraciones</p> <ul style="list-style-type: none"> Las notificaciones deben basarse en cálculos automáticos considerando la fecha de adquisición, vida útil, y estado del activo. Los periodos de revisión deben ser configurables (mensual, trimestral, etc.). <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> No se consideran notificaciones para activos ya revisados en los últimos 6 meses. No se generan notificaciones para activos en baja o en proceso de venta.
	<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> Normal: El sistema detecta que un activo requiere revisión y genera una notificación para el administrador.

	<ul style="list-style-type: none"> • Alternativo 1: No hay activos con revisión pendiente; el sistema muestra un mensaje indicando que no hay acciones requeridas. • Alternativo 2: Un activo revisado recientemente no genera una nueva notificación. 	
REQMS-010	<p>Mostrar el estado de un activo fijo</p>	<ol style="list-style-type: none"> 1. El Administrador utiliza la interfaz gráfica para visualizar el estado de un activo fijo: <ul style="list-style-type: none"> • El sistema permite que el administrador seleccione un activo en la lista de activos registrados para consultar su estado actual. • El estado puede ser "Alta", "Depreciado", "Baja" o "Baja por venta". 2. El sistema verifica los datos del activo seleccionado: <ul style="list-style-type: none"> • Verifica que el activo tenga un estado asignado en función de su vida útil y posibles modificaciones previas (como ventas o bajas). • Si el activo ya ha alcanzado el final de su vida útil, su estado se actualiza automáticamente a "Depreciado". 3. Validaciones durante la consulta: <ul style="list-style-type: none"> • Si el activo ha sido dado de baja o vendido, el sistema muestra un mensaje indicando que el activo ya no está disponible para operaciones. 4. Confirmación de operación: <ul style="list-style-type: none"> • El sistema muestra en pantalla el estado actual del activo, junto con detalles como la fecha de adquisición, vida útil restante y depreciación acumulada si es aplicable. 5. Mensajes y notificaciones: <ul style="list-style-type: none"> • Si el activo está en estado "Baja" o "Baja por venta", el sistema muestra un mensaje específico: "El activo seleccionado ha sido dado de baja y no está disponible para operaciones." <p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • El sistema debe verificar y mostrar el estado correcto del activo en tiempo real, considerando cualquier cambio reciente o actualización automática de estado. <p>Alcance No Contemplado</p>

		<ul style="list-style-type: none"> • No se contempla la modificación del estado del activo a través de esta consulta. • No se incluye la consulta del estado para activos no registrados en el sistema.
Escenarios del Negocio		
REQMS-011	Modificar activo fijo	<p>1. El Administrador selecciona un activo desde la interfaz gráfica para modificar:</p> <ul style="list-style-type: none"> • El sistema presenta una lista de activos registrados dentro de la interfaz gráfica, permitiendo al usuario seleccionar el activo a modificar. <p>2. El sistema permite modificar los datos del activo:</p> <ul style="list-style-type: none"> • Una vez seleccionado el activo, el usuario puede editar campos como el nombre, categoría, valor inicial y fecha de adquisición a través de formularios en la interfaz gráfica. • El sistema actualiza automáticamente la vida útil del activo según su nueva categoría. <p>3. Validaciones durante la modificación:</p> <ul style="list-style-type: none"> • El nombre debe tener al menos 3 caracteres, el valor inicial debe ser mayor a 0, y la fecha de adquisición debe ser válida. • Si los datos no cumplen con los criterios, el sistema muestra un cuadro de diálogo con un mensaje de error. <p>4. Confirmación de la modificación:</p> <ul style="list-style-type: none"> • Al finalizar la modificación, el sistema muestra un mensaje confirmando los detalles del activo actualizado: "Activo modificado correctamente: [detalle del activo]."

Actores

Administrador

Criterios de Cumplimiento

Necesario

Prioridad

Media

Consideraciones

- El sistema valida los datos modificados antes de guardar los cambios.

		<p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No incluye la edición masiva de activos. • No permite la modificación de activos con estado "Baja" o "Depreciado".
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El Administrador selecciona un activo existente, modifica sus datos (nombre, categoría, valor, fecha de adquisición), y el sistema valida los cambios antes de confirmarlos. • Alternativo 1: El usuario intenta modificar un activo, pero ingresa datos inválidos (por ejemplo, una fecha incorrecta o un valor inicial negativo). El sistema muestra un mensaje de error y no guarda los cambios hasta que los datos sean corregidos.
REQMS-012	Ver historial de cambios de activos fijos	<ol style="list-style-type: none"> 1. El Administrador utiliza la interfaz gráfica para ver el historial de cambios de todos los activos fijos: <ul style="list-style-type: none"> • El sistema presenta una vista general en la que se listan todos los cambios o movimientos realizados en los activos, sin necesidad de seleccionar uno en particular. • Cada entrada en el historial incluye detalles como el nombre del activo, tipo de cambio (por ejemplo, cambio de estado, venta, baja), fecha y descripción del cambio. 2. El sistema muestra los datos de todos los movimientos registrados: <ul style="list-style-type: none"> • El historial incluye información detallada de cada cambio en todos los activos, proporcionando una visión general de las modificaciones a lo largo del tiempo. • Los registros están ordenados cronológicamente. 3. Validaciones durante la consulta: <ul style="list-style-type: none"> • Si no existen registros en el historial, el sistema muestra un mensaje indicando que no hay movimientos registrados. 4. Confirmación de operación: <ul style="list-style-type: none"> • Una vez que el historial se ha cargado, el sistema muestra en pantalla la lista de cambios realizada en todos los activos de forma cronológica. 5. Mensajes y notificaciones: <ul style="list-style-type: none"> • Si no existen registros de cambios, el sistema muestra un mensaje: "No existen movimientos registrados en el historial de activos."
		Actores
		Administrador
		Criterios de Cumplimiento
		Necesario
		Prioridad

		<p>Media</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • El sistema debe garantizar que cada cambio en cualquier activo se registre adecuadamente y esté disponible para consulta en la vista de historial general. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No se contempla la edición o eliminación de registros de cambios en el historial. • No incluye la visualización de cambios de activos que no han registrado movimientos desde su registro inicial. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador accede al historial y visualiza todos los movimientos realizados en los activos con detalles organizados cronológicamente. • Alternativo 1: No existen registros de cambios en el historial general. El sistema muestra un mensaje indicando que no hay movimientos registrados.
		<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • Incluir opciones para visualizar el reporte anual por categoría de activo o ubicación. • Posibilidad de programar la generación automática del reporte anual. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No se contempla la generación de reportes que no incluyan la depreciación de activos fijos. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador genera el reporte anual que resume toda la información de depreciación del año, y se utiliza para la planificación financiera del próximo año. • Alternativo 1: Se detectan errores en los datos de depreciación, y el sistema solicita una revisión antes de completar la generación del reporte anual.
REQMS-013	Ordenar activos fijos	<ol style="list-style-type: none"> 1. El administrador utiliza la interfaz para ordenar activos: <ul style="list-style-type: none"> • El sistema presenta opciones para ordenar activos según criterios específicos: <ul style="list-style-type: none"> ○ Nombre ○ Categoría ○ Estado ○ Fecha de adquisición ○ Costo inicial 2. El sistema procesa la solicitud de ordenamiento:

		<ul style="list-style-type: none"> Permite seleccionar un criterio desde botones en la interfaz. Cambia dinámicamente el orden de los activos mostrados en la lista principal. <p>3. Validaciones durante el ordenamiento:</p> <ul style="list-style-type: none"> Asegurar que el criterio seleccionado sea válido y esté disponible en el sistema. <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> Una vez realizado el ordenamiento, el sistema actualiza la lista visible en la pantalla según el criterio seleccionado. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Si el usuario intenta ordenar sin seleccionar un criterio válido, se muestra: "Seleccione un criterio de orden válido."
		<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> La funcionalidad de ordenamiento afecta solo la presentación de datos, no modifica la información almacenada en la base de datos. Se debe implementar tanto el orden ascendente como descendente para cada criterio. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> Ordenamientos combinados por múltiples criterios simultáneamente.
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> Normal: El administrador selecciona "Ordenar por nombre" y la lista se reordena alfabéticamente. Alternativo 1: No hay activos registrados.
REQMS-014	Filtrar activos fijos	<ol style="list-style-type: none"> El administrador utiliza la interfaz gráfica para aplicar filtros a los activos: <ul style="list-style-type: none"> El sistema ofrece filtros por: <ul style="list-style-type: none"> Rango de costos iniciales. Periodo de fechas de adquisición. Categoría del activo. Estado del activo (Alta, Depreciado, Baja, Baja por venta). El sistema verifica los datos ingresados para aplicar el filtro: <ul style="list-style-type: none"> Valida que los valores ingresados sean correctos:

	<ul style="list-style-type: none"> ○ Los rangos de costo inicial deben ser positivos y coherentes (mínimo ≤ máximo). ○ Las fechas deben estar en un formato válido (DD-MM-AAAA) y ser cronológicamente consistentes. ○ Categorías seleccionadas deben existir en el catálogo del sistema. ○ Estados deben ser válidos y reconocidos por el sistema. <p>3. Validaciones durante el filtrado:</p> <ul style="list-style-type: none"> • Si no se encuentran activos que cumplan con los criterios seleccionados, el sistema muestra el mensaje: "No se encontraron resultados para los filtros aplicados." • El sistema no permite aplicar filtros simultáneamente que resulten contradictorios (e.g., un rango de fechas incoherente). <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> • El sistema actualiza dinámicamente la vista de activos según los criterios de filtrado seleccionados. • Los activos que no cumplen con los criterios quedan ocultos temporalmente en la interfaz. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Para criterios de búsqueda sin coincidencias: "No se encontraron activos con las características especificadas."
	Actores
	Administrador
	Criterios de Cumplimiento
	Necesario
	Prioridad
	Alta
	Consideraciones
	<ul style="list-style-type: none"> • Los filtros deben ser combinables, permitiendo la aplicación de múltiples criterios simultáneamente. • La funcionalidad no altera los datos almacenados en la base de datos, solo afecta la presentación de la información.
	Alcance No Contemplado
	<ul style="list-style-type: none"> • No se contempla guardar las combinaciones de filtros para uso futuro. • No se permite filtrar por campos personalizados o atributos que no estén definidos previamente.
	Escenarios del Negocio

		<ul style="list-style-type: none"> Normal: El administrador filtra activos por categoría “Equipo de Cómputo” y rango de costo entre S/1,000 y S/10,000; el sistema muestra los resultados correctamente. Alternativo 1: No hay activos que cumplan con los criterios; se muestra un mensaje indicando la falta de coincidencias.
REQMS-015	Buscar activo fijo	<p>1. El administrador utiliza un campo de búsqueda para localizar activos específicos:</p> <ul style="list-style-type: none"> El sistema ofrece un campo de texto para realizar búsquedas por: <ul style="list-style-type: none"> Nombre del activo. Código patrimonial. Fecha de Adquisición <p>2. El sistema procesa el texto ingresado:</p> <ul style="list-style-type: none"> Identifica las palabras clave ingresadas por el administrador. Compara estas palabras clave con los valores almacenados en los campos relevantes de los activos. <p>3. Validaciones durante la búsqueda:</p> <ul style="list-style-type: none"> Si el texto ingresado no coincide con ningún activo, el sistema muestra: “No se encontraron activos que coincidan con el criterio de búsqueda.” Si el campo de búsqueda está vacío y se intenta realizar la operación, se muestra: “Por favor, ingrese un término de búsqueda.” <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> El sistema muestra en la interfaz los activos que coinciden con el término de búsqueda ingresado. Los resultados incluyen solo los activos visibles según los filtros aplicados previamente. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Para búsquedas sin resultados: “No se encontraron coincidencias.” <p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> La búsqueda debe ser insensible a mayúsculas y minúsculas para facilitar coincidencias.

		<ul style="list-style-type: none"> Debe funcionar en conjunto con los filtros aplicados previamente, mostrando solo resultados dentro del subconjunto filtrado.
		<p>Alcance No Contemplado</p> <ul style="list-style-type: none"> No se incluyen búsquedas avanzadas utilizando operadores lógicos (OR, NOT).
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> Normal: El administrador busca el activo "Impresora HP" y el sistema lo muestra en los resultados. Alternativo 1: No se encuentran activos con el texto ingresado; se muestra un mensaje de error.
REQMS-016	Vender activo fijo	<ol style="list-style-type: none"> El Administrador utiliza la interfaz gráfica para iniciar la venta de un activo fijo: <ul style="list-style-type: none"> El sistema muestra un formulario que permite ingresar los datos necesarios para la venta, como el precio de venta, comprador, RUC del comprador, forma de pago y detalles adicionales sobre la venta. Además, se ofrece una vista previa de la factura. El sistema verifica los datos ingresados: <ul style="list-style-type: none"> El formulario de venta requiere que el comprador y su RUC estén especificados, y el precio de venta debe ser positivo. El sistema valida que el estado actual del activo permita su venta (no debe estar en "Baja" o "Baja por venta"). Proceso de venta: <ul style="list-style-type: none"> Si el formulario es válido, el sistema pregunta al Administrador mediante un cuadro de confirmación si desea proceder con la venta del activo. Al confirmar, el sistema cambia el estado del activo a "Baja por venta" y actualiza visualmente el estado en la lista de activos. Se registra un movimiento en el historial de transacciones del activo con los detalles de la venta. Previsualización de factura: <ul style="list-style-type: none"> El sistema genera una vista previa de la factura electrónica con detalles como el nombre del comprador, RUC, fecha de emisión, detalles del activo y forma de pago. El usuario puede revisar la factura antes de completar la venta. Confirmación de la venta: <ul style="list-style-type: none"> Tras finalizar la venta, el sistema muestra un mensaje de éxito: "Activo vendido"

		<p>exitosamente por S/ [precio de venta]", y actualiza el contador de facturas.</p> <p>6. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Si los datos son incorrectos o el activo no cumple las condiciones para la venta, el sistema muestra un mensaje de error específico. <table border="1"> <tr><th>Actores</th></tr> <tr><td>Administrador</td></tr> <tr><th>Criterios de Cumplimiento</th></tr> <tr><td>Necesario</td></tr> <tr><th>Prioridad</th></tr> <tr><td>Alta</td></tr> <tr><th>Consideraciones</th></tr> <tr> <td> <ul style="list-style-type: none"> La venta solo es permitida si el activo tiene valor residual positivo y no está en estado "Baja" o "Baja por venta". La vista previa de la factura permite revisar los detalles antes de realizar la transacción. </td></tr> <tr><th>Alcance No Contemplado</th></tr> <tr> <td> <ul style="list-style-type: none"> No se permite la venta de activos sin un comprador o RUC válido. No se considera la venta de activos automáticamente desde otros sistemas externos. </td></tr> <tr><th>Escenarios del Negocio</th></tr> <tr> <td> <ul style="list-style-type: none"> Normal: El Administrador ingresa correctamente todos los datos del activo para venta, confirma la operación y el sistema registra la venta, actualizando el estado y mostrando un mensaje de confirmación. Alternativo 1: El Administrador ingresa datos incompletos o el activo está en un estado que no permite su venta. El sistema muestra un mensaje de error y no permite continuar </td></tr> </table>	Actores	Administrador	Criterios de Cumplimiento	Necesario	Prioridad	Alta	Consideraciones	<ul style="list-style-type: none"> La venta solo es permitida si el activo tiene valor residual positivo y no está en estado "Baja" o "Baja por venta". La vista previa de la factura permite revisar los detalles antes de realizar la transacción. 	Alcance No Contemplado	<ul style="list-style-type: none"> No se permite la venta de activos sin un comprador o RUC válido. No se considera la venta de activos automáticamente desde otros sistemas externos. 	Escenarios del Negocio	<ul style="list-style-type: none"> Normal: El Administrador ingresa correctamente todos los datos del activo para venta, confirma la operación y el sistema registra la venta, actualizando el estado y mostrando un mensaje de confirmación. Alternativo 1: El Administrador ingresa datos incompletos o el activo está en un estado que no permite su venta. El sistema muestra un mensaje de error y no permite continuar
Actores														
Administrador														
Criterios de Cumplimiento														
Necesario														
Prioridad														
Alta														
Consideraciones														
<ul style="list-style-type: none"> La venta solo es permitida si el activo tiene valor residual positivo y no está en estado "Baja" o "Baja por venta". La vista previa de la factura permite revisar los detalles antes de realizar la transacción. 														
Alcance No Contemplado														
<ul style="list-style-type: none"> No se permite la venta de activos sin un comprador o RUC válido. No se considera la venta de activos automáticamente desde otros sistemas externos. 														
Escenarios del Negocio														
<ul style="list-style-type: none"> Normal: El Administrador ingresa correctamente todos los datos del activo para venta, confirma la operación y el sistema registra la venta, actualizando el estado y mostrando un mensaje de confirmación. Alternativo 1: El Administrador ingresa datos incompletos o el activo está en un estado que no permite su venta. El sistema muestra un mensaje de error y no permite continuar 														
REQMS-017	Dar de baja activo fijo	<p>1. El Administrador utiliza la interfaz gráfica para dar de baja un activo fijo:</p> <ul style="list-style-type: none"> El sistema permite que el administrador seleccione un activo específico para darlo de baja. El administrador debe ingresar el motivo de la baja y el tipo de baja (por ejemplo, "Obsolescencia", "Daño irreparable", etc.). <p>2. El sistema verifica los datos ingresados:</p> <ul style="list-style-type: none"> Verifica que el activo esté en estado "Alta" o "Depreciado" antes de permitir que sea dado de baja. El sistema valida que se haya ingresado un motivo y que el tipo de baja seleccionado sea válido. <p>3. Validaciones durante el proceso de baja:</p>												

		<ul style="list-style-type: none"> Si el activo ya está en estado "Baja" o "Baja por venta", el sistema muestra un mensaje de error y no permite la operación. Si el motivo o tipo de baja son inválidos o están vacíos, el sistema solicita corregir los datos antes de proceder. <p>4. Confirmación de operación:</p> <ul style="list-style-type: none"> Al finalizar el proceso, el sistema cambia el estado del activo a "Baja" y guarda el movimiento en el historial, incluyendo la fecha, el motivo y el tipo de baja. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Si el activo no puede darse de baja debido a su estado actual, el sistema muestra un mensaje: "El activo ya se encuentra en estado de baja o fue vendido." En caso de éxito, se muestra el mensaje: "Activo dado de baja exitosamente: [nombre del activo], motivo: [motivo de la baja]."
		Actores
		Administrador
		Criterios de Cumplimiento
		Necesario
		Prioridad
		Alta
		Consideraciones
		<ul style="list-style-type: none"> El sistema debe asegurar que los activos dados de baja no puedan ser reactivados o utilizados en operaciones futuras. La operación de baja debe estar respaldada en el historial con todos los detalles relevantes.
		Alcance No Contemplado
		<ul style="list-style-type: none"> No contempla la reactivación de activos dados de baja. No incluye la posibilidad de modificar o eliminar el registro de baja una vez completado.
		Escenarios del Negocio
		<ul style="list-style-type: none"> Normal: El administrador selecciona un activo en estado "Alta" o "Depreciado", ingresa el motivo y tipo de baja, y el sistema registra la baja y cambia el estado del activo. Alternativo 1: El activo ya está en estado "Baja". El sistema muestra un mensaje indicando que el activo no puede ser dado de baja nuevamente. Alternativo 2: El administrador no ingresa un motivo de baja o elige un tipo inválido. El sistema muestra un mensaje de error solicitando que complete los datos antes de proceder.

REQMS-018	Exportar reporte de depreciación	<p>1. El sistema genera reportes de depreciación en tiempo real:</p> <ul style="list-style-type: none"> Los reportes se presentan directamente en una tabla dentro de la interfaz gráfica, permitiendo al usuario visualizar los datos de depreciación en tiempo real, ya sea en modo simulación o con la base de datos real. La tabla muestra información detallada sobre cada activo, como el valor inicial, la depreciación mensual y acumulada, y el valor neto contable. <p>2. Exportación a Excel:</p> <ul style="list-style-type: none"> El sistema incluye una opción para exportar el reporte visualizado en la tabla directamente a un archivo Excel, ya formateado para facilitar su análisis externo. Los valores exportados incluyen la depreciación mensual, la acumulada, y el valor neto de cada activo, con los mismos filtros aplicados en la visualización de la tabla. <p>3. Integración con otras funcionalidades:</p> <ul style="list-style-type: none"> Los reportes reflejan los cálculos de depreciación en tiempo real para activos en la base de datos real o en simulación. Los datos presentados son coherentes con los filtros aplicados, como el estado del activo (Alta, Baja, Depreciado) o su categoría. <p>4. Validaciones:</p> <ul style="list-style-type: none"> El sistema valida que existan activos que cumplan con los filtros seleccionados antes de generar la tabla. Si no se encuentran activos, se muestra un mensaje emergente: "No se encontraron activos que cumplan con los criterios seleccionados". <p>5. Notificaciones y mensajes:</p> <ul style="list-style-type: none"> Al exportar el reporte, el sistema confirma la exportación exitosa con un cuadro de diálogo: "Reporte exportado exitosamente a Excel." <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="background-color: #002B36; color: white; padding: 2px;">Actores</td></tr> <tr> <td style="background-color: #F0F0F0; padding: 2px;">Administrador</td></tr> <tr> <td style="background-color: #002B36; color: white; padding: 2px;">Criterios de Cumplimiento</td></tr> <tr> <td style="background-color: #F0F0F0; padding: 2px;">Necesario</td></tr> <tr> <td style="background-color: #002B36; color: white; padding: 2px;">Prioridad</td></tr> <tr> <td style="background-color: #F0F0F0; padding: 2px;">Alta</td></tr> </table>	Actores	Administrador	Criterios de Cumplimiento	Necesario	Prioridad	Alta
Actores								
Administrador								
Criterios de Cumplimiento								
Necesario								
Prioridad								
Alta								

		<p>Consideraciones</p> <ul style="list-style-type: none"> • El reporte se genera y visualiza en tiempo real dentro de la interfaz gráfica. • Los reportes pueden ser exportados a Excel ya formateados, facilitando su análisis externo. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No se contempla la exportación a otros formatos como PDF o CSV. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El usuario visualiza el reporte de depreciación en tiempo real dentro de la tabla en la interfaz gráfica, aplicando filtros por estado y categoría. Posteriormente, decide exportar el reporte a Excel. • Alternativo 1: El usuario intenta generar un reporte, pero no hay activos que cumplan con los filtros seleccionados. El sistema muestra un mensaje informando que no se encontraron activos que coincidan. • Alternativo 2: El usuario selecciona los activos y genera el reporte en tiempo real, pero decide no exportarlo a Excel. Los cambios en la tabla se mantienen dentro del sistema para futuras consultas.
REQMS-019	Proyectar depreciación futura	<ol style="list-style-type: none"> 1. El Administrador utiliza la interfaz gráfica para proyectar la depreciación futura de un activo fijo: <ul style="list-style-type: none"> • La proyección calcula el valor de depreciación acumulada para cada año o mes en el período definido, según la vida útil del activo. 2. El sistema verifica los datos del activo seleccionado: <ul style="list-style-type: none"> • Verifica que el activo sea depreciable y esté en estado "Alta". • Confirma que la vida útil restante del activo sea suficiente para realizar la proyección. 3. Validaciones durante la proyección: <ul style="list-style-type: none"> • Si el activo ya alcanzó el final de su vida útil, el sistema muestra un mensaje indicando que no es posible proyectar más depreciación. • Si el período solicitado excede la vida útil restante, el sistema ajusta la proyección hasta el fin de la vida útil del activo. 4. Confirmación de operación: <ul style="list-style-type: none"> • Al finalizar el cálculo, el sistema muestra en pantalla la depreciación proyectada para cada período solicitado, mostrando la fecha y el valor estimado de depreciación acumulada. 5. Mensajes y notificaciones: <ul style="list-style-type: none"> • Si el activo ha alcanzado su vida útil y no se puede proyectar más depreciación, el sistema muestra un mensaje: "El activo ha

		<p>alcanzado el fin de su vida útil; no se puede realizar una proyección futura."</p> <p>Actores Administrador</p> <p>Criterios de Cumplimiento Prioridad</p> <p>Prioridad Media</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • El sistema debe proyectar la depreciación futura con base en el método de línea recta y ajustar la proyección a la vida útil restante. • La proyección solo es válida para activos que continúan en estado "Alta". <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No incluye la proyección de depreciación para activos no depreciables o que ya están en estado "Baja". • No contempla ajustes en la proyección si la vida útil o el método de depreciación cambia posteriormente. <p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador selecciona un activo con vida útil restante; el sistema calcula y muestra la depreciación futura acumulada para cada año. • Alternativo 1: El activo ya ha alcanzado su vida útil total. El sistema muestra un mensaje indicando que no es posible proyectar la depreciación.
REQMS-020	Generar reporte de inventario	<ol style="list-style-type: none"> 1. El administrador genera un reporte consolidado del inventario: <ul style="list-style-type: none"> • El sistema ofrece un menú para generar reportes y exportarlo en PNG. 2. El sistema recopila y organiza los datos para el reporte: <ul style="list-style-type: none"> • Incluye campos básicos como: <ul style="list-style-type: none"> ○ Nombre del activo. ○ Categoría. ○ Fecha de adquisición. ○ Estado. ○ Valor residual. • Añade totales y resúmenes según corresponda (por ejemplo, costo total inicial, valor residual total). 3. Validaciones durante la generación del reporte: <ul style="list-style-type: none"> • Si no hay activos registrados o filtrados, el sistema muestra: "No hay datos disponibles para generar el reporte." 4. Confirmación de operación:

		<ul style="list-style-type: none"> • Se ofrece la opción de previsualizar el reporte antes de descargarlo. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Para errores (e.g., fallas técnicas): "Error al generar el reporte. Intente nuevamente."
		<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> • Los datos deben estar ordenados y presentados de manera clara en el formato de salida. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • Generación de reportes con análisis estadísticos avanzados (e.g., proyecciones de depreciación).
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador genera un reporte del inventario completo y lo descarga en formato PNG. • Alternativo 1: No hay activos registrados; se muestra un mensaje indicando la imposibilidad de generar el reporte.
REQMS-021	Gestionar la revaluación de activos fijos	<ol style="list-style-type: none"> 1. El administrador inicia una revaluación de activos desde la interfaz gráfica: <ul style="list-style-type: none"> • Selecciona un activo específico desde el listado principal. • Proporciona los nuevos valores del activo, como: <ul style="list-style-type: none"> ○ Valor revaluado. ○ Motivo de la revaluación. 2. El sistema calcula los valores actualizados: <ul style="list-style-type: none"> • Recalcula la depreciación mensual y acumulada con base en el nuevo valor. • Calcula el impacto porcentual de la revaluación respecto al valor anterior. 3. Validaciones durante la revaluación: <ul style="list-style-type: none"> • Asegura que el valor revaluado no sea menor al valor residual actual del activo. • Verifica que el motivo de revaluación esté presente y sea válido. 4. Confirmación de operación: <ul style="list-style-type: none"> • Una vez completada la revaluación, el sistema muestra un mensaje: "Revaluación completada exitosamente para el activo [nombre]."

		<ul style="list-style-type: none"> Los valores recalculados se actualizan en la base de datos y se reflejan en la interfaz. <p>5. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> Para errores: "El valor revaluado no puede ser menor al valor residual actual." Para éxito: "Revaluación registrada correctamente."
		<p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p> <p>Alta</p> <p>Consideraciones</p> <ul style="list-style-type: none"> El sistema debe recalcular automáticamente los valores relacionados con el activo (depreciación mensual, acumulada, y valor residual) después de la revaluación. Se debe registrar un movimiento de revaluación en el historial del activo, incluyendo detalles como el valor anterior, valor revaluado, y motivo de la revaluación.
		<p>Alcance No Contemplado</p> <ul style="list-style-type: none"> No se permite revaluar activos en estado "Baja" o "Baja por venta". No se consideran revaluaciones múltiples dentro del mismo periodo para un mismo activo. No se incluyen métodos avanzados de valoración o modelos externos para determinar el nuevo valor del activo (solo valores ingresados manualmente por el administrador).
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> Normal: El administrador selecciona un activo elegible, ingresa un nuevo valor, y el sistema recalcula los valores relacionados, registrando correctamente el movimiento de revaluación. Alternativo 1: El administrador intenta revaluar un activo en estado no elegible (e.g., "Baja"); el sistema muestra un mensaje de error. Alternativo 2: El nuevo valor revaluado es menor al valor residual actual; el sistema no permite la revaluación y muestra un mensaje indicando la inconsistencia.
REQMS-022	Gestionar la renovación de activos	<p>1. El administrador inicia la renovación de un activo desde la interfaz gráfica:</p> <ul style="list-style-type: none"> Selecciona un activo en estado "Depreciado", "Baja", o "Baja por venta". Proporciona detalles de la renovación, como: <ul style="list-style-type: none"> Motivo de la renovación. Nuevo código patrimonial (generado automáticamente).

	<p>2. El sistema verifica la información del activo antes de la renovación:</p> <ul style="list-style-type: none"> • Confirma que el activo esté en un estado renovable. • Valida que el activo no haya sido renovado previamente en el periodo actual. <p>3. El sistema realiza las operaciones necesarias para la renovación:</p> <ul style="list-style-type: none"> • Actualiza el estado del activo original a "Renovado". • Crea un nuevo registro de activo, duplicando los datos principales (nombre, categoría, costo inicial) y asignando: <ul style="list-style-type: none"> ◦ Nueva fecha de adquisición (fecha actual). ◦ Nueva vida útil calculada según la categoría. ◦ Nuevo código patrimonial. <p>4. Validaciones durante la renovación:</p> <ul style="list-style-type: none"> • Asegurar que el activo original cumpla con los criterios de renovación. • Prevenir errores en la asignación del nuevo código patrimonial. • Validar que los datos del nuevo activo sean consistentes y completos antes de guardar. <p>5. Confirmación de operación:</p> <ul style="list-style-type: none"> • El sistema muestra un mensaje: "Renovación completada exitosamente. El activo [nombre] ahora está disponible como un nuevo registro con el código [código patrimonial]." <p>6. Mensajes y notificaciones:</p> <ul style="list-style-type: none"> • Para errores: "El activo no es elegible para renovación. Revise el estado y los criterios." • Para éxito: "Renovación registrada correctamente."
Actores	
Administrador	
Criterios de Cumplimiento	
Necesario	
Prioridad	
Alta	
Consideraciones	
<ul style="list-style-type: none"> • El nuevo activo debe heredar los datos relevantes del activo original, como la categoría y el nombre. • La operación de renovación genera un registro en el historial de movimientos. 	

		<p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No se contempla la renovación de activos en estado "Alta". • No se incluye la asignación de vida útil personalizada para renovaciones.
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El administrador renueva un activo "Depreciado" y el sistema crea un nuevo registro correctamente. • Alternativo 1: El activo no cumple los criterios de renovación; el sistema muestra un mensaje de error.
REQMS-023	Ubicar activos	<ol style="list-style-type: none"> 1. El administrador busca la ubicación de un activo específico: <ul style="list-style-type: none"> • Selecciona un activo desde la lista principal. • Consulta detalles de la ubicación, como: <ul style="list-style-type: none"> ◦ Nombre de la sede. ◦ Dirección completa. 2. El sistema presenta la información de ubicación: <ul style="list-style-type: none"> • Extrae los datos de la sede asociada al activo y los muestra en un panel emergente. • Si el activo no tiene una sede asociada, muestra un mensaje: "El activo [nombre] no tiene ubicación registrada." 3. Validaciones durante la consulta: <ul style="list-style-type: none"> • Verificar que el activo tenga una sede asignada. • Asegurar que la información de la sede esté completa (nombre y dirección). 4. Confirmación de operación: <ul style="list-style-type: none"> • El sistema muestra la ubicación del activo con el mensaje: "Activo ubicado en [nombre de la sede], dirección: [dirección]." 5. Mensajes y notificaciones: <ul style="list-style-type: none"> • Si no se encuentra la ubicación: "No se pudo determinar la ubicación del activo seleccionado." • Para éxito: "La ubicación del activo [nombre] es: [nombre de la sede], [dirección]."
		Actores
		Administrador
		Criterios de Cumplimiento
		Necesario
		Prioridad
		Media
		Consideraciones

		<ul style="list-style-type: none"> • La ubicación debe estar actualizada y reflejar cambios realizados en las sedes. • Esta funcionalidad se limita a activos con una sede asignada explícitamente. <p>Alcance No Contemplado</p> <ul style="list-style-type: none"> • No incluye la localización de activos en tiempo real o mediante geolocalización.
		<p>Escenarios del Negocio</p> <ul style="list-style-type: none"> • Normal: El sistema muestra correctamente la ubicación de un activo con sede asignada. • Alternativo 1: El activo no tiene ubicación registrada; se muestra un mensaje de error.
REQMS-024	Analizar activos en baja	<ol style="list-style-type: none"> 1. El administrador solicita un análisis detallado de activos en baja: <ul style="list-style-type: none"> • Selecciona el módulo de análisis desde la interfaz principal. • Elige el alcance del análisis, como: <ul style="list-style-type: none"> ◦ Activos en baja general. ◦ Activos dados de baja por venta. 2. El sistema recopila y procesa los datos: <ul style="list-style-type: none"> • Calcula métricas clave, como: <ul style="list-style-type: none"> ◦ Pérdida bruta (costo inicial menos valor residual). ◦ Monto recuperado de ventas. ◦ Pérdida neta (pérdida bruta menos monto recuperado). • Genera gráficos interactivos (e.g., barras) que muestran pérdidas por categoría. 3. Validaciones durante el análisis: <ul style="list-style-type: none"> • Asegurar que los datos de los activos estén completos y actualizados. • Excluir activos renovados o que no cumplan con los criterios de baja. 4. Confirmación de operación: <ul style="list-style-type: none"> • El sistema presenta un reporte visual en la interfaz con: <ul style="list-style-type: none"> ◦ Tabla detallada de activos en baja. ◦ Gráficos que resaltan las categorías con mayores pérdidas. 5. Mensajes y notificaciones: <ul style="list-style-type: none"> • Para ausencia de datos: "No se encontraron activos en baja para analizar." • Para éxito: "El análisis de activos en baja ha sido generado correctamente." <p>Actores</p> <p>Administrador</p> <p>Criterios de Cumplimiento</p> <p>Necesario</p> <p>Prioridad</p>

		Alta Consideraciones
		<ul style="list-style-type: none"> • El análisis debe incluir tanto datos tabulares como gráficos para facilitar la interpretación. • Los datos deben ser exportables (PNG) para informes externos.
		Alcance No Contemplado
		<ul style="list-style-type: none"> • No se consideran análisis predictivos basados en tendencias históricas.

	Escenarios del Negocio
	<ul style="list-style-type: none"> • Normal: El sistema genera un análisis de activos en baja con métricas y gráficos claros. • Alternativo 1: No hay activos en baja; el sistema muestra un mensaje indicando la ausencia de datos.

8 PRINCIPALES PANTALLAS

REQMS-001: Controlar acceso basado en roles



REQMS-002: Registrar activos fijos

Valorium

Lista de Activos Registrados

ID	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisici.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
48	2024-0048	Servidor HP	Mobiliario	Baja	13-10-2000	10	10916.42	99.97	10916.42	0.00
49	2024-0049	Auto Hyundai	Equipo de Computo	Depreciado	13-01-2023	4	10429.00	217.27	4779.96	5649.04
50	2024-0050	Servidor HP	Terreno	Baja	14-12-2011	0	9405.77	0.00	0.00	9405.77
51	2024-0051	Camioneta Ford	Vehículo	Baja	17-06-2011	5	5997.44	99.95	4198.28	1799.95
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	5	1593.41	26.58	0.00	1593.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	48.31	5557.23	0.00
54	2024-0054	Servidor HP	Mobiliario	Alta	01-11-2012	10	3698.82	30.98	3606.82	0.00
55	2024-0055	Terreno Urbano	Equipo de Laborat.	Baja	17-03-2001	5	5333.83	89.19	5333.63	0.00
56	2024-0056	Monitor Samsung	Equipo de Laborat.	Depreciado	28-04-2012	5	3050.05	59.83	3050.05	0.00
58	2024-0058	Auto Hyundai	Mobiliario	Baja	08-04-2013	10	5121.26	42.68	5121.26	0.00
59	2024-0059	Laboratorio Químico	Terreno	Alta	19-03-2000	0	3959.11	0.00	0.00	3959.11
60	2024-0060	Terreno Urbano	Equipo de Computo	Baja	21-03-2017	4	1543.03	49.00	1443.03	0.00
61	2024-0061	Terreno Urbano	Terreno	Baja	09-11-2020	0	2251.96	0.00	0.00	2251.96
62	2024-0062	Escritorio de Madera	Vehículo	Baja	25-03-2000	5	8246.33	137.44	8246.33	0.00
63	2024-0063	Auto Hyundai	Equipo de Computo	Baja	12-05-2010	4	5644.41	117.59	5644.41	0.00
64	2024-0064	Auto Toyota	Terreno	Alta	03-04-2018	0	5424.00	0.00	0.00	5424.00
65	2024-0065	Terreno Rústico	Mobiliario	Alta	03-12-2006	10	7277.17	69.64	7277.17	0.00
66	2024-0066	Camioneta Ford	Vehículo	Alta	03-04-2008	5	6114.44	101.91	6114.44	0.00
67	2024-0067	Computadora Dell	Mobiliario	Depreciado	02-01-2002	10	8661.99	72.18	8661.99	0.00
68	2024-0068	Silla Ergonómica	Mobiliario	Alta	14-06-2023	10	1901.78	15.88	253.97	1648.19
69	2024-0069	Laboratorio Químico	Equipo de Computo	Baja	11-03-2000	4	1634.41	28.52	1629.51	0.00
70	2024-0070	Servidor HP	Equipo de Computo	Depreciado	25-12-2024	4	5629.69	117.29	0.00	5629.69
71	2024-0071	Microscopio Olymp.	Terreno	Baja por venta	24-06-2006	0	10013.61	0.00	0.00	10013.61
72	2024-0072	Auto Toyota	Vehículo	Baja	21-01-2005	5	3488.61	68.14	3488.61	0.00
73	2024-0073	Silla Ergonómica	Terreno	Alta por venta	04-04-2016	0	3189.51	0.00	0.00	3189.51
74	2024-0074	Microscopio Olymp.	Terreno	Alta	05-04-2010	5	5717.51	0.00	0.00	5717.51
75	2024-0075	Microscopio Olymp.	Vehículo	Baja	19-03-2007	5	10792.37	179.87	10792.37	0.00
76	2024-0076	Terreno Urbano	Equipo de Laborat.	Depreciado	02-05-2007	5	7983.96	133.07	7983.96	0.00
77	2024-0077	Microscopio Olymp.	Mobiliario	Alta	25-02-2000	10	5129.73	242.45	5129.73	0.00
78	2024-0078	Terreno Urbano	Terreno	Alta por venta	03-04-2007	0	9373.94	0.00	0.00	9373.94
79	2024-0079	Mesa de Reuniones	Equipo de Computo	Baja	04-10-2016	4	5776.93	120.35	5776.93	0.00
80	2024-0080	Servidor HP	Equipo de Laborat.	Baja por venta	28-05-2005	5	6552.73	109.21	6552.73	0.00
81	2024-0081	Computadora Dell	Equipo de Computo	Alta	02-08-2003	4	10201.19	202.52	10201.19	0.00
82	2024-0082	Monitor Samsung	Equipo de Laborat.	Baja por venta	20-06-2000	5	8898.11	148.17	8898.11	0.00
83	2024-0083	Auto Hyundai	Mobiliario	Baja por venta	11-05-2013	10	10321.47	66.01	10321.47	0.00
84	2024-0084	Computadora Dell	Mobiliario	Depreciado	09-09-2011	10	4237.65	35.31	4237.65	0.00
85	2024-0085	Servidor HP	Equipo de Laborat.	Baja	07-06-2009	5	1028.71	17.40	1028.71	0.00
86	2024-0086	Terreno Rústico	Mobiliario	Alta	12-06-2001	10	2352.72	239.50	2352.72	0.00
87	2024-0087	Mesa de Reuniones	Terreno	Alta	15-12-2011	0	1980.39	0.00	0.00	1980.39
88	2024-0088	Monitor Samsung	Equipo de Laborat.	Baja	17-03-2022	5	8788.38	146.47	4887.14	4101.24
89	2024-0089	Servidor HP	Mobiliario	Depreciado	13-01-2007	10	8012.83	88.77	8012.83	0.00
90	2024-0090	Mesa de Reuniones	Mobiliario	Alta	24-06-2000	10	9710.13	99.99	9710.13	0.00
91	2024-0091	Terreno Urbano	Equipo de Laborat.	Baja	18-09-2005	5	3570.77	69.51	3570.77	0.00
92	2024-0092	Terreno Rústico	Mobiliario	Alta	20-03-2024	10	1635.85	62.95	370.64	5983.21
93	2024-0093	Servidor HP	Vehículo	Baja por venta	16-01-2004	5	8688.20	144.80	8688.20	0.00
94	2024-0094	Escritorio de Madera	Terreno	Baja por venta	06-02-2024	0	2815.90	0.00	0.00	2815.90
95	2024-0095	Placa de Agregado	Mobiliario	Alta	19-06-2002	10	5451.59	142.11	5451.60	0.00
96	2024-0096	Auto Hyundai	Terreno	Baja por venta	10-06-2005	0	10341.29	0.00	0.00	10341.29
97	2024-0097	Auto Toyota	Equipo de Laborat.	Baja por venta	05-03-2009	5	3542.26	59.04	3542.26	0.00
98	2024-0098	Computadora Dell	Equipo de Computo	Baja por venta	01-12-2006	4	2303.24	47.98	2303.24	0.00
99	2024-0099	Microscopio Olymp.	Terreno	Alta por venta	11-06-2017	5	3083.81	14.47	3088.01	0.00
100	2024-0100	Monitor Samsung	Terreno	Alta por venta	13-06-2019	0	5668.83	0.00	0.00	5669.82
101	2022-0001	TEST	Equipo de Computo	Baja	02-12-2022	4	800.00	12.50	297.50	312.50

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 101 | Total de Costo Inicial: \$/598.42 K | Total de Depreciación Mensual: \$/7.31 K | Total de Depreciación Acumulada: \$/432.60 K | Total de Valor Residual: \$/165.81 K

REQMS-003: Calcular la vida útil según categoría

```
4 usages  ± Fernando Injoque
public static int determinarVidaUtil(String categoria) {
    switch (categoria) {
        case "Equipo de Computo":
            return 4;
        case "Mobiliario":
            return 10;
        case "Equipo de Laboratorio":
            return 5;
        case "Vehiculo":
            return 5;
        case "Terreno":
            return 0;
        default:
            return 1;
    }
}
```

REQMS-004: Aplicar el método de depreciación por línea recta

```
② DepreciacionLineaRecta.java ×  
1 package modelo;  
2  
3     2 usages ▾ Fernando Injoque  
4 public class DepreciacionLineaRecta extends Depreciacion {  
5  
6     1 usage ▾ Fernando Injoque  
7     public DepreciacionLineaRecta(double costoInicial, int vidaUtil) { super(costoInicial, vidaUtil); }  
8  
9     1 usage ▾ Fernando Injoque  
10    @Override  
11    public double calcularDepreciacion() {  
12        return costoInicial / vidaUtil;  
13    }  
14}
```

REQMS-005: Calcular depreciación mensual

```
15 usages ▾ Fernando Injoque  
@Override  
public double calcularDepreciacionMensual() {  
    if (categoria.equals("Terreno")) {  
        return 0.0;  
    }  
    DepreciacionLineaRecta depreciacion = new DepreciacionLineaRecta(costoInicial, vidaUtil);  
    return depreciacion.calcularDepreciacion() / 12;  
}
```

REQMS-006: Generar depreciación acumulada mensual

```
13 usages ▾ Fernando Injoque  
public double calcularDepreciacionAcumulada() {  
    if (vidaUtil == 0) {  
        return 0.0;  
    }  
  
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");  
    LocalDate fechaInicioDepreciacion = LocalDate.parse(this.fechaInicioDepreciacion, formatter);  
    LocalDate fechaActual = LocalDate.now();  
  
    long diasTranscurridos = ChronoUnit.DAYS.between(fechaInicioDepreciacion, fechaActual);  
  
    if (diasTranscurridos < 30) {  
        return 0.0;  
    }  
  
    long mesesTranscurridos = diasTranscurridos / 30;  
  
    long mesesVidaUtil = (long) vidaUtil * 12;  
    if (mesesTranscurridos > mesesVidaUtil) {  
        mesesTranscurridos = mesesVidaUtil;  
    }  
  
    return mesesTranscurridos * (costoInicial / (double) mesesVidaUtil);  
}
```

REQMS-007: Generar depreciación acumulada anual

Proyección de Depreciación para 'TEST'					
Mes	Año	Depreciación Mensual	Depreciación Acumulada	Valor en Libros	
enero	2023	S/ 12.50	S/ 12.50	S/ 587.50	
febrero	2023	S/ 12.50	S/ 25.00	S/ 575.00	
marzo	2023	S/ 12.50	S/ 37.50	S/ 562.50	
abril	2023	S/ 12.50	S/ 50.00	S/ 550.00	
mayo	2023	S/ 12.50	S/ 62.50	S/ 537.50	
junio	2023	S/ 12.50	S/ 75.00	S/ 525.00	
julio	2023	S/ 12.50	S/ 87.50	S/ 512.50	
agosto	2023	S/ 12.50	S/ 100.00	S/ 500.00	
septiembre	2023	S/ 12.50	S/ 112.50	S/ 487.50	
octubre	2023	S/ 12.50	S/ 125.00	S/ 475.00	
noviembre	2023	S/ 12.50	S/ 137.50	S/ 462.50	
diciembre	2023	S/ 12.50	S/ 150.00	S/ 450.00	
enero	2024	S/ 12.50	S/ 162.50	S/ 437.50	
febrero	2024	S/ 12.50	S/ 175.00	S/ 425.00	
marzo	2024	S/ 12.50	S/ 187.50	S/ 412.50	
abril	2024	S/ 12.50	S/ 200.00	S/ 400.00	
mayo	2024	S/ 12.50	S/ 212.50	S/ 387.50	
junio	2024	S/ 12.50	S/ 225.00	S/ 375.00	
julio	2024	S/ 12.50	S/ 237.50	S/ 362.50	
agosto	2024	S/ 12.50	S/ 250.00	S/ 350.00	
septiembre	2024	S/ 12.50	S/ 262.50	S/ 337.50	
octubre	2024	S/ 12.50	S/ 275.00	S/ 325.00	
noviembre	2024	S/ 12.50	S/ 287.50	S/ 312.50	
diciembre	2024	S/ 12.50	S/ 300.00	S/ 300.00	
enero	2025	S/ 12.50	S/ 312.50	S/ 287.50	

[Exportar a Excel](#) [Cerrar](#)

REQMS-008: Calcular valor residual

```
18 usages  ± Fernando Injoque
@Override
public double calcularValorResidual() {
    if (vidaUtil == 0) {
        return costoInicial;
    }

    double depreciacionAcumulada = calcularDepreciacionAcumulada();
    return Math.max(0, costoInicial - depreciacionAcumulada);
}
```

REQMS-009: Notificar para la revisión de activos

Valorium



Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisici.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
51	2024-0051	Camioneta Ford	Vehículo	Baja	17-05-2021	5	5997.54	99.96	4198.28	1799.26
52	2024-0052	Computadora Dell	Móvil/ano	Depreciado	17-12-2024	5	1593.41	26.56	0.00	1593.41
53	2024-0053	Laboratorio Químico	Móvil/ano	Depreciado	03-04-2003	10	5557.23	45.31	5557.23	0.00
54	2024-0054	Silla Ergonómica	Vehículo	Baja	01-01-2019	10	3605.00	36.00	3605.00	0.00
55	2024-0055	Terrero Urbano	Equipo de Laborat.	Alta	17-03-2001	10	5333.63	68.89	5333.63	0.00
56	2024-0056	Microscopio Olymp.	Terreno	Baja por venta	02-10-2005	0	3583.95	0.00	0.00	3583.95
57	2024-0057	Monitor Samsung	Equipo de Laborat.	Depreciado	28-04-2012	5	3050.05	50.83	3050.05	0.00
58	2024-0058	Auto Hyundai	Vehículo	Baja	08-04-2013	10	5121.28	42.28	5121.28	0.00
59	2024-0059	Laboratorio Químico	Terreno	Baja	10-03-2007	0	3851.11	0.00	0.00	3859.11
60	2024-0060	Terreno Urbano	Equipo de Computo	Alta	21-03-2017	4	1344.03	28.00	1344.03	0.00
61	2024-0061	Terreno Urbano	Terreno	Baja	09-11-2020	0	2251.36	0.00	0.00	2251.36
62	2024-0062	Equipo de Madera	Vehículo	Baja	25-03-2000	5	8246.33	19.44	8246.33	0.00
63	2024-0063	Auto Toyota	Equipo de Computo	Alta	12-05-2010	4	5654.41	117.59	5654.41	0.00
64	2024-0064	Auto Toyota	Terreno	Alta	03-04-2018	0	5435.18	0.00	0.00	5435.18
65	2024-0065	Terrero Rústico	Móvil/ano	Alta	03-12-2006	10	7277.17	60.64	7277.17	0.00
66	2024-0066	Camioneta Ford	Vehículo	Alta	03-04-2008	5	6114.44	101.91	6114.44	0.00
67	2024-0067	Computadora Dell	Móvil/ano	Depreciado	02-04-2010	10	8697.99	98.99	8697.99	0.00
68	2024-0068	Silla Ergonómica	Móvil/ano	Alta	14-06-2023	10	1991.76	15.85	1991.76	0.00
69	2024-0069	Laboratorio Químico	Equipo de Computo	Baja	11-03-2002	4	1839.51	38.32	1839.51	0.00
70	2024-0070	Senidor HP	Equipo de Computo	Depreciado	25-12-2024	4	5629.69	117.29	0.00	5629.69
71	2024-0071	Microscopio Olymp.	Terreno	Baja por venta	24-06-2006	0	10013.61	0.00	0.00	10013.61
72	2024-0072	Auto Toyota	Equipo de Computo	Alta	21-01-2010	5	3485.68	62.14	3486.61	0.00
73	2024-0073	Silla Ergonómica	Terreno	Baja por venta	04-07-2016	0	3180.51	0.00	0.00	3180.51
74	2024-0074	Microscopio Olymp.	Terreno	Alta	05-04-2010	0	5717.51	0.00	0.00	5717.51
75	2024-0075	Microscopio Olymp.	Vehículo	Baja	19-03-2007	5	10792.37	179.87	10792.37	0.00
76	2024-0076	Terrero Urbano	Equipo de Laborat.	Depreciado	02-04-2010	5	7983.07	179.07	7983.07	0.00
77	2024-0077	Microscopio Olymp.	Móvil/ano	Depreciado	25-02-2000	10	5129.73	42.75	5129.73	0.00
78	2024-0078	Terrero Rústico	Terreno	Baja por venta	06-07-2007	0	9373.94	0.00	0.00	9373.94
79	2024-0079	Mesa de Reuniones	Equipo de Computo	Baja	04-10-2016	4	5776.93	120.35	5776.93	0.00
80	2024-0080	Senidor HP	Equipo de Computo	Depreciado	28-06-2009	5	6557.31	105.21	6557.31	0.00
81	2024-0081	Computadora Dell	Equipo de Computo	Alta	15-06-2003	4	10201.19	213.52	10201.19	0.00
82	2024-0082	Microscopio Olymp.	Equipo de Laborat.	Baja por venta	28-10-2003	5	8890.11	148.17	8890.11	0.00
83	2024-0083	Auto Hyundai	Móvil/ano	Baja por venta	11-09-2013	10	10321.47	86.01	10321.47	0.00
84	2024-0084	Senidor HP	Móvil/ano	Depreciado	09-09-2011	10	4237.85	35.31	4237.85	0.00
85	2024-0085	Senidor HP	Equipo de Laborat.	Alta	07-01-2019	5	1027.11	17.15	1027.11	0.00
86	2024-0086	Terrero Urbano	Móvil/ano	Baja	12-10-2001	10	2339.72	19.50	2339.72	0.00
87	2024-0087	Mesa de Reuniones	Terreno	Baja	15-12-2011	0	1980.39	0.00	0.00	1980.39
88	2024-0088	Monitor Samsung	Equipo de Laborat.	Depreciado	17-03-2022	5	8783.38	146.47	8687.14	4101.24
89	2024-0089	Senidor HP	Equipo de Computo	Depreciado	01-07-2007	10	8013.77	207.77	8013.77	0.00
90	2024-0090	Mesa de Reuniones	Móvil/ano	Baja por venta	24-09-2005	10	9718.43	80.99	9718.43	0.00
91	2024-0091	Mesa de Reuniones	Equipo de Laborat.	Alta	19-05-2005	5	3570.77	59.51	3570.77	0.00
92	2024-0092	Terrero Rústico	Móvil/ano	Alta	20-03-2024	10	6353.85	82.95	370.64	5983.21
93	2024-0093	Senidor HP	Vehículo	Baja por venta	06-01-2010	5	8868.00	186.80	8868.00	0.00
94	2024-0094	Escritorio de Madera	Terreno	Baja por venta	06-01-2024	0	2815.00	0.00	0.00	2815.00
95	2024-0095	Parcela Agrícola	Móvil/ano	Baja	06-15-2002	10	5412.60	45.11	5412.60	0.00
96	2024-0096	Auto Hyundai	Terreno	Baja por venta	10-06-2005	0	10341.29	0.00	0.00	10341.29
97	2024-0097	Auto Toyota	Equipo de Laborat.	Baja por venta	05-03-2009	5	3542.20	89.33	3542.20	0.00
98	2024-0098	Computadora Dell	Equipo de Computo	Alta	20-01-2019	5	2303.34	48.67	2303.34	0.00
99	2024-0099	Microscopio Olymp.	Vehículo	Baja por venta	11-10-2017	5	3088.01	16.00	3088.01	0.00
100	2024-0100	Monitor Samsung	Terreno	Alta	13-06-2019	0	5690.82	0.00	0.00	5690.82
101	2022-0001	TEST 1	Equipo de Computo	Depreciado	02-12-2022	4	600.00	0.00	0.00	600.00
102	2024-0101	TEST NOTIFICADO	Equipo de Computo	Depreciado	01-06-2024	4	0.00	0.00	0.00	0.00
103	2024-0102	TEST INSTANT	Equipo de Computo	Depreciado	03-02-2020	4	40.00	0.00	0.00	40.00
104	2021-0001	TEST 2	Equipo de Computo	Alta	09-01-2021	1	90.00	1.00	90.00	0.00

Notificación de Activo
El activo 'TEST 2' se depreciará en menos de 6 meses.

Recordar Listo

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$598.66 K | Total de Depreciación Mensual: \$/7.31 K | Total de Depreciación Acumulada: \$/432.84 K | Total de Valor Residual: \$/165.81 K

REQMS-010: Mostrar el estado de un activo fijo

Valorium



Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisici.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
1	2024-0001	Laboratorio Químico	Móvil/ano	Baja por venta	15-10-2012	5	4414.17	73.57	4414.17	0.00
2	2024-0002	Mesa de Reuniones	Móvil/ano	Baja	04-02-2015	10	3215.01	28.79	3161.43	53.58
3	2024-0003	Microscopio Olymp.	Vehículo	Baja	20-01-2001	10	5751.08	147.93	5751.08	0.00
4	2024-0004	Monitor Samsung	Vehículo	Baja	20-01-2019	9	9881.93	100.00	9881.93	0.00
5	2024-0005	Parcela Agrícola	Terreno	Depreciado	19-05-2018	10	1114.12	0.00	0.00	1114.12
6	2024-0006	Computadora Dell	Móvil/ano	Alta	20-09-2006	10	7898.26	65.80	7896.26	0.00
7	2024-0007	Silla Ergonómica	Equipo de Laborat.	Depreciado	09-04-2013	5	10792.42	179.87	10792.42	0.00
8	2024-0008	Senidor HP	Equipo de Laborat.	Depreciado	19-01-2001	10	9940.50	104.50	9940.50	0.00
9	2024-0009	Auto Hyundai	Equipo de Laborat.	Alta	03-03-2024	5	6291.08	104.85	838.81	5422.27
10	2024-0010	Terrero Rústico	Vehículo	Depreciado	05-01-2003	5	3733.30	62.22	3733.30	0.00
11	2024-0011	Laboratorio Químico	Equipo de Computo	Baja por venta	22-08-2011	4	4789.93	99.79	4789.93	0.00
12	2024-0012	Silla Ergonómica	Móvil/ano	Baja	01-12-2011	10	8339.75	89.50	8339.75	0.00
13	2024-0013	Terrero Urbano	Vehículo	Depreciado	20-01-2019	5	4853.94	102.23	4853.94	0.00
14	2024-0014	Auto Toyota	Equipo de Computo	Alta	15-06-2022	4	9713.32	202.32	5887.29	3844.09
15	2024-0015	Monitor Samsung	Vehículo	Baja	09-02-2012	5	5943.66	99.06	5943.66	0.00
16	2024-0016	Terrero Urbano	Equipo de Laborat.	Alta	23-06-2007	5	1723.07	28.53	1723.07	0.00
17	2024-0017	Senidor HP	Móvil/ano	Depreciado	19-01-2015	5	4212.62	100.00	4212.62	0.00
18	2024-0018	Auto Hyundai	Móvil/ano	Depreciado	23-11-2015	10	3616.81	30.09	3439.45	361.06
19	2024-0019	Parcela Agrícola	Móvil/ano	Depreciado	03-09-2019	10	3268.94	27.24	1688.95	1579.99
20	2024-0020	Parcela Agrícola	Móvil/ano	Baja por venta	11-07-2019	10	8247.36	68.73	4399.59	3848.77
21	2024-0021	Auto Toyota	Equipo de Computo	Alta	17-02-2021	4	1474.27	100.00	1474.27	92.04
22	2024-0022	Computadora Dell	Equipo de Laborat.	Baja por venta	29-02-2023	4	4210.10	82.14	4929.16	0.00
23	2024-0023	Monitor Samsung	Equipo de Computo	Baja por venta	29-02-2023	4	4210.10	82.14	1750.46	2450.64
24	2024-0024	Senidor HP	Equipo de Laborat.	Depreciado	21-04-2023	5	9915.37	165.26	2974.61	6940.76
25	2024-0025	Computadora Dell	Equipo de Laborat.	Alta	21-10-2010	5	9958.01	165.98	9958.01	0.00
26	2024-0026	Camioneta Ford	Vehículo	Baja	25-04-2007	4	7234.52	60.70	7234.52	0.00
27	2024-0027	Auto Toyota	Móvil/ano	Depreciado	17-11-2011	10	9011.51	75.10	9011.51	0.00
28	2024-0028	Terrero Urbano	Terreno	Baja	05-10-2016	4	9849.08	205.17	9848.09	0.00
29	2024-0029	Senidor HP	Móvil/ano	Depreciado	17-12-2005	10	3179.71	25.50	3179.71	0.00
30	2024-0030	Terrero Urbano	Equipo de Computo	Depreciado	04-03-2004	5	3324.34	87.50	2713.46	554.76
31	2024-0031	Mesa de Reuniones	Móvil/ano	Depreciado	03-07-2002	10	8957.24	82.14	9857.24	0.00
32	2024-0032	Microscopio Nikon	Equipo de Laborat.	Baja	27-07-2018	5	1483.91	80.66	4839.31	0.00
33	2024-0033	Auto Toyota	Equipo de Computo	Alta	14-01-2011	10	1733.53	103.53	10411.85	0.00
34	2024-0034	Microscopio Nikon	Equipo de Computo	Alta	21-08-2014	4	1982.59	148.11	1982.59	0.00
35	2024-0035	Senidor HP	Móvil/ano	Depreciado	25-03-2008	10	7234.52	60.70	7234.52	0.00
36	2024-0036	Auto Toyota	Terreno	Alta	25-09-2015	0	4593.50	0.00	0.00	4593.50
37	2024-0037	Microscopio Nikon	Equipo de Computo	Depreciado	19-08-2024	4	2349.42	48.95	97.89	2251.63
38	2024-0038	Computadora Dell	Equipo de Computo	Alta	10-12-2018	4	1524.88	152.98	7342.88	0.00
39										

REQMS-011: Modificar activo fijo

Valorium



Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisici.	Vida Util	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
51	2024-0051	Camioneta Ford	Vehículo	Depreciado	17-07-2024	5	5997.54	99.96	4198.28	1799.28
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	5	1593.41	26.56	0.00	1593.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	46.31	5557.23	0.00
54	2024-0054	Servidor HP	Mobiliario	Depreciado	01-11-2012	10	3696.82	30.06	3606.82	0.00
55	2024-0055	Terreno Urbano	Equipo de Laborat.	Alta	17-03-2001	5	5333.63	88.89	5333.63	0.00
56	2024-0056	Microscopio Olymp.	Terrero	Baja por venta	02-10-2005	0	3583.95	0.00	0.00	3583.95
57	2024-0057	Monitor Samsung	Equipo de Laborat.	Depreciado	28-04-2012	5	3050.05	50.83	3050.05	0.00
58	2024-0058	Auto Hyundai	Vehículo	Depreciado	08-04-2013	10	5121.26	49.28	5121.26	0.00
59	2024-0059	Laboratorio Químico	Terrero	Baja	10-03-2007	0	3951.11	0.00	0.00	3959.11
60	2024-0060	Terreno Urbano	Equipo de Computo	Alta	21-03-2024	4	1344.03	28.00	1344.03	0.00
61	2024-0061	Terreno Urbano	Terrero	Baja	09-11-2020	0	2251.36	0.00	0.00	2251.36
62	2024-0062	Escritorio de Madera	Vehículo	Baja	25-03-2020	5	2846.33	10.44	2846.33	0.00
63	2024-0063	Adaptador de Computo	Vehículo	Depreciado	12-07-2010	4	5644.41	117.59	5644.41	0.00
64	2024-0064	Auto Toyota	Terrero	Alta	03-04-2018	0	5435.18	0.00	0.00	5435.18
65	2024-0065	Terreno Rustico	Mobiliario	Alta	03-12-2006	10	7277.17	60.64	7277.17	0.00
66	2024-0066	Camioneta Ford	Vehículo	Alta	03-04-2008	5	6114.44	101.91	6114.44	0.00
67	2024-0067	Computadora Dell	Mobiliario	Depreciado	02-01-2009	10	8691.99	99.96	8691.99	0.00
68	2024-0068	Silla Ergonomica	Mobiliario	Alta	14-06-2007	10	1901.76	15.85	1933.63	0.00
69	2024-0069	Laboratorio Químico	Equipo de Computo	Baja	11-03-2002	4	1839.51	38.32	1839.51	0.00
70	2024-0070	Sevidor HP	Equipo de Computo	Depreciado	25-12-2024	4	5629.69	117.29	0.00	5629.69
71	2024-0071	Microscopio Olymp.	Terrero	Baja por venta	24-04-2016	0	1001.61	0.00	0.00	1001.61
72	2024-0072	Auto Toyota	Vehículo	Depreciado	21-07-2010	5	3486.14	10.44	3486.14	0.00
73	2024-0073	Silla Ergonomica	Terrero	Baja por venta	24-07-2016	0	3180.51	0.00	0.00	3180.51
74	2024-0074	Microscopio Olymp.	Terrero	Alta	05-04-2010	0	5717.51	0.00	0.00	5717.51
75	2024-0075	Microscopio Olymp.	Vehículo	Baja	19-03-2007	5	10792.37	179.87	10792.37	0.00
76	2024-0076	Terreno Urbano	Equipo de Laborat.	Depreciado	01-01-2007	5	7983.96	19.00	7983.96	0.00
77	2024-0077	Microscopio Olymp.	Mobiliario	Baja por venta	25-02-2000	10	5129.73	42.75	5129.73	0.00
78	2024-0078	Terreno Rustico	Terrero	Baja por venta	06-07-2007	0	9373.94	0.00	0.00	9373.94
79	2024-0079	Mesa de Reuniones	Equipo de Computo	Baja	04-10-2016	4	5776.93	120.35	5776.93	0.00
80	2024-0080	Sevidor HP	Equipo de Computo	Depreciado	28-09-2005	5	6557.31	10.44	6557.31	0.00
81	2024-0081	Computadora Dell	Equipo de Computo	Depreciado	15-06-2003	4	10201.19	21.52	10201.19	0.00
82	2024-0082	Microscopio Olymp.	Equipo de Laborat.	Baja por venta	28-10-2003	5	8890.11	148.17	8890.11	0.00
83	2024-0083	Auto Hyundai	Mobiliario	Baja por venta	10-01-2013	10	10321.47	86.01	10321.47	0.00
84	2024-0084	Senador HP	Mobiliario	Depreciado	09-09-2011	10	4237.65	35.31	4237.65	0.00
85	2024-0085	Senador HP	Mobiliario	Depreciado	07-01-2009	5	10237.11	17.15	10237.11	0.00
86	2024-0086	Terreno Rustico	Mobiliario	Alta	12-10-2001	10	2339.72	19.50	2339.72	0.00
87	2024-0087	Mesa de Reuniones	Terrero	Baja	15-12-2011	0	1980.39	0.00	0.00	1980.39
88	2024-0088	Monitor Samsung	Equipo de Laborat.	Depreciado	17-03-2022	5	8788.38	146.47	4887.14	4101.24
89	2024-0089	Microscopio Olymp.	Vehículo	Depreciado	01-01-2007	10	3013.28	77.07	80.83	0.00
90	2024-0090	Mesa de Reuniones	Mobiliario	Baja por venta	24-09-2005	5	9718.43	80.99	9718.43	0.00
91	2024-0091	Mesa de Reuniones	Equipo de laborat.	Alta	19-09-2005	5	3570.77	59.51	3570.77	0.00
92	2024-0092	Terreno Rustico	Mobiliario	Alta	20-03-2024	10	6353.85	82.95	370.64	5983.21
93	2024-0093	Terreno Urbano	Vehículo	Baja por venta	16-01-2009	5	8868.00	144.80	8868.00	0.00
94	2024-0094	Escritorio de Madera	Vehículo	Baja por venta	06-02-2004	0	2815.00	0.00	0.00	2815.00
95	2024-0095	Parcera Arpícola	Mobiliario	Baja	06-15-2002	10	5412.60	45.11	5412.60	0.00
96	2024-0096	Auto Hyundai	Terrero	Baja por venta	10-06-2005	0	10341.29	0.00	0.00	10341.29
97	2024-0097	Auto Toyota	Equipo de Laborat.	Baja	05-03-2009	5	3542.26	89.04	3542.26	0.00
98	2024-0098	Computadora Dell	Equipo de Computo	Depreciado	01-01-2009	10	2303.24	55.38	2303.24	0.00
99	2024-0099	Microscopio Olymp.	Vehículo	Baja por venta	11-10-2017	5	3088.01	61.47	3088.01	0.00
100	2024-0100	Monitor Samsung	Terrero	Alta	13-06-2019	0	5690.82	0.00	0.00	5690.82
101	2022-0001	TEST-NOTIFICACION	Equipo de Computo	Depreciado	01-12-2022	4	600.00	12.50	287.50	312.50
102	2020-0001	TEST-NOTIFICACION	Equipo de Computo	Depreciado	01-01-2020	4	60.00	1.25	60.00	0.00
103	2020-0002	TEST-NOTIFICACION	Equipo de Computo	Depreciado	03-07-2020	4	30.00	0.88	30.00	0.00
104	2021-0001	TEST-2	Equipo de Computo	Alta	09-01-2021	4	80.00	1.88	80.00	31.75

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$5/98.66 K | Total de Depreciación Mensual: \$/7.31 K | Total de Depreciación Acumulada: \$/432.84 K | Total de Valor Residual: \$/165.81 K

REQMS-012: Ver historial de cambios de activos fijos

Valorium



Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisici.	Vida Util	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
51	2024-0051	Camioneta Ford	Vehículo	Depreciado	17-07-2024	5	5997.54	99.96	4198.28	1799.28
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	5	1593.41	26.56	0.00	1593.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	46.31	5557.23	0.00
54	2024-0054	Senidor HP	Mobiliario	Alta	01-11-2012	10	3696.82	30.06	3606.82	0.00
55	2024-0055	Terreno Urbano	Equipo de Laborat.	Alta	17-03-2001	5	5333.63	88.89	5333.63	0.00
56	2024-0056	Microscopio Olymp.	Terrero	Baja por venta	02-10-2005	0	3583.95	0.00	0.00	3583.95
57	2024-0057	Monitor Samsung	Equipo de Laborat.	Depreciado	28-04-2012	5	3050.05	50.83	3050.05	0.00
58	2024-0058	Auto Hyundai	Mobiliario	Baja	08-04-2013	10	5121.26	42.68	5121.26	0.00
59	2024-0059	Laboratorio Químico	Terrero	Baja	10-03-2007	0	3951.11	0.00	0.00	3959.11
60	2024-0060	Terreno Urbano	Equipo de Computo	Alta	21-03-2024	4	1344.03	28.00	1344.03	0.00
61	2024-0061	Terreno Urbano	Terrero	Baja	09-11-2020	0	2251.36	0.00	0.00	2251.36
62	2024-0062	Escritorio de Madera	Vehículo	Baja	25-03-2020	5	2846.33	10.44	2846.33	0.00
63	2024-0063	Adaptador de Computo	Vehículo	Depreciado	12-07-2010	4	5644.41	117.59	5644.41	0.00
64	2024-0064	Auto Toyota	Terrero	Alta	03-04-2018	0	5435.18	0.00	0.00	5435.18
65	2024-0065	Terreno Rustico	Mobiliario	Alta	03-12-2006	10	7277.17	60.64	7277.17	0.00
66	2024-0066	Camioneta Ford	Vehículo	Alta	03-04-2008	5	6114.44	101.91	6114.44	0.00
67	2024-0067	Computadora Dell	Mobiliario	Depreciado	02-01-2009	10	8691.99	92.18	8691.99	0.00
68	2024-0068	Silla Ergonomica	Mobiliario	Alta	07-06-2007	10	1901.76	15.85	1933.63	0.00
69	2024-0069	Microscopio Olymp.	Vehículo	Baja por venta	02-10-2005	0	3583.95	0.00	0.00	3583.95
70	2024-0070	Monitor Samsung	Equipo de Laborat.	Depreciado	28-04-2012	5	3050.05	50.83	3050.05	0.00
71	2024-0071	Senidor HP	Equipo de Computo	Depreciado	01-01-2020	4	600.00	12.50	287.50	312.50
72	2024-0072	Auto Toyota	Vehículo	Depreciado	21-07-2010	5	3486.14	10.44	3486.14	0.00
73	2024-0073	Silla Ergonomica	Terrero	Baja por venta	24-07-2016	0	3180.51	0.00	0.00	3180.51
74	2024-0074	Microscopio Olymp.	Terrero	Alta	05-04-2010	0	5717.51	0.00	0.00	5717.51
75	2024-0075	Microscopio Olymp.	Vehículo	Baja	19-03-2007	5	10792.37	179.87	10792.37	0.00
76	2024-0076	Terreno Urbano	Equipo de Laborat.	Depreciado	01-01-2007	10	7983.96	133.07	7983.96	0.00
77	2024-0077	Terreno Rustico	Mobiliario	Alta	15-12-2011	0	5129.73	42.75	5129.73	0.00
78	2024-0078	Terreno Urbano	Terrero	Baja por venta	24-09-2023	0	3573.94	0.00	0.00	3573.94
79	2024-0079	Monitor Samsung	Equipo de Laborat.	Depreciado	17-03-2022	5	5759.93	120.35	5776.93	0.00
80	2024-0080	Terreno Urbano	Mobiliario	Baja por venta	24-09-2005	10	6552.73	109.21	6552.73	0.00
81	2024-0081	Mesa de Reuniones	Mobiliario	Alta	19-01-2004	5	3571.77	10.44	3571.77	0.00
82	2024-0082	Terreno Rustico	Mobiliario	Alta	20-03-2024	10	6353.85	52.95	370.64	5983.21
83	2024-0083	Senidor HP	Vehículo	Baja por venta	06-01-2004	5	8688.20	144.80	8688.20	0.00
84	2024-0084	Escritorio de Madera	Terrero	Baja por venta	06-02-2024	0	2815.00	0.00	0.00	2815.00
85	2024-0085	Parcera Arpícola	Mobiliario	Baja	06-06-2002	10	5412.50	54.11	5412.50	0.00
86	2024-0086	Mesa de Reuniones	Terrero	Baja por venta	15-12-2001	0	1880.38	0.00	0.00	1880.38
87	2024-0087	Monitor Samsung	Equipo de Laborat.	Baja	15-12-2011	0	8691.99	72.18	8691.99	0.00
88	2024-0088	Computadora Dell	Equipo de Computo	Baja por venta	01-12-2006	4	2301.24	47.98	2301.24	0.00
89	2024-0089	Microscopio Olymp.	Vehículo	Baja por venta						

REQMS-013: Ordenar activos fijos

Valorium

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquis.	Vida Útil	Costo Inicial	Depreciación Mens.	Depreciación Acum.	Valor Residual
77	2024-0077	Microscopio Olymp. Mobiliario	Baja por venta	10	25-02-2000	10	5129.73	42.75	5129.73	0.00
78	2024-0042	Escritorio de Madera	Vehículo	Baja	13-10-2000	10	10181.42	69.37	10191.42	0.00
48	2024-0048	Servidor HP	Mobiliario	Baja	13-10-2000	10	10181.42	69.37	10191.42	0.00
55	2024-0055	Terrero Urbano	Equipo de Laborat.	Alta	17-03-2001	6	5333.63	68.89	5333.63	0.00
86	2024-0086	Terrero Urbano	Mobiliario	Baja	12-10-2001	10	2339.72	19.50	2339.72	0.00
3	2024-0003	Servidor HP	Mobiliario	Alta	13-11-2001	10	5751.08	47.93	5751.08	0.00
9	2024-0009	Servidor HP	Equipo de Laborat.	Alta	11-03-2002	5	9851.93	94.70	9851.93	0.00
67	2024-0067	Computadora Dell	Mobiliario	Depreciado	22-01-2002	10	8861.99	72.18	8861.99	0.00
69	2024-0069	Laboratorio Químico	Equipo de Computo	Baja	11-03-2002	4	1839.51	3.32	1839.51	0.00
22	2024-0022	Computadora Dell	Equipo de Laborat.	Depreciado	29-04-2002	5	4928.16	82.14	4928.16	0.00
31	2024-0111	Mesa de Reuniones	Mobiliario	Depreciado	04-05-2002	10	9851.93	94.70	9851.93	0.00
26	2024-0095	Parcela Agricola	Mobiliario	Baja	16-11-2002	10	5412.60	45.11	5412.60	0.00
10	2024-0010	Terrero Rústico	Vehículo	Depreciado	05-01-2003	6	3733.30	62.22	3733.30	0.00
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	68.31	5557.23	0.00
81	2024-0081	Computadora Dell	Equipo de Computo	Baja	16-08-2003	4	10201.19	212.52	10201.19	0.00
66	2024-0046	Laboratorio Químico	Equipo de Laborat.	Baja por venta	17-03-2003	6	5245.95	89.37	5245.95	0.00
82	2024-0082	Microscopio Olymp.	Equipo de Laborat.	Baja por venta	28-10-2003	6	8890.11	148.17	8890.11	0.00
93	2024-0095	Servidor HP	Vehículo	Baja por venta	15-01-2004	5	8868.20	144.80	8868.20	0.00
72	2024-0112	Auto Toyota	Vehículo	Baja	21-01-2005	5	1488.61	68.14	1488.61	0.00
30	2024-0080	Servidor HP	Equipo de Laborat.	Baja por venta	03-05-2005	5	6557.23	120.21	6557.23	0.00
96	2024-0096	Auto Hyundai	Terreno	Baja por venta	10-05-2005	0	10341.29	0.00	10341.29	0.00
91	2024-0091	Mesa de Reuniones	Equipo de Laborat.	Alta	19-05-2005	5	3570.77	59.51	3570.77	0.00
90	2024-0090	Mesa de Reuniones	Mobiliario	Baja por venta	24-09-2005	10	9718.43	80.93	9718.43	0.00
28	2024-0018	Microscopio Olymp.	Terreno	Baja por venta	20-07-2005	5	9851.93	160.00	9851.93	0.00
29	2024-0029	Servidor HP	Mobiliario	Depreciado	17-12-2005	5	3179.71	26.50	3179.71	0.00
47	2024-0047	Servidor HP	Mobiliario	Alta	25-12-2005	10	2620.66	218.84	2620.66	0.00
71	2024-0071	Microscopio Olymp.	Terreno	Baja por venta	24-06-2006	0	10013.61	0.00	10013.61	0.00
6	2024-0066	Computadora Dell	Mobiliario	Alta	29-09-2006	10	7898.20	155.80	7898.20	0.00
65	2024-0065	Terrero Rústico	Mobiliario	Baja por venta	17-01-2007	4	2324.14	46.98	2324.14	0.00
65	2024-0066	Computadora Dell	Equipo de Computo	Baja por venta	01-02-2007	5	7277.17	60.54	7277.17	0.00
89	2024-0089	Servidor HP	Mobiliario	Depreciado	13-01-2007	10	8012.83	66.77	8012.83	0.00
59	2024-0059	Laboratorio Químico	Mobiliario	Depreciado	10-03-2007	0	3595.11	0.00	3595.11	0.00
5	2024-0005	Microscopio Olympus	Terreno	Baja por venta	17-07-2007	5	10739.37	177.97	10739.37	0.00
76	2024-0076	Terrero Urbano	Equipo de Laborat.	Depreciado	12-05-2007	5	7983.96	133.07	7983.96	0.00
25	2024-0025	Camioneta Ford	Equipo de Computo	Alta	25-05-2007	4	3541.19	73.77	3541.19	0.00
16	2024-0016	Terrero Urbano	Equipo de Laborat.	Baja	23-06-2007	5	1712.07	26.53	1712.07	0.00
78	2024-0078	Terrero Rústico	Terreno	Baja por venta	06-07-2007	0	9373.94	0.00	9373.94	0.00
35	2024-0066	Camioneta Ford	Vehículo	Baja	25-07-2008	5	5114.44	111.91	5114.44	0.00
65	2024-0053	Microscopio Nikon	Mobiliario	Baja	25-08-2008	10	7284.52	60.70	7284.52	0.00
13	2024-0013	Terrero Urbano	Vehículo	Depreciado	29-12-2008	5	4893.94	78.23	4893.94	0.00
2	2024-0007	Auto Toyota	Equipo de Laborat.	Baja por venta	15-03-2009	5	3542.20	59.94	3542.20	0.00
43	2024-0043	Monitor Samsung	Vehículo	Baja	27-08-2009	5	8944.07	160.47	8944.07	0.00
85	2024-0085	Servidor HP	Equipo de Laborat.	Baja	27-08-2009	5	1028.71	17.15	1028.71	0.00
44	2024-0044	Microscopio Nikon	Terreno	Depreciado	09-09-2009	5	10887.81	181.46	10887.81	0.00
74	2024-0074	Microscopio Olympus	Terreno	Alta	25-04-2010	10	5717.51	0.00	5717.51	0.00
63	2024-0063	Auto Hyundai	Equipo de Computo	Baja	15-05-2010	5	5644.11	117.59	5644.11	0.00
25	2024-0025	Servidor HP	Equipo de Laborat.	Baja	27-10-2010	5	9556.91	165.98	9556.91	0.00
33	2024-0033	Terrero Rústico	Equipo de Laborat.	Baja por venta	14-01-2011	5	10411.85	173.53	10411.85	0.00
11	2024-0011	Laboratorio Químico	Equipo de Computo	Baja por venta	22-08-2011	4	4789.93	99.79	4789.93	0.00
84	2024-0084	Computadora Dell	Mobiliario	Depreciado	05-09-2011	10	4237.65	35.51	4237.65	0.00
37	2024-0037	Auto Toyota	Mobiliario	Depreciado	17-11-2011	10	3011.11	90.51	3011.11	0.00
12	2024-0012	Silla Ergonómica	Mobiliario	Baja	21-12-2011	10	8339.75	69.50	8339.75	0.00
50	2024-0050	Servidor HP	Terreno	Baja	14-12-2011	0	9405.77	0.00	9405.77	0.00
87	2024-0087	Mesa de Reuniones	Terreno	Baja	15-12-2011	0	1980.39	0.00	1980.39	0.00
4	2024-0004	Monitor Samsung	Vehículo	Baja	15-12-2011	0	10358.84	215.80	10358.84	0.00
49	2024-0049	Auto Hyundai	Equipo de Computo	Depreciado	15-12-2011	0	4542.00	87.21	4542.00	0.00

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: S/598.66 K | Total de Depreciación Mensual: S/7.31 K | Total de Depreciación Acumulada: S/432.84 K | Total de Valor Residual: S/165.81 K

REQMS-014: Filtrar activos fijos

Valorium

Lista de Activos Registrados

ID	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquis.	Vida Útil	Costo Inicial	Depreciación Mens.	Depreciación Acum.	Valor Residual
101	2022-0001	TEST	Equipo de Computo	Alta	02-12-2022	4	600.00	12.50	287.50	312.50
21	2024-0021	Auto Toyota	Equipo de Computo	Baja por venta	13-02-2021	4	1472.57	30.68	1380.53	92.04
68	2024-0068	Monitor Samsung	Mobiliario	Baja	14-05-2023	10	1991.76	15.85	253.57	1648.19
30	2024-0030	Terrero Rústico	Equipo de Laborat.	Depreciado	01-08-2020	5	3128.74	56.48	2773.95	554.79
23	2024-0023	Monitor Samsung	Equipo de Computo	Baja por venta	26-02-2023	4	4201.10	87.52	1750.46	2450.64
45	2024-0045	Parcela Agricola	Equipo de Laborat.	Depreciado	24-02-2020	5	4847.69	80.79	4605.31	242.38
51	2024-0051	Computadora Dell	Vehículo	Baja	17-03-2007	5	2774.54	44.06	4199.98	1795.25
88	2024-0088	Monitor Samsung	Equipo de Laborat.	Baja	17-03-2022	7	1788.38	145.47	4687.14	4101.24
14	2024-0014	Auto Toyota	Equipo de Computo	Baja	15-08-2022	4	9711.38	202.32	5867.29	3840.09
24	2024-0024	Servidor HP	Equipo de Laborat.	Depreciado	21-04-2023	5	9915.37	165.26	2974.61	6940.76
39	2024-0039	Terrero Rústico	Equipo de Computo	Depreciado	24-05-2023	4	10358.84	215.80	3452.88	6905.75
49	2024-0049	Auto Hyundai	Equipo de Computo	Depreciado	15-12-2011	0	4542.00	87.21	4542.00	5649.04

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: S/733.80 K | Total de Depreciación Mensual: S/1.33 K | Total de Depreciación Acumulada: S/37.01 K | Total de Valor Residual: S/36.79 K

REQMS-015: Buscar activo fijo

Valorium

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisic.	Vida Útil	Costo Inicial	Depreciación Mens.	Depreciación Acum.	Valor Residual
101	2022-0001	TEST	Equipo de Computo	Alta	02-12-2022	4	600.00	12.50	287.50	312.50
102	2022-0002	TEST-NOTIFICACI	Equipo de Computo	Depreciado	01-07-2020	4	60.00	1.88	50.00	0.00
103	2020-0002	TEST-NOTI	Equipo de Computo	Depreciado	02-07-2020	4	90.00	1.88	90.00	0.00
104	2021-0001	TEST 2	Equipo de Computo	Alta	06-01-2021	4	90.00	1.88	88.25	37.50

Nombre:

Categoría: Equipo de Computo

Fecha de Adquisición:

Costo Inicial:

Sede: Torre Arequipa

Mensajes

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$598.66 K | Total de Depreciación Mensual: \$7.31 K | Total de Depreciación Acumulada: \$432.84 K | Total de Valor Residual: \$165.81 K

REQMS-016: Vender activo fijo

Valorium

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisic.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acum.	Valor Residual
51	2022-0001	Camioneta Ford	Vehículo	Baja	17-02-2021	5	5997.54	99.55	1198.26	5800.00
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	4	1593.41	26.56	0.00	1563.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	46.31	5557.23	0.00
54	2024-0054	Servidor HP	Mobiliario	Alta	01-11-2012	10	3608.82	30.06	3608.82	0.00

Nombre:

Categoría: Equipo de Computo

Fecha de Adquisición:

Costo Inicial:

Sede: Torre Arequipa

Formulario de Venta

Detalles de Venta

Número de Factura: 04122024-0001

Activos: TEST 2

Categoría: Equipo de Computo

Fecha de Adquisición: 06-01-2021

Valor Residual: \$1.375

Precio de Venta: 3

Comprador: TEST2

RUC Comprador: 12345678912

Forma de Pago: Tarjeta de Crédito

Detalles de la Venta: TEST

Vista Previa de Factura

FACTURA
Universidad Tecnológica del Perú
Av. Arequipa, 265, Lima, Peru
Número de Factura: 04122024-0001
Fecha de Emisión: 04/12/2024
Comprador: TEST2
RUC Comprador: 12345678912
Activos: TEST 2
Categoría: Equipo de Computo
Fecha de Adquisición: 06-01-2021
Precio de Venta: S/ 3
Comprador:
RUC Comprador:
Forma de Pago: Tarjeta de Crédito
Detalles:
TEST
Gracias por su compra.

Vender **Salir**

Mensajes

19-40-08| Debe seleccionar un activo para vender.

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$598.66 K | Total de Depreciación Mensual: \$7.31 K | Total de Depreciación Acumulada: \$432.84 K | Total de Valor Residual: \$165.81 K

REQMS-017: Dar de baja activo fijo

Valorium

Lista de Activos Registrados

ID	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisici.	Vida Útil	Costo Inicial	Depreciación Mensual	Depreciación Acum.	Valor Residual
51	2024-0051	Camioneta Ford	Vehículo	Depreciado	17-12-2024	5	1593.41	25.6	0.00	1593.41
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	5	1593.41	25.6	0.00	1593.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	165.31	5557.23	0.00
54	2024-0054	Silla Ergonómica	Mobiliario	Depreciado	01-01-2010	10	3650.00	365.00	5333.62	0.00
55	2024-0055	Terrero Urbano	Equipo de Laboratorio	Alta	17-03-2001	5	5333.63	68.89	5333.63	0.00
56	2024-0056	Microscopio Olympia	Terreno	Baja por venta	02-10-2005	0	3583.95	0.00	3583.95	0.00
57	2024-0057	Monitor Samsung	Equipo de Laboratorio	Depreciado	28-04-2012	5	3054.45	50.83	3050.05	0.00
58	2024-0058	Auto Hyundai	Mobiliario	Depreciado	08-04-2013	10	2121.25	42.8	2121.25	0.00
59	2024-0059	Laboratorio Químico	Terreno	Baja por venta	10-03-2007	0	1021.11	0.00	1021.11	3595.11
60	2024-0060	Terrero Urbano	Equipo de Computo	Alta	21-03-2017	4	1344.03	28.00	1344.03	0.00
61	2024-0061	Terrero Urbano	Terreno	Baja	09-11-2020	0	2251.36	0.00	2251.36	0.00
62	2024-0062	Escrivorio de Madera	Vehículo	Depreciado	25-03-2000	5	8246.33	151.44	8246.33	0.00
63	2024-0063	Auto Toyota	Equipo de Computo	Alta	12-03-2010	4	5654.41	117.59	5654.41	0.00
64	2024-0064	Auto Toyota	Terreno	Alta	03-04-2018	0	5435.18	0.00	5435.18	0.00
65	2024-0065	Terrero Rústico	Mobiliario	Alta	03-12-2006	10	7277.17	7277.17	0.00	0.00
66	2024-0066	Camioneta Ford	Vehículo	Alta	03-04-2008	5	6114.44	101.91	6114.44	0.00
67	2024-0067	Computadora Dell	Mobiliario	Depreciado	02-04-2009	10	8693.79	8693.79	8693.79	0.00
68	2024-0068	Silla Ergonómica	Mobiliario	Alta	14-06-2003	10	1991.76	15.85	1991.76	1648.19
69	2024-0069	Laboratorio Químico	Escrivorio de Computo	Baja	11-03-2002	4	1839.51	38.32	1839.51	0.00
70	2024-0070	Servidor HP	Input	Depreciado	2024	4	5629.69	117.29	5629.69	0.00
71	2024-0071	Microscopio Olympia	Terreno	Baja por venta	2006	0	10013.61	0.00	10013.61	0.00
72	2024-0072	Auto Toyota	Equipo de Computo	Alta	12-03-2010	4	5435.18	117.29	5435.18	0.00
73	2024-0073	Silla Ergonómica	Terreno	Baja por venta	2016	0	3180.51	0.00	3180.51	0.00
74	2024-0074	Microscopio Olympia	Terreno	Baja por venta	2010	0	5717.51	0.00	5717.51	0.00
75	2024-0075	Microscopio Olympia	Terreno	Baja por venta	2007	0	7393.07	179.87	10792.37	0.00
76	2024-0076	Terrero Urbano	Equipo de Laboratorio	Alta	07-03-2007	5	5129.73	42.75	5129.73	0.00
77	2024-0077	Microscopio Olympia	Terreno	Baja por venta	2000	0	9373.94	0.00	9373.94	0.00
78	2024-0078	Terrero Rústico	Terreno	Baja por venta	06-07-2007	0	9373.94	0.00	9373.94	0.00
79	2024-0079	Mesa de Reuniones	Equipo de Computo	Terreno	04-10-2016	4	5779.93	120.35	5779.93	0.00
80	2024-0080	Servidor HP	Equipo de Computo	Terreno	08-03-2009	5	6552.21	135.21	6552.21	0.00
81	2024-0081	Computadora Dell	Equipo de Computo	Terreno	15-06-2003	4	10201.19	212.52	10201.19	0.00
82	2024-0082	Microscopio Olympia	Equipo de Laboratorio	Baja por venta	28-10-2003	5	8890.11	148.17	8890.11	0.00
83	2024-0083	Auto Hyundai	Mobiliario	Depreciado	11-05-2013	10	10321.47	86.01	10321.47	0.00
84	2024-0084	Computadora Dell	Mobiliario	Depreciado	09-09-2011	10	4237.85	35.31	4237.85	0.00
85	2024-0085	Servidor HP	Equipo de Laboratorio	Alta	07-01-2009	5	10201.19	17.15	10201.19	0.00
86	2024-0086	Terrero Rústico	Mobiliario	Alta	12-10-2001	10	2339.72	19.50	2339.72	0.00
87	2024-0087	Mesa de Reuniones	Terreno	Baja	15-12-2011	0	1980.39	0.00	1980.39	0.00
88	2024-0088	Monitor Samsung	Equipo de Laboratorio	Alta	17-03-2022	5	8788.33	146.47	4887.14	4101.24
89	2024-0089	Servidor HP	Equipo de Laboratorio	Depreciado	13-03-2007	10	3013.29	30.77	891.83	0.00
90	2024-0090	Mesa de Reuniones	Mobiliario	Depreciado	24-09-2005	10	9718.43	80.99	9718.43	0.00
91	2024-0091	Mesa de Reuniones	Equipo de Laboratorio	Alta	19-09-2005	5	3570.77	59.51	3570.77	0.00
92	2024-0092	Terrero Rústico	Mobiliario	Alta	20-03-2024	10	6353.85	62.95	370.64	5983.21
93	2024-0093	Servidor HP	Equipo de Laboratorio	Alta	16-01-2009	5	8868.00	140.80	8868.00	0.00
94	2024-0094	Escrivorio de Madera	Terreno	Baja por venta	01-01-2024	0	2815.00	0.00	2815.00	0.00
95	2024-0095	Parcela Agrícola	Mobiliario	Alta	06-15-2002	10	5412.60	45.11	5412.60	0.00
96	2024-0096	Auto Hyundai	Terreno	Baja por venta	10-06-2005	0	10341.29	0.00	10341.29	0.00
97	2024-0097	Auto Toyota	Equipo de Laboratorio	Alta	05-03-2009	5	3542.20	89.04	3542.20	0.00
98	2024-0098	Computadora Dell	Equipo de Computo	Alta	12-03-2009	4	2330.34	55.58	2330.34	0.00
99	2024-0099	Microscopio Olympia	Vehículo	Baja por venta	11-10-2017	5	3088.01	61.47	3088.01	0.00
100	2024-0100	Monitor Samsung	Terreno	Alta	13-06-2019	0	5693.82	0.00	5693.82	0.00
101	2022-0001	TEST	Equipo de Computo	Alta	05-12-2022	-4	600.00	12.50	287.50	312.50
102	2022-0002	TEST-NOTIFICADO	Equipo de Computo	Depreciado	01-01-2024	4	600.00	12.50	600.00	0.00
103	2022-0002	TEST-ADVERTIDO	Equipo de Computo	Depreciado	03-07-2020	4	200.00	5.00	200.00	0.00
104	2021-0001	TEST 2	Equipo de Computo	Baja por venta	06-01-2021	4	90.00	1.88	86.25	3.75

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: S/598.66 K | Total de Depreciación Mensual: S/7.31 K | Total de Depreciación Acumulada: S/432.84 K | Total de Valor Residual: S/165.81 K

REQMS-018: Exportar reporte de depreciación

reporte_valoriumpat - Excel (Error de activación de productos)

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisición	Vida útil (años)	Costo Inicial	Depreciación Mensual	Depreciación Acumulada	Valor Residual
1	2024-0001	Laboratorio Químico	Vehículo	Baja por venta	16-10-2012	5	4414.17	73.57	4414.17	0.00
2	2024-0002	Mesa de Reuniones	Mobiliario	Baja	04-02-2015	10	3215.00	26.79	3161.43	53.58
3	2024-0003	Servidor HP	Mobiliario	Alta	13-11-2001	10	575.00	47.93	5751.00	0.00
4	2024-0004	Microscopio Olympia	Terreno	Baja	22-05-2019	0	9818.85	0.00	9818.85	0.00
5	2024-0005	Parcela Agrícola	Terreno	Depreciado	19-05-2020	0	1114.12	0.00	1114.12	0.00
6	2024-0006	Computadora Dell	Mobiliario	Alta	20-09-2006	10	7896.26	65.80	7896.26	0.00
7	2024-0007	Silla Ergonómica	Equipo de Laboratorio	Depreciado	09-04-2013	5	10792.42	179.87	10792.42	0.00
8	2024-0008	Servidor HP	Equipo de Laboratorio	Alta	11-12-2001	5	9942.03	165.70	9942.03	0.00
9	2024-0009	Auto Hyundai	Equipo de Laboratorio	Alta	03-03-2024	5	6251.08	104.85	838.81	5452.27
10	2024-0010	Terrero Rústico	Vehículo	Depreciado	05-01-2003	5	3733.30	62.22	3733.30	0.00
11	2024-0011	Laboratorio Químico	Equipo de Computo	Baja por venta	22-06-2011	4	4789.93	99.79	4789.93	0.00
12	2024-0012	Silla Ergonómica	Mobiliario	Baja	01-12-2011	10	8339.75	69.50	8339.75	0.00
13	2024-0013	Terrero Urbano	Vehículo	Depreciado	09-12-2008	5	4691.94	78.23	4693.94	0.00
14	2024-0014	Auto Toyota	Equipo de Computo	Baja por venta	15-06-2022	4	9711.38	202.32	5867.29	3844.09
15	2024-0015	Monitor Samsung	Vehículo	Baja	06-02-2012	5	5943.66	99.06	5943.66	0.00
16	2024-0016	Terrero Urbano	Equipo de Laboratorio	Baja	23-06-2007	5	1712.07	28.53	1712.07	0.00
17	2024-0017	Servidor HP	Vehículo	Depreciado	14-05-2015	5	4217.62	70.29	4217.62	0.00
18	2024-0018	Auto Hyundai	Mobiliario	Depreciado	23-11-2015	10	3610.61	30.09	3249.55	361.06
19	2024-0019	Parcela Agrícola	Mobiliario	Depreciado	09-09-2019	10	3268.94	27.24	1688.95	1579.99
20	2024-0020	Parcela Agrícola	Mobiliario	Baja por venta	11-07-2019	10	8247.36	68.73	4398.59	3848.77
21	2024-0021	Auto Toyota	Equipo de Computo	Baja por venta	13-02-2021	4	1472.57	30.68	1380.53	92.04
22	2024-0022	Computadora Dell	Equipo de Laboratorio	Depreciado	09-04-2002	5	4928.16	82.14	4928.16	0.00
23	2024-0023	Monitor Samsung	Equipo de Computo	Baja por venta	26-02-2023	4	4201.10	87.52	1750.46	2450.64
24	2024-0024	Servidor HP	Equipo de Laboratorio	Depreciado	26-04-2023	5	9913.37	165.26	2974.61	6540.76
25	2024-0025	Servidor HP	Equipo de Laboratorio	Baja	27-05-2010	5	9950.41	165.98	9940.41	0.00
26	2024-0026	Computadora Dell	Equipo de Computo	Alta	24-05-2007	4	3541.15	73.77	3541.15	0.00
27	2024-0027	Auto Toyota	Mobiliario	Depreciado	17-11-2011	10	9011.53	75.10	9011.53	0.00
28	2024-0028	Terrero Urbano	Equipo de Computo	Alta	05-10-2016	4	9884.09	265.57	9848.69	0.00
29	2024-0029	Servidor HP	Mobiliario	Depreciado	17-12-2005	10	3179.71	26.50	3179.71	0.00
30	2024-0030	Terrero Rústico	Equipo de Laboratorio	Depreciado	01-09-2020	5	3326.74	55.48	2773.95	554.79
31	2024-0031	Mesa de Reuniones	Mobiliario	Depreciado	09-07-2002	10	9857.24	82.14	9857.24	0.00
32	2024-0032	Microscopio Nikon	Equipo de Laboratorio	Baja	27-07-2018	5	4839.31	80.66	4839.31	0.00
33	2024-0033	Terreno Rústico	Equipo de Laboratorio	Baja por venta	14-01-2011	5	10411.85	173.53	10411.85	0.00
34	2024-0034	Microscopio Nikon	Equipo de Computo	Alta	21-09-2014	4	1987.58	41.41	1987.58	0.00
35	2024-0035	Microscopio Nikon	Mobiliario	Baja	26-05-2008	10	7284.52	60.70	7284.52	0.00
36	2024-0036	Auto Toyota	Terreno	Alta	20-06-2015	0	4591.50	0.00	0.00	4593.50

Activos

REQMS-019: Proyectar depreciación futura

Valorium

Lista de Activos Registrados

ID	Código Patrimonial	NOMBRE	TENENCIA	Categoría	Estado	Fecha de Adquisici.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acum.	Valor Residual
50	2024-0051	Camioneta Ford	Vehículo	Depreciado	17-12-2024	5	5997.54	99.96	4198.28	1799.26	0.00
51	2024-0052	Computadora Dell	Vehículo	Depreciado	03-04-2003	10	1593.41	26.56	0.00	1593.41	0.00
52	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	01-01-2010	10	5557.23	45.31	5557.23	0.00	0.00
53	2024-0054	Silla Ergonomica	Vehículo	Depreciado	01-01-2010	10	3605.60	36.00	3605.60	0.00	0.00
54	2024-0055	Terreno Urbano	Equipo de Laboratorio	Alta	17-03-2001	5	5333.63	68.89	5333.63	0.00	0.00
55	2024-0056	Microscopio Olymp. Terreno	Baja por venta	02-10-2005	0	3583.95	0.00	0.00	3583.95	0.00	0.00
56	2024-0057							50.83	3050.05		
57								42.28	5121.26		
58								0.00	0.00	3589.11	
59								28.00	1344.03	0.00	
60								0.00	0.00	2251.36	
61								117.44	8246.33	0.00	
62								117.59	5544.41	0.00	
63								0.00	0.00	5435.18	
64								60.64	7277.17	0.00	
65								101.91	6114.44	0.00	
66								21.68	891.09	0.00	
67								15.85	253.67	1648.19	
68								38.32	1839.51	0.00	
69								117.29	0.00	5629.69	
70								0.00	0.00	10013.61	
71								14.14	3488.61	0.00	
72								0.00	0.00	3180.51	
73								0.00	0.00	5717.51	
74								179.87	10792.37	0.00	
75								117.07	793.00	0.00	
76								42.75	5129.73	0.00	
77								0.00	0.00	9373.94	
78								120.35	5778.93	0.00	
79								115.21	8557.01	0.00	
80								212.52	10201.18	0.00	
81								148.17	8890.11	0.00	
82								86.01	10321.47	0.00	
83								35.51	4237.65	0.00	
84								19.15	1020.00	0.00	
85								19.50	2339.72	0.00	
86								0.00	0.00	1980.39	
87								146.47	4687.14	4101.24	
88	2024-0068	Monitor Samsung	Equipo de Laboratorio	Baja	17-03-2002	5	8786.50	0.00	0.00	0.00	
89	2024-0069	Senador B. Vehículo	Vehículo	Depreciado	15-01-2007	10	8012.83	27.77	8012.83	0.00	
90	2024-0070	Mesa de Reuniones	Mobiliario	Baja por venta	24-08-2005	10	9718.43	89.99	9718.43	0.00	
91	2024-0091	Mesa de Reuniones	Equipo de Laboratorio	Alta	19-08-2005	5	3570.77	59.51	3570.77	0.00	
92	2024-0092	Terreno Rústico	Mobiliario	Baja	20-03-2024	10	6353.85	62.95	3706.64	5983.21	
93	2024-0093	Senador H. Vehículo	Vehículo	Baja por venta	01-01-2004	5	8868.00	143.80	8868.00	0.00	
94	2024-0094	Estante de Madera	Vehículo	Baja por venta	06-01-2004	5	2851.00	0.00	2851.00	0.00	
95	2024-0095	Parcela A. Terreno	Mobiliario	Baja	06-11-2002	10	5412.60	45.11	5412.60	0.00	
96	2024-0096	Auto Hyundai	Terreno	Baja por venta	10-06-2005	0	10341.29	0.00	0.00	10341.29	
97	2024-0097	Parcelsa A. Vehículo	Equipo de Laboratorio	Baja por venta	05-03-2009	4	3542.20	89.04	3542.20	0.00	
98	2024-0098	Microscopio Olymp. Terreno	Vehículo	Baja por venta	11-10-2017	5	3088.01	81.47	3088.01	0.00	
99	2024-0099	Microscopio Olymp. Terreno	Equipo de Computo	Baja	13-06-2019	0	5690.82	0.00	0.00	5690.82	
100	2024-0100	Monitor Samsung	Terreno	Alta	02-12-2022	4	600.00	12.50	287.50	312.50	
101	2024-0001	TEST	Equipo de Computo	Baja	02-12-2022	4	0.00	0.00	0.00	0.00	
102	2024-0002	TEST-NOTIFICADO	Equipo de Computo	Depreciado	03-01-2020	4	0.00	0.00	0.00	0.00	
103	2024-0002	TEST-NOTIFICADO	Equipo de Computo	Depreciado	03-01-2020	4	0.00	0.00	0.00	0.00	
104	2024-0001	TEST 2	Equipo de Computo	Baja por venta	06-01-2021	4	90.00	1.88	86.25	3.75	

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$/598.66 K | Total de Depreciación Mensual: \$/7.31 K | Total de Depreciación Acumulada: \$/432.84 K | Total de Valor Residual: \$/165.81 K

REQMS-020: Generar reporte de inventario

Valorium

Lista de Activos Registrados

ID	Código Patrimonial	NOMBRE	TENENCIA	Categoría	Estado	Fecha de Adquisici.	Vida Útil	Costo Inicial	Depreciación Mens.	Depreciación Acum.	Valor Residual
88	2024-0068	Monitor Samsung	Equipo de Laboratorio	Baja	17-03-2002	5	8786.50	0.00	0.00	0.00	
89	2024-0069	Senador H. Vehículo	Vehículo	Depreciado	15-01-2007	10	8012.83	27.77	8012.83	0.00	
90	2024-0070	Mesa de Reuniones	Mobiliario	Baja por venta	24-08-2005	10	9718.43	89.99	9718.43	0.00	
91	2024-0091	Mesa de Reuniones	Equipo de Laboratorio	Alta	19-08-2005	5	3570.77	59.51	3570.77	0.00	
92	2024-0092	Terreno Rústico	Mobiliario	Baja	20-03-2024	10	6353.85	62.95	3706.64	5983.21	
93	2024-0093	Senador H. Vehículo	Vehículo	Baja por venta	01-01-2004	5	8868.00	143.80	8868.00	0.00	
94	2024-0094	Estante de Madera	Vehículo	Baja por venta	06-01-2004	5	2851.00	0.00	2851.00	0.00	
95	2024-0095	Parcela A. Terreno	Mobiliario	Baja	23-06-2007	1712.07	28.53	1712.07	0.00	0.00	
96	2024-0096	Terreno U. Equipo d.	Baja	Depreciado	14-05-2015	4217.62	70.29	4217.62	0.00	0.00	
97	2024-0097	Senador B. Vehículo	Vehículo	Depreciado	14-05-2015	4217.62	30.59	3049.55	58.06	0.00	
98	2024-0098	Parcelsa A. Vehículo	Baja por v.	03-09-2019	1298.94	27.05	1688.55	179.99	1688.55	0.00	
99	2024-0099	Parcelsa A. Vehículo	Baja por v.	11-07-2019	2427.36	68.73	4398.59	3848.77	4398.59	0.00	
100	2024-0100	Auto Toyota	Equipo d.	Baja por v.	22-08-2011	1472.56	30.68	1380.53	92.04	0.00	
101	2024-0101	Silla Ergo	Mobiliario	Baja	01-12-2011	6339.75	69.50	8339.75	0.00	0.00	
102	2024-0102	Auto Toyota	Equipo d.	Baja	09-12-2008	4993.94	78.23	4993.94	0.00	0.00	
103	2024-0103	Terreno U. Vehículo	Vehículo	Depreciado	09-12-2008	202.32	507.30	202.32	444.09	0.00	
104	2024-0104	Monter S. Vehículo	Vehículo	Baja	05-02-2012	1943.66	99.43	1943.66	0.00	0.00	
105	2024-0105	Terreno U. Equipo d.	Baja	Depreciado	23-06-2007	1712.07	28.53	1712.07	0.00	0.00	
106	2024-0106	Terreno R. Equipo d.	Baja	Depreciado	14-05-2015	4217.62	70.29	4217.62	0.00	0.00	
107	2024-0107	Senador B. Vehículo	Vehículo	Depreciado	14-05-2015	4217.62	30.59	3049.55	58.06	0.00	
108	2024-0108	Parcelsa A. Vehículo	Baja por v.	03-09-2019	1298.94	27.05	1688.55	179.99	1688.55	0.00	
109	2024-0109	Terreno U. Equipo d.	Baja	Depreciado	17-12-2005	3179.71	26.59	3179.71	0.00	0.00	
110	2024-0110	Terreno R. Equipo d.	Baja	Depreciado	01-09-2020	3228.74	55.48	2773.95	554.79	0.00	
111	2024-0111	Terreno R. Equipo d.	Baja	Depreciado	01-09-2020	3228.74	24.24	2984.50	244.64	0.00	
112	2024-0112	Terreno R. Equipo d.	Baja	Depreciado	01-09-2020	3228.74	17.53	3141.85	57.15	0.00	
113	2024-0113	Terreno R. Equipo d.	Baja	Depreciado	14-01-2011	10411.88	14.11	10411.88	0.00	0.00	
114	2024-0114	Parcelsa A. Vehículo	Baja por v.	27-07-2018	1298.94	80.84	4801.00	80.00	0.00	0.00	
115	2024-0115	Terreno R. Equipo d.	Baja	Depreciado	21-08-2014	1987.58	41.41	1987.58	0.00	0.00	
116	2024-0116	Terreno R. Equipo d.	Baja	Depreciado	26-05-2018	2784.52	60.70	2784.52	0.00	0.00	
117	2024-0117	Terreno R. Equipo d.	Baja	Depreciado	24-06-2023	10358.64	216.80	3452.88	6905.76	0.00	
118	2024-0118	Terreno R. Equipo d.	Baja	Depreciado	24-06-2023	10358.64	181.45	4905.28	242.38	0.00	
119	2024-0119	Terreno R. Equipo d.	Baja	Depreciado	24-02-2020	487.69	0.00	487.69	0.00	0.00	
120	2024-0120	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
121	2024-0121	Terreno R. Equipo d.	Baja	Depreciado	07-06-2009	6904.36	115.07	6904.36	0.00	0.00	
122	2024-0122	Terreno R. Equipo d.	Baja	Depreciado	18-11-2024	2293.43	52.34	0.00	2293.43	0.00	
123	2024-0123	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
124	2024-0124	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
125	2024-0125	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
126	2024-0126	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
127	2024-0127	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
128	2024-0128	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
129	2024-0129	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
130	2024-0130	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
131	2024-0131	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64	0.00	0.00	
132	2024-0132	Terreno R. Equipo d.	Baja	Depreciado	05-11-2013	8991.64	0.00	8991.64</td			

REQMS-021: Gestionar la revaluación de activos fijos

Valorium

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisic.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
3	2024-0003	Servidor HP	Mobiliario	Alta	13-11-2001	10	6000.00	50.00	0.00	5943.66
4	2024-0005	Microscopio Olymp.	Terreno	Baja	22-05-2019	0	0.00	0.00	0.00	5988.65
5	2024-0006	Computadora Dell	Mobiliario	Depreciado	19-05-2018	0	1114.12	0.00	0.00	1114.12
6	2024-0006	Computadora Dell	Mobiliario	Baja	20-09-2008	10	9000.00	108.87	0.00	8992.00
7										10792.42
8										9942.03
9										0.00
10										5452.27
11										3733.30
12										0.00
13										4789.93
14										8339.75
15										0.00
16										0.00
17										4843.00
18										5867.29
19										3844.09
20										5943.66
21										0.00
22										4217.00
23										301.06
24										4217.00
25										301.06
26										4217.00
27										301.06
28										4217.00
29										301.06
30										4217.00
31										301.06
32										4217.00
33										301.06
34										4217.00
35										301.06
36										4217.00
37										301.06
38										4217.00
39										301.06
40										4217.00
41										301.06
42										4217.00
43										301.06
44										4217.00
45										301.06
46										4217.00
47										301.06
48										4217.00
49										301.06
50										4217.00
51										301.06
52										4217.00
53										301.06
54										4217.00
55										301.06
56										4217.00
57										301.06
58										4217.00
59										301.06
60										4217.00
61										301.06
62										4217.00
63										301.06
64										4217.00
65										301.06
66										4217.00
67										301.06
68										4217.00
69										301.06
70										4217.00
71										301.06
72										4217.00
73										301.06
74										4217.00
75										301.06
76										4217.00
77										301.06
78										4217.00
79										301.06
80										4217.00
81										301.06
82										4217.00
83										301.06
84										4217.00
85										301.06
86										4217.00
87										301.06
88										4217.00
89										301.06
90										4217.00
91										301.06
92										4217.00
93										301.06
94										4217.00
95										301.06
96										4217.00
97										301.06
98										4217.00
99										301.06
100										4217.00

Ubicación de Activos Cerrar sesión

Mensajes

[19-40.08] Debe seleccionar un activo para vender.
[19-40.60] Activo vendido exitosamente por \$r 3.00
[19-41.30] Activo dado de baja como: Obsolescencia
[19-42.54] Reporte exportado correctamente
[19-43.24] Por favor seleccionar un activo para proyectar la depreciación.
[19-44.47] Activo Servidor HP (ID: 3) revaluado de 5751.08 a 6000.00.
[19-44.47] Activo Computadora Dell (ID: 6) revaluado de 8798.26 a 8000.00.
[19-44.47] Activo Camioneta Ford (ID: 26) revaluado de 3541.19 a 4550.68.

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$598.66 K | Total de Depreciación Mensual: \$7.31 K | Total de Depreciación Acumulada: \$432.84 K | Total de Valor Residual: \$165.81 K

Valorium

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquisic.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
3	2024-0003	Servidor HP	Mobiliario	Alta	13-11-2001	10	6000.00	50.00	0.00	5943.66
4	2024-0004	Microscopio NIKON	Equipo de Laboratorio	Depreciado	09-09-2009	5	1087.51	181.46	1087.81	0.00
5	2024-0004	Parcelsa Agrícola	Equipo de Laboratorio	Depreciado	24-02-2005	5	4847.69	89.79	4805.31	242.38
6	2024-0004	Laboratorio Químico	Equipo de Laboratorio	Depreciado	17-09-2003	5	6249.66	104.16	6249.66	0.00
7	2024-0047	Servidor HP	Mobiliario	Alta	26-12-2005	10	2620.66	218.14	2620.66	0.00
8	2024-0047	Terreno Urbano	Equipo de Laboratorio	Baja	01-01-2007	10	1091.42	109.14	1091.42	0.00
9	2024-0047	Camioneta Ford	Equipo de Computo	Depreciado	13-01-2023	4	10429.00	217.27	10799.95	5649.94
10	2024-0050	Servidor HP	Mobiliario	Baja	14-12-2011	0	9405.77	0.00	9405.77	0.00
11	2024-0051	Camioneta Ford	Vehículo	Baja	17-05-2021	5	5997.54	99.96	4198.28	1799.26
12	2024-0052	Computadora Dell	Vehículo	Depreciado	03-04-2003	10	1593.41	26.56	0.00	1593.41
13	2024-0052	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5567.23	11.11	5567.23	0.00
14	2024-0053	Terreno Urbano	Equipo de Laboratorio	Baja	01-11-2012	10				

Ubicación de Activos Cerrar sesión

Mensajes

[19-40.08] Debe seleccionar un activo para vender.
[19-40.60] Activo vendido exitosamente por \$r 3.00
[19-41.30] Activo dado de baja como: Obsolescencia
[19-42.54] Reporte exportado correctamente
[19-43.24] Por favor seleccionar un activo para proyectar la depreciación.
[19-44.47] Activo Servidor HP (ID: 3) revaluado de 5751.08 a 6000.00.
[19-44.47] Activo Computadora Dell (ID: 6) revaluado de 8798.26 a 8000.00.
[19-44.47] Activo Camioneta Ford (ID: 26) revaluado de 3541.19 a 4550.68.

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$598.66 K | Total de Depreciación Mensual: \$7.31 K | Total de Depreciación Acumulada: \$432.84 K | Total de Valor Residual: \$165.81 K

REQMS-023: Ubicar activos

Valorium

Distribución de Activos por Ubicación

Ubicación	Cantidad
Torre Petit Thouars (24)	24
Torre Pacifico (28)	28
Torre Arequipa (32)	32
Sede Central (21)	21

Distribución de Activos por Ubicación

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquis.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
47	2024-0047	Servidor HP	Mobiliario	Alta	26-12-2005	10	2620.66	218.4	2620.66	0.00
48	2024-0048	Servidor HP	Mobiliario	Baja	13-10-2000	10	10916.42	909.7	10916.42	0.00
49	2024-0049	Auto Hyundai	Equipo de Computo	Depreciado	13-07-2023	4	10429.00	217.27	4779.96	5649.04
50	2024-0050	Servidor HP	Terrero	Baja	14-12-2011	0	9405.77	0.00	9405.77	0.00
51	2024-0051	Camioneta Ford	Vehículo	Baja	17-05-2021	5	5997.54	99.96	4198.28	1799.26
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	5	1593.41	26.56	0.00	1593.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	465.72	0.00	5557.23
54	2024-0054	Servidor HP	Mobiliario	Alta	01-11-2012	10	3606.82	30.08	3606.82	0.00
55	2024-0055	Terreno Urbano	Equipo de Laborat.	Alta	17-03-2001	5	5333.63	88.89	5333.63	0.00
56	2024-0056	Microscopio Olymp.	Terrero	Baja por venta	02-10-2005	0	3563.95	0.00	0.00	3563.95
57	2024-0057	Microscopio Olympus	Equipo de Laborat.	Alta	02-10-2005	5	5333.63	10.00	5333.63	0.00

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$600.02 K | Total de Depreciación Mensual: \$7.34 K | Total de Depreciación Acumulada: \$434.20 K | Total de Valor Residual: \$165.81 K

REQMS-024: Analizar activos en baja

Valorium

Reporte de Activos en Baja

Pérdida Bruta: S/ 23070.92 **Monto Recuperado:** S/ 18381.91 **Pérdida Neta:** S/ 21239.01

Pérdidas por Categoría

Categoría	Monto (\$/s)
Vehículo	~10,000
Mobiliario	~6,000
Equipo de Computo	~5,000
Equipo de Laboratorio	~4,000

Lista de Activos Registrados

Nº	Código Patrimonial	Nombre	Categoría	Estado	Fecha de Adquis.	Vida Útil	Costo Inicial	Depreciación Men.	Depreciación Acu.	Valor Residual
47	2024-0047	Servidor HP	Mobiliario	Alta	26-12-2005	10	2620.66	218.4	2620.66	0.00
48	2024-0048	Servidor HP	Mobiliario	Baja	13-10-2000	10	10916.42	909.7	10916.42	0.00
49	2024-0049	Auto Hyundai	Equipo de Computo	Depreciado	13-07-2023	4	10429.00	217.27	4779.96	5649.04
50	2024-0050	Servidor HP	Terrero	Baja	14-12-2011	0	9405.77	0.00	9405.77	0.00
51	2024-0051	Camioneta Ford	Vehículo	Baja	17-05-2021	5	5997.54	99.96	4198.28	1799.26
52	2024-0052	Computadora Dell	Vehículo	Depreciado	17-12-2024	5	1593.41	26.56	0.00	1593.41
53	2024-0053	Laboratorio Químico	Mobiliario	Depreciado	03-04-2003	10	5557.23	465.72	0.00	5557.23
54	2024-0054	Servidor HP	Mobiliario	Alta	01-11-2012	10	3606.82	30.08	3606.82	0.00
55	2024-0055	Terreno Urbano	Equipo de Laborat.	Alta	17-03-2001	5	5333.63	88.89	5333.63	0.00
56	2024-0056	Microscopio Olymp.	Terrero	Baja por venta	02-10-2005	0	3563.95	0.00	0.00	3563.95
57	2024-0057	Microscopio Olympus	Equipo de Laborat.	Alta	02-10-2005	5	5333.63	10.00	5333.63	0.00

Usuario: Fernando Fecha: 04/12/2024 Versión: 4.0.0 Número de Activos: 104 | Total de Costo Inicial: \$600.02 K | Total de Depreciación Mensual: \$7.34 K | Total de Depreciación Acumulada: \$434.20 K | Total de Valor Residual: \$165.81 K

9 CONCLUSIONES

1. **Fortalecimiento en la Programación Orientada a Objetos (POO):** Durante el desarrollo de "Valorium", empleamos los principios de la programación orientada a objetos (POO) en Java, como la encapsulación, herencia y polimorfismo. La implementación de clases como ActivoFijo, Usuario, y Movimiento nos permitió organizar el sistema de manera modular, facilitando la extensión y modificación de sus funcionalidades.
2. **Uso de Interfaces y Clases Abstractas:** Para estructurar la lógica de depreciación y movimientos, empleamos interfaces y clases abstractas en Java, como Depreciable y Depreciacion, que permitieron estandarizar métodos comunes para distintos tipos de depreciación. Esto facilitó la creación de métodos específicos en clases derivadas, como DepreciacionLineaRecta, asegurando una implementación coherente y extensible.
3. **Manejo de Excepciones:** La gestión de errores y validaciones fue importante para la robustez del sistema. A través del manejo de excepciones, el sistema valida y maneja entradas incorrectas, como fechas no válidas o costos iniciales incorrectos. Esto no solo garantiza la integridad de los datos, sino que también mejora la experiencia del usuario al ofrecer retroalimentación precisa.
4. **Uso de Java Swing para la Interfaz Gráfica de Usuario (GUI):** La implementación de la interfaz gráfica mediante Java Swing nos permitió diseñar una experiencia de usuario intuitiva y funcional. Con clases como JFrame, JTable, y componentes personalizados, desarrollamos un entorno visual donde los usuarios pueden realizar el registro, modificación y baja de activos, así como ver reportes detallados. Asimismo, la personalización y control de eventos en Swing enriquecieron la experiencia interactiva del sistema.
5. **Conexión con Bases de Datos mediante JDBC:** La integración con una base de datos MySQL mediante JDBC (Java Database Connectivity) permitió que el sistema registre y consulte información de manera persistente. Utilizamos conexiones y sentencias preparadas (PreparedStatement) para realizar operaciones de inserción, modificación y consulta en la base de datos, garantizando así la seguridad y eficiencia en el manejo de datos.
6. **Implementación de Arrays y Estructuras de Datos:** La clase Movimiento usa un arreglo circular para almacenar el historial de hasta 100 movimientos, lo que nos ayudó a practicar la administración de estructuras de datos. Además, el uso de colecciones como List y LinkedList en Java optimizó la gestión de activos, simplificando el proceso de búsqueda, almacenamiento y filtrado de información.
7. **Validación de Datos y Uso de Clases de Utilidad:** La validación de datos fue esencial para mantener la integridad del sistema. Desarrollamos clases de utilidad como ValidadorFechas y ValidadorActivo, donde se centralizan los métodos de validación, asegurando consistencia y facilidad de mantenimiento en la validación de entradas en la interfaz de usuario.
8. **Exportación de Reportes con Apache POI:** Para la generación de reportes en formato Excel, implementamos la biblioteca Apache POI, la cual nos permitió exportar datos detallados sobre los activos y sus depreciaciones. Este enfoque ofrece al usuario una buena herramienta para la planificación financiera y auditorías, al tiempo que refuerza nuestras habilidades en el manejo de bibliotecas adicionales en Java.

10 APÉNDICE

Apéndice A. Cuestionario de preguntas presentado (Avance 1)

Figura A1

CUESTIONARIO DE PREGUNTAS INICIAL PARA EL SISTEMA DE DEPRECIACIÓN DE ACTIVO FIJO PARA LA UNIVERSIDAD TECNOLÓGICA DEL PERÚ [GRUPO 1]

1. ¿Qué es un activo fijo en el Perú y cuánto vale?
2. ¿Cuánto es el monto para que el bien adquirido se convierta en un activo?
3. ¿Cuáles son los principales datos que se registran al dar de alta un activo fijo?
4. ¿Cómo se registran los activos fijos en la universidad?
5. ¿Cómo se maneja en una institución el registro de los activos fijos en el ámbito contable?
6. ¿Qué tipos de activos fijos tiene la universidad?
7. ¿Cómo se gestiona la incorporación de nuevos tipos de activos fijos en el sistema?
8. ¿Cuáles son los estados posibles de un activo fijo en el sistema?
9. ¿Qué criterios se utilizan para calcular la vida útil estimada de un activo?
10. ¿Qué métodos utiliza la universidad para depreciar sus activos fijos?
11. ¿Cuántos tipos de depreciación hay en el Perú?
12. ¿Cómo se calcula la depreciación de un activo fijo en el Perú?
13. ¿Desde cuándo se deprecia un activo?
14. ¿Qué funcionalidades puede ofrecer el sistema para la simulación de diferentes escenarios de depreciación?
15. ¿Cómo se gestionan las excepciones o errores detectados durante el cálculo de la depreciación?
16. ¿Qué campos permitirían filtrar activos en los reportes?
17. ¿Qué formatos de exportación estarían disponibles para los reportes generados por el sistema?
18. ¿Qué actores interactuarían con el sistema?
19. ¿El sistema tiene que integrarse con otras áreas de la universidad?
20. ¿Los usuarios finales requerirán una capacitación para usar el sistema?
21. ¿Cuáles son las razones más comunes para la renovación de un activo fijo?
22. ¿Qué sucede con los activos que perdieron su vida útil o la universidad dio de baja?
23. ¿Qué procedimiento realiza el sistema cuando un activo fijo pierde completamente su valor?
24. ¿Cuáles son los estados finales posibles de un activo fijo al concluir su ciclo de vida?

Apéndice B. Estructura del Código Fuente para 24 requerimientos (Entrega final)

Resumen del Código

Este apéndice detalla la estructura del sistema Valorium, desarrollado para gestionar la depreciación de activos fijos en la Universidad Tecnológica del Perú. Este sistema permite realizar operaciones como registrar, modificar y dar de baja activos, así como generar reportes de depreciación, ofreciendo una operación en base de datos confiable y eficiente.

El código se organiza en cinco paquetes que representan capas funcionales claras, facilitando la modularidad y el mantenimiento del sistema. Cada paquete contiene clases ordenadas alfabéticamente para mejorar la accesibilidad. Los paquetes incluyen:

- **controlador:** Gestiona el flujo de la aplicación y conecta la interfaz de usuario con la lógica de negocio. Incluye clases como *Autenticacion*, que permite el acceso seguro al sistema, y *ActivoControlador*, que administra operaciones principales sobre los activos, como agregar, modificar y dar de baja.
- **modelo:** Define las entidades centrales del sistema, representando los activos y las reglas de depreciación. Contiene clases como *ActivoFijo*, que encapsula las propiedades y métodos relacionados con cada activo, y *DepreciacionLineaRecta*, que implementa el cálculo de depreciación mensual bajo el método de línea recta. También incluye clases de movimientos (*MovimientoBaja*, *MovimientoModificacion*, etc.) que registran las acciones realizadas sobre los activos.
- **servicio:** Implementa la lógica de negocio del sistema. Su clase principal, *ActivoServicio*, administra operaciones complejas como la carga desde la base de datos, el cálculo de valores de depreciación acumulada y la generación de reportes históricos de movimientos de activos. Además, centraliza las reglas que rigen el comportamiento del sistema.
- **util:** Contiene clases de utilidad y validación. Ejemplos incluyen *ValidadorActivo*, que asegura que los datos ingresados sean correctos y consistentes antes de ser procesados, y *ConexionBD*, que gestiona la conexión a la base de datos de manera eficiente y segura.
- **vista:** Es responsable de la interfaz gráfica, proporcionando ventanas como *LoginFrame* y *MainFrame*, que permiten la interacción del usuario con las funcionalidades del sistema de forma intuitiva y visualmente organizada. Otras ventanas, como *RenovacionFrame* y *AnalisisDeBajasFrame*, presentan reportes y operaciones específicas de los activos.

Paquete: controlador; Clase: ActivoControlador.java

```
1. package controlador;
2.
3. import modelo.ActivoFijo;
4. import modelo.Sede;
5. import servicio.ActivoServicio;
6.
7. import java.util.List;
8.
9. public class ActivoControlador {
10.
11.     private final ActivoServicio activoServicio;
12.
13.     public ActivoControlador(ActivoServicio activoServicio) {
14.         this.activoServicio = activoServicio;
15.     }
16.
17.     public List<ActivoFijo> obtenerInventario() {
18.         return activoServicio.cargarActivosDesdeBD();
19.     }
20. }
```

Paquete: controlador; Clase: Autenticacion.java

```
1. package controlador;
2.
3. import modelo.Usuario;
4. import modelo.UsuarioSesion;
5. import util.ConexionBD;
6.
7. import java.sql.Connection;
8. import java.sql.PreparedStatement;
9. import java.sql.ResultSet;
10. import java.sql.SQLException;
11.
12. public class Autenticacion {
13.
14.     public Usuario autenticarUsuario(String nombreUsuario, String contrasena) {
15.         String query = "SELECT * FROM usuarios WHERE nombre_usuario = ? AND
contrasena = ?";
16.
17.         try (Connection conn = ConexionBD.getConnection();
18.              PreparedStatement stmt = conn.prepareStatement(query)) {
19.             stmt.setString(1, nombreUsuario);
20.             stmt.setString(2, contrasena);
21.
22.             ResultSet rs = stmt.executeQuery();
23.             if (rs.next()) {
24.                 Usuario usuario = new Usuario(rs.getString("nombre_usuario"),
rs.getString("contrasena"), rs.getString("rol"));
25.                 UsuarioSesion.getInstancia().setUsuarioAutenticado(usuario);
26.                 return usuario;
27.             }
28.         } catch (SQLException e) {
29.             e.printStackTrace();
30.         }
31.         return null;
32.     }
33. }
```

Paquete: controlador; Clase: Main.java

```

1. package controlador;
2.
3. import modelo.Usuario;
4. import servicio.ActivoServicio;
5. import vista.LoginFrame;
6. import vista.MainFrame;
7.
8. import javax.swing.*;
9.
10. public class Main {
11.
12.     public static Usuario usuarioAutenticado;
13.
14.     public static void main(String[] args) {
15.         SwingUtilities.invokeLater(() -> {
16.             ActivoServicio activoServicio = new ActivoServicio();
17.             ActivoControlador activoControlador = new
18.                 ActivoControlador(activoServicio);
19.                 mostrarLogin();
20.             });
21.
22.     private static void mostrarLogin() {
23.         LoginFrame loginFrame = new LoginFrame();
24.         loginFrame.setVisible(true);
25.     }
26.
27.     public static void iniciarMainFrame() {
28.         MainFrame mainFrame = new MainFrame();
29.         mainFrame.setVisible(true);
30.     }
31. }
```

Paquete: modelo; Clase: ActivoFijo.java

```

1. package modelo;
2.
3. import java.time.LocalDate;
4. import java.time.format.DateTimeFormatter;
5. import java.time.temporal.ChronoUnit;
6.
7. public class ActivoFijo extends Bien implements Depreciable {
8.
9.     private int numero;
10.    private String codigoPatrimonial;
11.    private String fechaAdquisicion;
12.    private String fechaInicioDepreciacion;
13.    private int vidaUtil;
14.    private double depreciacionMensual;
15.    private String responsableRegistro;
16.    private String motivoBaja;
17.    private String tipoBaja;
18.    private Sede sede;
19.
20.    public ActivoFijo(String nombre, String categoria, double costoInicial, String
21.                      fechaAdquisicion, String responsableRegistro, Sede sede) {
22.        super(nombre, categoria, costoInicial);
23.        this.fechaAdquisicion = fechaAdquisicion;
24.        this.vidaUtil = determinarVidaUtil(categoría);
25.        actualizarEstado();
26.        this.depreciacionMensual = calcularDepreciacionMensual();
27.        this.responsableRegistro = responsableRegistro;
28.        this.sede = sede;
```

```

29.         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
30.         LocalDate fechaAdq = LocalDate.parse(fechaAdquisicion, formatter);
31.         this.fechaInicioDepreciacion = fechaAdq.plusMonths(1).format(formatter);
32.         this.codigoPatrimonial = "";
33.     }
34.
35.     public Sede getSede() {
36.         return sede;
37.     }
38.
39.     public int getNumero() {
40.         return this.numero;
41.     }
42.
43.     public void setNumero(int numero) {
44.         this.numero = numero;
45.     }
46.
47.     public void setId(int id) {
48.     }
49.
50.     public String getCodigoPatrimonial() {
51.         return codigoPatrimonial;
52.     }
53.
54.     public void setCodigoPatrimonial(String codigoPatrimonial) {
55.         this.codigoPatrimonial = codigoPatrimonial;
56.     }
57.
58.     public String getFechaAdquisicion() {
59.         return fechaAdquisicion;
60.     }
61.
62.     public int getVidaUtil() {
63.         return vidaUtil;
64.     }
65.
66.     public void setFechaAdquisicion(String fechaAdquisicion) {
67.         this.fechaAdquisicion = fechaAdquisicion;
68.
69.         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
70.         LocalDate fechaAdq = LocalDate.parse(fechaAdquisicion, formatter);
71.         this.fechaInicioDepreciacion = fechaAdq.plusMonths(1).format(formatter);
72.     }
73.
74.     public void setVidaUtil(String nuevaCategoria) {
75.         this.vidaUtil = determinarVidaUtil(nuevaCategoria);
76.     }
77.
78.     @Override
79.     public double calcularDepreciacionMensual() {
80.         if (categoria.equals("Terreno")) {
81.             return 0.0;
82.         }
83.         DepreciacionLineaRecta depreciacion = new
84.             DepreciacionLineaRecta(costoInicial, vidaUtil);
85.         return depreciacion.calcularDepreciacion() / 12;
86.     }
87.
88.     public double calcularDepreciacionAcumulada() {
89.         if (vidaUtil == 0) {
90.             return 0.0;
91.         }
92.         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");

```

```

93.         LocalDate fechaInicioDepreciacion =
94.             LocalDate.parse(this.fechaInicioDepreciacion, formatter);
95.
96.         long diasTranscurridos = ChronoUnit.DAYS.between(fechaInicioDepreciacion,
97.             fechaActual);
97.
98.         if (diasTranscurridos < 30) {
99.             return 0.0;
100.        }
101.
102.        long mesesTranscurridos = diasTranscurridos / 30;
103.
104.        long mesesVidaUtil = (long) vidaUtil * 12;
105.        if (mesesTranscurridos > mesesVidaUtil) {
106.            mesesTranscurridos = mesesVidaUtil;
107.        }
108.
109.        return mesesTranscurridos * (costoInicial / (double) mesesVidaUtil);
110.    }
111.
112.    @Override
113.    public double calcularValorResidual() {
114.        if (vidaUtil == 0) {
115.            return costoInicial;
116.        }
117.
118.        double depreciacionAcumulada = calcularDepreciacionAcumulada();
119.        return Math.max(0, costoInicial - depreciacionAcumulada);
120.    }
121.
122.    public static int determinarVidaUtil(String categoria) {
123.        switch (categoria) {
124.            case "Equipo de Computo":
125.                return 4;
126.            case "Mobiliario":
127.                return 10;
128.            case "Equipo de Laboratorio":
129.                return 5;
130.            case "Vehiculo":
131.                return 5;
132.            case "Terreno":
133.                return 0;
134.            default:
135.                return 1;
136.        }
137.    }
138.
139.    @Override
140.    public void actualizarEstado() {
141.        if ("Baja".equalsIgnoreCase(this.estado) || "Baja por
142. venta".equalsIgnoreCase(this.estado)) {
143.            return;
144.        }
145.        if ("Terreno".equalsIgnoreCase(categoria) || vidaUtil == 0) {
146.            this.estado = "Alta";
147.            return;
148.        }
149.
150.        LocalDate fechaAdq = LocalDate.parse(fechaAdquisicion,
151. DateTimeFormatter.ofPattern("dd-MM-yyyy"));
152.        LocalDate fechaFinVidaUtil = fechaAdq.plusYears(vidaUtil);
153.        LocalDate fechaActual = LocalDate.now();
154.        if (fechaActual.isAfter(fechaFinVidaUtil)) {

```

```

155.             this.estado = "Depreciado";
156.         } else {
157.             this.estado = "Alta";
158.         }
159.     }
160.
161.     public void darDeBaja(String motivo, String tipoBaja) {
162.         this.motivoBaja = motivo;
163.         this.tipoBaja = tipoBaja;
164.         this.estado = "Baja";
165.     }
166. }
```

Paquete: modelo; Clase: Bien.java

```

1. package modelo;
2.
3. abstract class Bien {
4.
5.     protected String nombre;
6.     protected String categoria;
7.     protected double costoInicial;
8.     protected String estado;
9.
10.    public Bien(String nombre, String categoria, double costoInicial) {
11.        this.nombre = nombre;
12.        this.categoria = categoria;
13.        this.costoInicial = costoInicial;
14.        this.estado = "Alta";
15.    }
16.
17.    public String getNombre() {
18.        return nombre;
19.    }
20.
21.    public String getCategoría() {
22.        return categoria;
23.    }
24.
25.    public double getCostoInicial() {
26.        return costoInicial;
27.    }
28.
29.    public String getEstado() {
30.        return estado;
31.    }
32.
33.    public void setNombre(String nombre) {
34.        this.nombre = nombre;
35.    }
36.
37.    public void setCategoría(String categoria) {
38.        this.categoria = categoria;
39.    }
40.
41.    public void setCostoInicial(double costoInicial) {
42.        this.costoInicial = costoInicial;
43.    }
44.
45.    public void setEstado(String estado) {
46.        this.estado = estado;
47.    }
48.
49.    public abstract double calcularValorResidual();
```

```
50. }
```

Paquete: modelo; Clase: Depreciable.java

```
1. package modelo;
2.
3. public interface Depreciable {
4.
5.     double calcularDepreciacionMensual();
6.     void actualizarEstado();
7. }
```

Paquete: modelo; Clase: Depreciacion.java

```
1. package modelo;
2.
3. public abstract class Depreciacion {
4.
5.     protected double costoInicial;
6.     protected int vidaUtil;
7.
8.     public Depreciacion(double costoInicial, int vidaUtil) {
9.         this.costoInicial = costoInicial;
10.        this.vidaUtil = vidaUtil;
11.    }
12.
13.    public abstract double calcularDepreciacion();
14.
15. }
```

Paquete: modelo; Clase: DepreciacionLineaRecta.java

```
1. package modelo;
2.
3. public class DepreciacionLineaRecta extends Depreciacion {
4.
5.     public DepreciacionLineaRecta(double costoInicial, int vidaUtil) {
6.         super(costoInicial, vidaUtil);
7.     }
8.
9.     @Override
10.    public double calcularDepreciacion() {
11.        return costoInicial / vidaUtil;
12.    }
13. }
```

Paquete: modelo; Clase: GuardarActivoResultado.java

```
1. package modelo;
2.
3. public class GuardarActivoResultado {
4.     private int id;
5.     private String codigoPatrimonial;
6.
7.     public GuardarActivoResultado(int id, String codigoPatrimonial) {
8.         this.id = id;
9.         this.codigoPatrimonial = codigoPatrimonial;
10.    }
```

```

11.     public int getId() {
12.         return id;
13.     }
14.
15.     public String getCodigoPatrimonial() {
16.         return codigoPatrimonial;
17.     }
18. }
19.

```

Paquete: modelo; Clase: Movimiento.java

```

1. package modelo;
2.
3. import util.ConexionBD;
4.
5. import java.sql.Connection;
6. import java.sql.PreparedStatement;
7. import java.sql.SQLException;
8. import java.sql.Timestamp;
9. import java.time.LocalDateTime;
10. import java.time.format.DateTimeFormatter;
11.
12. public abstract class Movimiento implements Registrable {
13.     protected String tipoMovimiento;
14.     protected LocalDateTime fechaHora;
15.
16.     public Movimiento(String tipoMovimiento) {
17.         this.tipoMovimiento = tipoMovimiento;
18.         this.fechaHora = LocalDateTime.now();
19.     }
20.
21.     public String getFechaHora() {
22.         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy
HH:mm:ss");
23.         return fechaHora.format(formatter);
24.     }
25.
26.     public void guardarMovimientoEnBD(int idActivo) {
27.         String sql = "INSERT INTO movimientos (id_activo, tipo_movimiento,
fecha_hora, motivo_baja, tipo_baja, precio_venta, detalle_general) VALUES (?, ?, ?, ?,
?, ?, ?, ?)";
28.         try (Connection conn = ConexionBD.getConnection();
29.              PreparedStatement pstmt = conn.prepareStatement(sql)) {
30.
31.             pstmt.setInt(1, idActivo);
32.             pstmt.setString(2, tipoMovimiento);
33.             pstmt.setTimestamp(3, Timestamp.valueOf(fechaHora));
34.         } catch (SQLException e) {
35.             e.printStackTrace();
36.         }
37.     }
38. }

```

Paquete: modelo; Clase: MovimientoBaja.java

```

1. package modelo;
2. import util.ConexionBD;
3.
4. import java.sql.Connection;
5. import java.sql.PreparedStatement;
6. import java.sql.SQLException;

```

```

7. import java.sql.Timestamp;
8.
9. public class MovimientoBaja extends Movimiento {
10.     private String motivoBaja;
11.     private String tipoBaja;
12.
13.     public MovimientoBaja(String motivoBaja, String tipoBaja) {
14.         super("Baja");
15.         this.motivoBaja = motivoBaja;
16.         this.tipoBaja = tipoBaja;
17.     }
18.
19.     @Override
20.     public void guardarMovimientoEnBD(int idActivo) {
21.         String sql = "INSERT INTO movimientos (id_activo, tipo_movimiento,
22.         fecha_hora, motivo_baja, tipo_baja) VALUES (?, ?, ?, ?, ?)";
23.         try (Connection conn = ConexionBD.getConnection();
24.             PreparedStatement pstmt = conn.prepareStatement(sql)) {
25.             pstmt.setInt(1, idActivo);
26.             pstmt.setString(2, "Baja");
27.             pstmt.setTimestamp(3, Timestamp.valueOf(fechaHora));
28.             pstmt.setString(4, motivoBaja);
29.             pstmt.setString(5, tipoBaja);
30.             pstmt.executeUpdate();
31.         } catch (SQLException e) {
32.             e.printStackTrace();
33.         }
34.     }

```

Paquete: modelo; Clase: MovimientoModificacion.java

```

1. package modelo;
2.
3. import util.ConexionBD;
4.
5. import java.sql.Connection;
6. import java.sql.PreparedStatement;
7. import java.sql.SQLException;
8. import java.sql.Timestamp;
9.
10. public class MovimientoModificacion extends Movimiento {
11.     private String detalleModificacion;
12.
13.     public MovimientoModificacion(String detalleModificacion) {
14.         super("Modificación");
15.         this.detalleModificacion = detalleModificacion;
16.     }
17.
18.     @Override
19.     public void guardarMovimientoEnBD(int idActivo) {
20.         String sql = "INSERT INTO movimientos (id_activo, tipo_movimiento,
21.         fecha_hora, detalle_general) VALUES (?, ?, ?, ?)";
22.         try (Connection conn = ConexionBD.getConnection();
23.             PreparedStatement pstmt = conn.prepareStatement(sql)) {
24.             pstmt.setInt(1, idActivo);
25.             pstmt.setString(2, "Modificación");
26.             pstmt.setTimestamp(3, Timestamp.valueOf(fechaHora));
27.             pstmt.setString(4, detalleModificacion);
28.             pstmt.executeUpdate();
29.         } catch (SQLException e) {
30.             e.printStackTrace();
31.         }

```

32. }

Paquete: modelo; Clase: MovimientoRegistro.java

```
1. package modelo;
2. import util.ConexionBD;
3.
4. import java.sql.Connection;
5. import java.sql.PreparedStatement;
6. import java.sql.SQLException;
7. import java.sql.Timestamp;
8.
9. public class MovimientoRegistro extends Movimiento {
10.
11.     public MovimientoRegistro() {
12.         super("Registro");
13.     }
14.
15.     @Override
16.     public void guardarMovimientoEnBD(int idActivo) {
17.         String sql = "INSERT INTO movimientos (id_activo, tipo_movimiento,
18. fecha_hora) VALUES (?, ?, ?)";
19.         try (Connection conn = ConexionBD.getConnection();
20.             PreparedStatement pstmt = conn.prepareStatement(sql)) {
21.             pstmt.setInt(1, idActivo);
22.             pstmt.setString(2, "Registro");
23.             pstmt.setTimestamp(3, Timestamp.valueOf(fechaHora));
24.             pstmt.executeUpdate();
25.         } catch (SQLException e) {
26.             e.printStackTrace();
27.         }
28.     }
}
```

Paquete: modelo; Clase: MovimientoRenovacion.java

```
1. package modelo;
2.
3. import util.ConexionBD;
4.
5. import java.sql.Connection;
6. import java.sql.PreparedStatement;
7. import java.sql.SQLException;
8.
9. public class MovimientoRenovacion extends Movimiento {
10.     private String motivo;
11.     private String nuevoCodigoPatrimonial;
12.
13.     public MovimientoRenovacion(String motivo, String nuevoCodigoPatrimonial) {
14.         super("Renovación");
15.         this.motivo = motivo;
16.         this.nuevoCodigoPatrimonial = nuevoCodigoPatrimonial;
17.     }
18.
19.     @Override
20.     public void guardarMovimientoEnBD(int idActivo) {
21.         String sql = "INSERT INTO movimientos (id_activo, tipo_movimiento,
22. fecha_hora, detalle_general) VALUES (?, ?, CURRENT_TIMESTAMP, ?)";
23.         try (Connection conn = ConexionBD.getConnection();
24.             PreparedStatement pstmt = conn.prepareStatement(sql)) {
25.             String detalle = String.format("\nMotivo: %s \nNuevo Código Patrimonial:
% s\n", motivo, nuevoCodigoPatrimonial);
26.         }
27.     }
}
```

```

25.         pstmt.setInt(1, idActivo);
26.         pstmt.setString(2, tipoMovimiento); // "Renovación"
27.         pstmt.setString(3, detalle);
28.         pstmt.executeUpdate();
29.     } catch (SQLException e) {
30.         e.printStackTrace();
31.     }
32. }
33. }
```

Paquete: modelo; Clase: MovimientoRevaluacion.java

```

1. package modelo;
2.
3. import util.ConexionBD;
4.
5. import java.sql.Connection;
6. import java.sql.PreparedStatement;
7. import java.sql.SQLException;
8. import java.sql.Timestamp;
9. import java.time.LocalDate;
10. import java.time.format.DateTimeFormatter;
11.
12. public class MovimientoRevaluacion extends Movimiento {
13.     private double valorAnterior;
14.     private double valorRevaluado;
15.     private String justificacion;
16.
17.     public MovimientoRevaluacion(int idActivo, double valorAnterior, double
18.         valorRevaluado, String justificacion) {
19.         super("Revaluación");
20.         this.valorAnterior = valorAnterior;
21.         this.valorRevaluado = valorRevaluado;
22.         this.justificacion = justificacion;
23.     }
24.
25.     @Override
26.     public void guardarMovimientoEnBD(int idActivo) {
27.         String sql = "INSERT INTO movimientos (id_activo, tipo_movimiento,
28.             fecha_hora, valor_anterior, valor_revaluado, justificacion) " +
29.             "VALUES (?, ?, CURRENT_TIMESTAMP, ?, ?, ?)";
30.         try (Connection conn = ConexionBD.getConnection();
31.             PreparedStatement pstmt = conn.prepareStatement(sql)) {
32.             pstmt.setInt(1, idActivo);
33.             pstmt.setString(2, tipoMovimiento); // "Revaluación"
34.             pstmt.setDouble(3, valorAnterior);
35.             pstmt.setDouble(4, valorRevaluado);
36.             pstmt.setString(5, justificacion);
37.             pstmt.executeUpdate();
38.         } catch (SQLException e) {
39.             e.printStackTrace();
40.         }
41.     }
42. }
```

Paquete: modelo; Clase: MovimientoVenta.java

```

1. package modelo;
2. import util.ConexionBD;
3.
4. import java.sql.*;
5.
```

```

6. public class MovimientoVenta extends Movimiento {
7.     private double precioVenta;
8.     private String detalleVenta;
9.     private String comprador;
10.    private String rucComprador;
11.    private String formaPago;
12.
13.    public MovimientoVenta(int idActivo, double precioVenta, String detalleVenta,
14.                           String comprador, String rucComprador, String formaPago) {
14.        super("Venta");
15.        this.precioVenta = precioVenta;
16.        this.detalleVenta = detalleVenta;
17.        this.comprador = comprador;
18.        this.rucComprador = rucComprador;
19.        this.formaPago = formaPago;
20.    }
21.
22.    @Override
23.    public void guardarMovimientoEnBD(int idActivo) {
24.        Connection conn = null;
25.        PreparedStatement pstmtVentas = null;
26.        PreparedStatement pstmtMovimientos = null;
27.
28.        try {
29.            conn = ConexionBD.getConnection();
30.            conn.setAutoCommit(false);
31.
32.            String sqlVentas = "INSERT INTO ventas (id_activo, comprador,
33.                                         ruc_comprador, fecha_venta, precio_venta, detalle_venta, forma_pago) "
34.                               + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
35.            pstmtVentas = conn.prepareStatement(sqlVentas);
36.            pstmtVentas.setInt(1, idActivo);
37.            pstmtVentas.setString(2, comprador);
38.            pstmtVentas.setString(3, rucComprador);
39.            pstmtVentas.setTimestamp(4, Timestamp.valueOf(fechaHora));
40.            pstmtVentas.setDouble(5, precioVenta);
41.            pstmtVentas.setString(6, detalleVenta);
42.            pstmtVentas.setString(7, formaPago);
43.            pstmtVentas.executeUpdate();
44.
45.            String detalleGeneral = String.format(
46.                "\nComprador: %s\nRUC: %s\nPrecio Venta: %.2f\nForma de Pago:
47.                %s\nDescripción: %s\n",
48.                comprador, rucComprador, precioVenta, formaPago, detalleVenta);
49.
50.            String sqlMovimientos = "INSERT INTO movimientos (id_activo,
51.                                         tipo_movimiento, fecha_hora, detalle_general) "
52.                               + "VALUES (?, ?, ?, ?)";
53.            pstmtMovimientos = conn.prepareStatement(sqlMovimientos);
54.            pstmtMovimientos.setInt(1, idActivo);
55.            pstmtMovimientos.setString(2, "Venta");
56.            pstmtMovimientos.setTimestamp(3, Timestamp.valueOf(fechaHora));
57.            pstmtMovimientos.setString(4, detalleGeneral);
58.            pstmtMovimientos.executeUpdate();
59.
60.            conn.commit();
61.
62.        } catch (SQLException e) {
63.            if (conn != null) {
64.                try {
65.                    conn.rollback();
66.                } catch (SQLException rollbackEx) {
67.                    rollbackEx.printStackTrace();
68.                }
69.            }
70.            e.printStackTrace();
71.        }
72.    }

```

```

68.         } finally {
69.             try {
70.                 if (pststmtVentas != null) pststmtVentas.close();
71.                 if (pststmtMovimientos != null) pststmtMovimientos.close();
72.                 if (conn != null) conn.setAutoCommit(true);
73.             } catch (SQLException ex) {
74.                 ex.printStackTrace();
75.             }
76.         }
77.     }
78. }
```

Paquete: modelo; Clase: Registrable.java

```

1. package modelo;
2.
3. public interface Registrable {
4.     void guardarMovimientoEnBD(int idActivo);
5. }
```

Paquete: modelo; Clase: Sede.java

```

1. package modelo;
2.
3. import java.util.Objects;
4.
5. public class Sede {
6.     private int id;
7.     private String nombre;
8.     private String direccion;
9.
10.    public Sede(int id, String nombre, String direccion) {
11.        this.id = id;
12.        this.nombre = nombre;
13.        this.direccion = direccion;
14.    }
15.
16.    public int getId() { return id; }
17.    public String getNombre() { return nombre; }
18.    public void setNombre(String nombre) { this.nombre = nombre; }
19.
20.    @Override
21.    public String toString() {
22.        return nombre;
23.    }
24.
25.    @Override
26.    public boolean equals(Object o) {
27.        if (this == o) return true;
28.        if (o == null || getClass() != o.getClass()) return false;
29.        Sede sede = (Sede) o;
30.        return Objects.equals(nombre, sede.nombre);
31.    }
32.
33.    @Override
34.    public int hashCode() {
35.        return Objects.hash(nombre);
36.    }
37. }
```

Paquete: modelo; Clase: Usuario.java

```
1. package modelo;
2.
3. public class Usuario {
4.
5.     private String nombreUsuario;
6.     private String password;
7.     private String rol;
8.
9.     public Usuario(String nombreUsuario, String password, String rol) {
10.         this.nombreUsuario = nombreUsuario;
11.         this.password = password;
12.         this.rol = rol;
13.     }
14.
15.     public String getNombreUsuario() {
16.         return nombreUsuario;
17.     }
18. }
```

Paquete: modelo; Clase: UsuarioSesion.java

```
1. package modelo;
2.
3. public class UsuarioSesion {
4.
5.     private static UsuarioSesion instancia;
6.     private Usuario usuarioAutenticado;
7.
8.     private UsuarioSesion() {}
9.
10.    public static UsuarioSesion getInstance() {
11.        if (instancia == null) {
12.            instancia = new UsuarioSesion();
13.        }
14.        return instancia;
15.    }
16.
17.    public void setUsuarioAutenticado(Usuario usuario) {
18.        this.usuarioAutenticado = usuario;
19.    }
20. }
```

Paquete: servicio; Clase: ActivoServicio.java

```
1. package servicio;
2.
3. import modelo.*;
4. import util.ConexionBD;
5. import util.ValidadorFechas;
6. import vista.MainFrame;
7.
8. import java.text.ParseException;
9. import java.text.SimpleDateFormat;
10. import java.time.LocalDate;
11. import java.time.format.DateTimeFormatter;
12. import java.time.temporal.ChronoUnit;
13. import java.util.*;
14. import java.util.Date;
15. import java.util.stream.Collectors;
```

```

16. import java.sql.*;
17. import java.util.List;
18.
19. public class ActivoServicio {
20.     private MainFrame mainFrame;
21.
22.     public ActivoServicio(MainFrame mainFrame) {
23.         this.mainFrame = mainFrame;
24.     }
25.
26.     public ActivoServicio() {
27.     }
28.
29.     public List<ActivoFijo> cargarActivosDesdeBD() {
30.         List<ActivoFijo> activos = new ArrayList<>();
31.         String sql = """
32.             SELECT a.id, a.codigo_patrimonial, a.nombre, a.categoría, a.estado,
33.                 a.fecha_adquisicion, a.vida_util, a.costo_inicial,
34.                 s.id_sede, s.nombre AS nombre_sede, s.direccion
35.             FROM activos a
36.             LEFT JOIN sedes s ON a.id_sede = s.id_sede
37.             ORDER BY a.id
38.         """;;
39.
40.         try (Connection conn = ConexionBD.getConnection();
41.              Statement stmt = conn.createStatement();
42.              ResultSet rs = stmt.executeQuery(sql)) {
43.
44.             while (rs.next()) {
45.                 int id = rs.getInt("id");
46.                 String codigoPatrimonial = rs.getString("codigo_patrimonial");
47.                 String nombre = rs.getString("nombre");
48.                 String categoría = rs.getString("categoría");
49.                 String estado = rs.getString("estado");
50.                 String fechaAdquisicion = rs.getString("fecha_adquisicion");
51.
52.                 String fechaFormateada;
53.                 try {
54.                     Date fecha = new SimpleDateFormat("yyyy-MM-
dd").parse(fechaAdquisicion);
55.                     fechaFormateada = new SimpleDateFormat("dd-MM-
yyyy").format(fecha);
56.                 } catch (Exception e) {
57.                     fechaFormateada = fechaAdquisicion;
58.                 }
59.
60.                 int vidaUtil = rs.getInt("vida_util");
61.                 double costoInicial = rs.getDouble("costo_inicial");
62.
63.                 Sede sede = null;
64.                 if (rs.getInt("id_sede") != 0) {
65.                     sede = new Sede(rs.getInt("id_sede"),
66.                         rs.getString("nombre_sede"), rs.getString("direccion"));
67.                 }
68.
69.                 ActivoFijo activo = new ActivoFijo(nombre, categoría, costoInicial,
70.                     fechaFormateada, "", sede);
71.                 activo.setCodigoPatrimonial(codigoPatrimonial);
72.                 activo.setEstado(estado);
73.                 activo.setVidaUtil(categoría);
74.                 activo.setNumero(id);
75.                 activos.add(activo);
76.             }
77.         } catch (SQLException e) {
78.             e.printStackTrace();
79.         }

```

```

78.         return activos;
79.     }
80.
81.     public ActivoFijo cargarActivoPorId(int id) {
82.         String sql = """
83.             SELECT a.id, a.codigo_patrimonial, a.nombre, a.categoría, a.estado,
84.                 a.fecha_adquisicion, a.vida_util, a.costo_inicial,
85.                 s.id_sede, s.nombre AS nombre_sede, s.direccion
86.             FROM activos a
87.             LEFT JOIN sedes s ON a.id_sede = s.id_sede
88.             WHERE a.id = ?
89.         """;
90.
91.         try (Connection conn = ConexionBD.getConnection();
92.              PreparedStatement pstmt = conn.prepareStatement(sql)) {
93.             pstmt.setInt(1, id);
94.
95.             ResultSet rs = pstmt.executeQuery();
96.             if (rs.next()) {
97.                 String codigoPatrimonial = rs.getString("codigo_patrimonial");
98.                 String nombre = rs.getString("nombre");
99.                 String categoría = rs.getString("categoría");
100.                String estado = rs.getString("estado");
101.                String fechaAdquisicion = rs.getString("fecha_adquisicion");
102.                String fechaFormatada = new SimpleDateFormat("dd-MM-
103.                    yyyy").format(new SimpleDateFormat("yyyy-MM-dd").parse(fechaAdquisicion));
104.                int vidaUtil = rs.getInt("vida_util");
105.                double costoInicial = rs.getDouble("costo_inicial");
106.
107.                // Crear instancia de Sede si existe
108.                Sede sede = null;
109.                if (rs.getInt("id_sede") != 0) {
110.                    sede = new Sede(rs.getInt("id_sede"),
111.                        rs.getString("nombre_sede"), rs.getString("direccion"));
112.
113.                // Crear ActivoFijo con sede
114.                ActivoFijo activo = new ActivoFijo(nombre, categoría,
115.                    costoInicial, fechaFormatada, "", sede);
116.                activo.setCodigoPatrimonial(codigoPatrimonial);
117.                activo.setEstado(estado);
118.                activo.setVidaUtil(String.valueOf(vidaUtil));
119.                return activo;
120.            } catch (Exception e) {
121.                e.printStackTrace();
122.            }
123.        }
124.
125.        public List<String> cargarHistorialMovimientos() {
126.            List<String> historial = new ArrayList<>();
127.            String sql = "SELECT m.tipo_movimiento, m.fecha_hora, m.motivo_baja,
128.                m.tipo_baja, m.precio_venta, m.detalle_general,
129.                + "a.nombre, a.categoría, a.fecha_adquisicion, a.costo_inicial
130.                "
131.                + "FROM movimientos m "
132.                + "JOIN activos a ON m.id_activo = a.id "
133.                + "ORDER BY m.fecha_hora DESC";
134.
135.            try (Connection conn = ConexionBD.getConnection();
136.                 PreparedStatement pstmt = conn.prepareStatement(sql);
137.                 ResultSet rs = pstmt.executeQuery()) {
138.                 while (rs.next()) {
139.                     StringBuilder movimiento = new StringBuilder();

```

```

139.                 movimiento.append("Activo:");
140.                 ".append(rs.getString("nombre")).append(", Categoría:");
141.                 ".append(rs.getString("categoria")).append("\n");
142.                 movimiento.append("Movimiento:");
143.                 ".append(rs.getString("tipo_movimiento")).append("\n");
144.                 movimiento.append("Fecha:");
145.                 ".append(rs.getTimestamp("fecha_hora")).append("\n");
146.                 String motivoBaja = rs.getString("motivo_baja");
147.                 if (motivoBaja != null) movimiento.append("\nMotivo Baja:");
148.                 .append(motivoBaja).append("\n");
149.                 String tipoBaja = rs.getString("tipo_baja");
150.                 if (tipoBaja != null) movimiento.append("Tipo Baja:");
151.                 .append(tipoBaja).append("\n\n");
152.                 double precioVenta = rs.getDouble("precio_venta");
153.                 if (precioVenta != 0.0) movimiento.append("Precio Venta:");
154.                 .append(String.format("%.2f", precioVenta)).append("\n");
155.                 String detalleVenta = rs.getString("detalle_general");
156.                 if (detalleVenta != null)
157.                 movimiento.append("");
158.                 movimiento.append("-----");
159.                 historial.add(movimiento.toString());
160.             }
161.         } catch (SQLException e) {
162.             e.printStackTrace();
163.         }
164.
165.         return historial;
166.     }
167.
168.     public List<Sede> cargarSedesDesdeBD() {
169.         List<Sede> sedes = new ArrayList<>();
170.         String sql = "SELECT id_sede, nombre, direccion FROM sedes";
171.
172.         try (Connection conn = ConexionBD.getConnection();
173.              PreparedStatement pstmt = conn.prepareStatement(sql);
174.              ResultSet rs = pstmt.executeQuery()) {
175.             while (rs.next()) {
176.                 Sede sede = new Sede(rs.getInt("id_sede"),
177.                     rs.getString("nombre"), rs.getString("direccion"));
178.                 sedes.add(sede);
179.             }
180.         } catch (SQLException e) {
181.             e.printStackTrace();
182.         }
183.
184.         return sedes;
185.     }
186.
187.     public Map<Sede, Long> obtenerConteoPorSede() {
188.         List<ActivoFijo> activos = cargarActivosDesdeBD();
189.         return activos.stream()
190.             .filter(activo -> activo.getSede() != null)
191.             .collect(Collectors.groupingBy(ActivoFijo::getSede,

```

```

192.
193.         public GuardarActivoResultado guardarActivoEnBD(String nombre, String
194.                                         categoria, String estado,
195.                                         String
196.                                         fechaAdquisicionStr, int vidaUtil,
197.                                         double depreciacionMensual,
198.                                         double depreciacionAcumulada, double valorResidual,
199.                                         int idSede) throws
200.                                         ParseException {
201.                                         String sql = """
202.                                         INSERT INTO activos (nombre, categoria, estado, fecha_adquisicion,
203.                                         vida_util, costo_inicial,
204.                                         depreciacion_mensual, depreciacion_acumulada,
205.                                         valor_residual,
206.                                         codigo_patrimonial, id_sede)
207.                                         VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
208.                                         """;
209.
210.                                         String fechaFormateada = new SimpleDateFormat("yyyy-MM-dd")
211.                                         .format(new SimpleDateFormat("dd-MM-
212.                                         yyyy").parse(fechaAdquisicionStr));
213.
214.                                         try (Connection conn = ConexionBD.getConnection();
215.                                         PreparedStatement pstmt = conn.prepareStatement(sql,
216.                                         Statement.RETURN_GENERATED_KEYS)) {
217.
218.                                         String codigoPatrimonial =
219.                                         generarCodigoPatrimonial(fechaFormateada, conn);
220.
221.                                         pstmt.setString(1, nombre);
222.                                         pstmt.setString(2, categoria);
223.                                         pstmt.setString(3, estado);
224.                                         pstmt.setString(4, fechaFormateada);
225.                                         pstmt.setInt(5, vidaUtil);
226.                                         pstmt.setDouble(6, costoInicial);
227.                                         pstmt.setDouble(7, depreciacionMensual);
228.                                         pstmt.setDouble(8, depreciacionAcumulada);
229.                                         pstmt.setDouble(9, valorResidual);
230.                                         pstmt.setString(10, codigoPatrimonial);
231.                                         pstmt.setInt(11, idSede);
232.
233.                                         pstmt.executeUpdate();
234.
235.                                         ResultSet rs = pstmt.getGeneratedKeys();
236.                                         if (rs.next()) {
237.                                             int id = rs.getInt(1);
238.                                             registrarNotificacion(id, nombre, fechaAdquisicionStr,
239.                                         vidaUtil, "Pendiente");
240.                                             return new GuardarActivoResultado(id, codigoPatrimonial);
241.                                         }
242.                                         } catch (Exception e) {
243.                                             e.printStackTrace();
244.                                         }
245.
246.                                         return null;
247.                                     }
248.
249.                                     private String generarCodigoPatrimonial(String fechaAdquisicion,
250.                                         Connection conn) {
251.                                         String anio = fechaAdquisicion.split("-")[0];
252.                                         String sql = "SELECT MAX(CAST(SUBSTRING_INDEX(codigo_patrimonial, '-',
253.                                         -1) AS UNSIGNED)) AS max_codigo " +
254.                                         "FROM activos WHERE codigo_patrimonial LIKE ?";
255.                                         int numeroConsecutivo = 1;

```

```

245.
246.        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
247.            pstmt.setString(1, anio + "-%");
248.            ResultSet rs = pstmt.executeQuery();
249.            if (rs.next() && rs.getString("max_codigo") != null) {
250.                numeroConsecutivo = rs.getInt("max_codigo") + 1;
251.            }
252.        } catch (SQLException e) {
253.            e.printStackTrace();
254.        }
255.
256.        String numeroFormateado = String.format("%04d", numeroConsecutivo);
257.
258.        return anio + "-" + numeroFormateado;
259.    }
260.
261.    public void registrarActivo(int idActivo) {
262.        MovimientoRegistro movimientoRegistro = new MovimientoRegistro();
263.        movimientoRegistro.guardarMovimientoEnBD(idActivo);
264.    }
265.
266.    public void modificarActivoEnBD(int id, String nombre, String categoria,
267.        String estado, String fechaAdquisicionStr, int vidaUtil, double costoInicial, double
268.        depreciacionMensual, double depreciacionAcumulada, double valorResidual) {
269.        String sql = "UPDATE activos SET nombre = ?, categoria = ?, estado =
270.        ?, fecha_adquisicion = ?, vida_util = ?, costo_inicial = ?, depreciacion_mensual = ?,
271.        depreciacion_acumulada = ?, valor_residual = ? WHERE id = ?";
272.
273.        String fechaFormateada;
274.        try {
275.            Date fechaAdquisicion = new SimpleDateFormat("dd-MM-
276.            yyyy").parse(fechaAdquisicionStr);
277.            fechaFormateada = new SimpleDateFormat("yyyy-MM-
278.            dd").format(fechaAdquisicion);
279.        } catch (Exception e) {
280.            e.printStackTrace();
281.            return;
282.        }
283.
284.        try (Connection conn = ConexionBD.getConnection();
285.            PreparedStatement pstmt = conn.prepareStatement(sql)) {
286.
287.            pstmt.setString(1, nombre);
288.            pstmt.setString(2, categoria);
289.            pstmt.setString(3, estado);
290.            pstmt.setString(4, fechaFormateada);
291.            pstmt.setInt(5, vidaUtil);
292.            pstmt.setDouble(6, costoInicial);
293.            pstmt.setDouble(7, depreciacionMensual);
294.            pstmt.setDouble(8, depreciacionAcumulada);
295.            pstmt.setDouble(9, valorResidual);
296.            pstmt.setInt(10, id);
297.            pstmt.executeUpdate();
298.
299.            registrarNotificacion(id, nombre, fechaAdquisicionStr, vidaUtil,
300.            "Pendiente");
301.        } catch (SQLException e) {
302.            e.printStackTrace();
303.        }
304.
305.    public List<ActivoFijo> filtrarActivos(Map<String, Object> criterios) {
306.        List<ActivoFijo> activosFiltrados = cargarActivosDesdeBD();
307.
308.        for (Map.Entry<String, Object> criterio : criterios.entrySet()) {
309.            String clave = criterio.getKey();

```

```

304.             Object valor = criterio.getValue();
305.
306.             switch (clave) {
307.                 case "costoMin":
308.                     activosFiltrados = activosFiltrados.stream()
309.                         .filter(activo -> activo.getCostoInicial() >=
310.                               (double) valor)
311.                         .collect(Collectors.toList());
312.                 break;
313.                 case "costoMax":
314.                     activosFiltrados = activosFiltrados.stream()
315.                         .filter(activo -> activo.getCostoInicial() <=
316.                               (double) valor)
317.                         .collect(Collectors.toList());
318.                 break;
319.                 case "fechaInicio":
320.                     activosFiltrados = activosFiltrados.stream()
321.                         .filter(activo -> {
322.                             LocalDate fechaInicio = (LocalDate) valor;
323.                             LocalDate fechaAdquisicion =
324.                                 LocalDate.parse(activo.getFechaAdquisicion(),
325.                                                 DateTimeFormatter.ofPattern("dd-MM-
326.                                                 yyyy")));
327.                         return
328.                             !fechaAdquisicion.isBefore(fechaInicio);
329.                         })
330.                         .collect(Collectors.toList());
331.                 break;
332.                 case "fechaFin":
333.                     activosFiltrados = activosFiltrados.stream()
334.                         .filter(activo -> {
335.                             LocalDate fechaFin = (LocalDate) valor;
336.                             LocalDate fechaAdquisicion =
337.                                 LocalDate.parse(activo.getFechaAdquisicion(),
338.                                                 DateTimeFormatter.ofPattern("dd-MM-
339.                                                 yyyy")));
340.                         return !fechaAdquisicion.isAfter(fechaFin);
341.                         })
342.                         .collect(Collectors.toList());
343.                 break;
344.                 case "categoria":
345.                     activosFiltrados = activosFiltrados.stream()
346.                         .filter(activo ->
347.                             activo.getCategoría().equalsIgnoreCase((String) valor))
348.                         .collect(Collectors.toList());
349.                 break;
350.             }
351.
352.             public void modificarActivo(int idActivo, String detalleModificacion) {
353.                 MovimientoModificacion movimientoModificacion = new
354.                     MovimientoModificacion(detalleModificacion);
355.                     movimientoModificacion.guardarMovimientoEnBD(idActivo);
356.             }
357.             public void darDeBaja(int idActivo, String motivo, String tipoBaja) {
358.                 MovimientoBaja movimientoBaja = new MovimientoBaja(motivo, tipoBaja);
359.                     movimientoBaja.guardarMovimientoEnBD(idActivo);

```

```

360.        }
361.
362.        public void registrarVenta(int idActivo, String comprador, String
363.            rucComprador, double precioVenta, String detalleVenta, String formaPago) {
364.            MovimientoVenta movimientoVenta = new MovimientoVenta(idActivo,
365.                precioVenta, detalleVenta, comprador, rucComprador, formaPago);
366.            movimientoVenta.guardarMovimientoEnBD(idActivo);
367.            actualizarEstadoActivo(idActivo, "Baja por venta");
368.        }
369.
370.        public void actualizarEstadoActivo(int id, String nuevoEstado) {
371.            String sql = "UPDATE activos SET estado = ? WHERE id = ?";
372.
373.            try (Connection conn = ConexionBD.getConnection();
374.                PreparedStatement pstmt = conn.prepareStatement(sql)) {
375.
376.                pstmt.setString(1, nuevoEstado);
377.                pstmt.setInt(2, id);
378.                pstmt.executeUpdate();
379.
380.            } catch (SQLException e) {
381.                e.printStackTrace();
382.            }
383.
384.            private void registrarNotificacion(int idActivo, String nombre, String
385.                fechaAdquisicion, int vidaUtil, String estado) {
386.                String mensaje = "El activo '" + nombre + "' se depreciará en menos de
387.                    6 meses.";
388.
389.                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-
390.                    yyyy");
391.                LocalDate fechaInicioDepreciacion = LocalDate.parse(fechaAdquisicion,
392.                    formatter).plusMonths(1);
393.                LocalDate fechaFinVidaUtil =
394.                    fechaInicioDepreciacion.plusYears(vidaUtil);
395.                LocalDate fechaActual = LocalDate.now();
396.                long mesesRestantes = ChronoUnit.MONTHS.between(fechaActual,
397.                    fechaFinVidaUtil);
398.
399.                if (mesesRestantes <= 6 && !fechaActual.isAfter(fechaFinVidaUtil)) {
400.                    String sql = "INSERT INTO notificaciones (id_activo, mensaje,
401.                        estado, fecha_creacion) VALUES (?, ?, ?, CURRENT_TIMESTAMP) "
402.                            + "ON DUPLICATE KEY UPDATE mensaje = VALUES(mensaje), estado
403.                                = VALUES(estado), fecha_creacion = CURRENT_TIMESTAMP";
404.
405.                    try (Connection conn = ConexionBD.getConnection();
406.                        PreparedStatement pstmt = conn.prepareStatement(sql)) {
407.
408.                        pstmt.setInt(1, idActivo);
409.                        pstmt.setString(2, mensaje);
410.                        pstmt.setString(3, estado);
411.                        pstmt.executeUpdate();
412.
413.                        if (mainFrame != null) {
414.                            mainFrame.mostrarNotificacionEmergente(mensaje, idActivo);
415.                        }
416.
417.                    } catch (SQLException e) {
418.                        e.printStackTrace();
419.                    }
420.                }
421.
422.                public void renovarActivo(int idActivo, String motivo) {
423.                    ActivoFijo activo = cargarActivoPorId(idActivo);
424.                    if (activo == null) {
425.                        throw new IllegalArgumentException("El activo no existe.");
426.                    }
427.
428.                }
429.
430.            }
431.
432.        }
433.
434.    }
435.
```

```

416.         }
417.
418.         if (!activo.getEstado().equalsIgnoreCase("Depreciado") &&
419.             !activo.getEstado().equalsIgnoreCase("Baja") &&
420.             !activo.getEstado().equalsIgnoreCase("Baja por venta")) {
421.             throw new IllegalArgumentException("Solo se pueden renovar activos
422. en estado 'Depreciado', 'Baja' o 'Baja por venta'.");
423.         }
424.         actualizarEstadoActivo(idActivo, "Renovado");
425.
426.         String nuevoCodigo;
427.         try (Connection conn = ConexionBD.getConnection()) {
428.             ActivoFijo nuevoActivo = new ActivoFijo(
429.                 activo.getNombre(),
430.                 activo.getCategoría(),
431.                 activo.getCostoInicial(),
432.                 LocalDate.now().format(DateTimeFormatter.ofPattern("dd-MM-
433.                 yyyy")),
434.                 "Administrador",
435.                 activo.getSede()
436.             );
437.             nuevoCodigo =
438.                 generarCodigoPatrimonial(LocalDate.now().format(DateTimeFormatter.ofPattern("yyyy-MM-
439.                 dd")), conn);
440.             nuevoActivo.setCodigoPatrimonial(nuevoCodigo);
441.             guardarActivoEnBD(
442.                 nuevoActivo.getNombre(),
443.                 nuevoActivo.getCategoría(),
444.                 "Alta",
445.                 nuevoActivo.getFechaAdquisicion(),
446.                 Integer.parseInt(String.valueOf(activo.getVidaUtil())),
447.                 nuevoActivo.getCostoInicial(),
448.                 nuevoActivo.calcularDepreciacionMensual(),
449.                 0.0,
450.                 nuevoActivo.calcularValorResidual(),
451.                 nuevoActivo.getSede().getId()
452.             );
453.             MovimientoRenovacion movimiento = new MovimientoRenovacion(motivo,
454.                 nuevoCodigo);
455.             movimiento.guardarMovimientoEnBD(idActivo);
456.         } catch (SQLException | ParseException e) {
457.             throw new RuntimeException("Error al generar el código patrimonial
458. o guardar el activo renovado.", e);
459.         }
460.     }
461.     public int contarActivosPorEstado(String estado) {
462.         String sql = "SELECT COUNT(*) AS total FROM activos WHERE estado = ?";
463.         try (Connection conn = ConexionBD.getConnection();
464.             PreparedStatement pstmt = conn.prepareStatement(sql)) {
465.             pstmt.setString(1, estado);
466.             ResultSet rs = pstmt.executeQuery();
467.             if (rs.next()) {
468.                 return rs.getInt("total");
469.             }
470.         } catch (SQLException e) {
471.             e.printStackTrace();
472.         }
473.         return 0;
474.     }

```

Paquete: servicio; Clase: RevaluacionServicio.java

```
1. package servicio;
2.
3. import util.ConexionBD;
4.
5. import java.sql.*;
6. import java.util.ArrayList;
7. import java.util.List;
8.
9. public class RevaluacionServicio {
10.
11.     public RevaluacionServicio() {
12.     }
13.
14.     public List<Object[]> cargarDatosDesdeBD() {
15.         List<Object[]> datos = new ArrayList<>();
16.         String query = """
17.             SELECT
18.                 a.id AS id_activo,
19.                 a.codigo_patrimonial,
20.                 a.nombre,
21.                 a.categoría,
22.                 a.estado,
23.                 COALESCE(r.valor_anterior, a.costo_inicial) AS valor_anterior,
24.                 COALESCE(r.valor_revaluado, '') AS valor_revaluado,
25.                 COALESCE(r.depreciacion_anterior, a.depreciacion_mensual) AS
26.                 depreciacion_anterior, -- Aquí se modifica para incluir la depreciación anterior
27.                 COALESCE(r.nueva_depreciacion, '') AS nueva_depreciacion,
28.                 COALESCE(r.impacto_total, 0) AS impacto_total
29.             FROM activos a
30.             LEFT JOIN movimientos m ON a.id = m.id_activo AND m.tipo_movimiento =
31.                 'Revaluación'
32.             LEFT JOIN revaluaciones r ON r.id_movimiento = m.id
33.             WHERE a.estado NOT IN ('Depreciado', 'Baja', 'Baja por venta', 'Renovado') --
34.                 Excluir activos con estados no deseados
35.                 """;;
36.
37.         try (Connection conn = ConexionBD.getConnection();
38.              PreparedStatement pstmt = conn.prepareStatement(query);
39.              ResultSet rs = pstmt.executeQuery()) {
40.
41.             while (rs.next()) {
42.                 double valorAnterior = rs.getDouble("valor_anterior");
43.                 double valorRevaluado = rs.getDouble("valor_revaluado");
44.                 double cambioAbsoluto = valorRevaluado - valorAnterior;
45.
46.                 Object[] row = {
47.                     rs.getInt("id_activo"),
48.                     rs.getString("nombre"),
49.                     rs.getString("categoría"),
50.                     valorAnterior,
51.                     valorRevaluado == 0 ? "" : valorRevaluado,
52.                     rs.getDouble("depreciacion_anterior"),
53.                     rs.getString("nueva_depreciacion"),
54.                     cambioAbsoluto,
55.                     String.format("%.2f%%", impactoTotal)
56.                 };
57.                 datos.add(row);
58.             }
59.         } catch (SQLException e) {
```

```

59.             e.printStackTrace();
60.         }
61.
62.         return datos;
63.     }
64.
65.     public void guardarRevaluacionEnBD(int idActivo, double valorRevaluado, String
66.                                         motivo) {
67.         try (Connection conn = ConexionBD.getConnection()) {
68.             PreparedStatement pstmtCheck = conn.prepareStatement(
69.                 "SELECT r.id FROM revaluaciones r " +
70.                 "INNER JOIN movimientos m ON r.id_movimiento = m.id " +
71.                 "WHERE m.id_activo = ? AND m.tipo_movimiento =
72.                     'Revaluación''");
73.             pstmtCheck.setInt(1, idActivo);
74.             ResultSet rsCheck = pstmtCheck.executeQuery();
75.
76.             double valorAnterior = obtenerValorAnterior(conn, idActivo);
77.             double depreciacionAnterior = obtenerDepreciacionAnterior(conn,
78.                 idActivo);
79.             double depreciacionMensual = calcularDepreciacionMensual(valorRevaluado,
80.                 idActivo, conn);
81.             double depreciacionAcumulada =
82.                 calcularDepreciacionAcumulada(valorRevaluado, idActivo, conn);
83.             double valorResidual = calcularValorResidual(valorRevaluado,
84.                 depreciacionAcumulada);
85.
86.             double impactoTotal = calcularImpactoPorcentual(valorAnterior,
87.                 valorRevaluado);
88.
89.             if (rsCheck.next()) {
90.                 int idRevaluacion = rsCheck.getInt("id");
91.                 PreparedStatement pstmtUpdate = conn.prepareStatement(
92.                     "UPDATE revaluaciones SET valor_revaluado = ?,"
93.                     "depreciacion_anterior = ?, nueva_depreciacion = ?, impacto_total = ? WHERE id = ?");
94.                 );
95.
96.                 pstmtUpdate.setDouble(1, valorRevaluado);
97.                 pstmtUpdate.setDouble(2, depreciacionAnterior);
98.                 pstmtUpdate.setDouble(3, depreciacionMensual);
99.                 pstmtUpdate.setDouble(4, impactoTotal);
100.                pstmtUpdate.setInt(5, idRevaluacion);
101.                pstmtUpdate.executeUpdate();
102.
103.                PreparedStatement pstmtMovimientoUpdate = conn.prepareStatement(
104.                    "UPDATE movimientos SET detalle_general = ? WHERE id ="
105.                    "(SELECT id_movimiento FROM revaluaciones WHERE id = ?)");
106.
107.                String detalleGeneral = String.format(
108.                    "\nMotivo: %s\nCosto Anterior: %.2f\nCosto Revaluado:
109.                    %.2f\nDepreciación Anterior: %.2f" +
110.                    "\nDepreciación Revaluada: %.2f\nImpacto
111.                    Total: %.2f%\n",
112.                    motivo, valorAnterior, valorRevaluado,
113.                    depreciacionAnterior, depreciacionMensual, impactoTotal
114.                    );
115.                pstmtMovimientoUpdate.setString(1, detalleGeneral);
116.                pstmtMovimientoUpdate.setInt(2, idRevaluacion);
117.                pstmtMovimientoUpdate.executeUpdate();
118.
119.            } else {
120.                PreparedStatement pstmtMovimiento = conn.prepareStatement(
121.                    "INSERT INTO movimientos (id_activo, tipo_movimiento,
122.                    fecha_hora, detalle_general) " +

```

```

112.                                "VALUES (?, 'Revaluación', NOW(), ?)" ,
113.                                Statement.RETURN_GENERATED_KEYS
114.                            );
115.
116.                                String detalleGeneral = String.format(
117.                                    "\nMotivo: %s\nCosto Anterior: %.2f\nCosto Revaluado:
118.                                     %.2f\nDepreciación Anterior: %.2f" +
119.                                         "\nDepreciación Revaluada: %.2f\nImpacto
120.                                         Total: %.2f%%\n",
121.                                         motivo, valorAnterior, valorRevaluado,
122.                                         depreciacionAnterior, depreciacionMensual, impactoTotal
123.                                         );
124.
125.                                pstmtMovimiento.setInt(1, idActivo);
126.                                pstmtMovimiento.setString(2, detalleGeneral);
127.                                pstmtMovimiento.executeUpdate();
128.
129.                                ResultSet rsMovimiento = pstmtMovimiento.getGeneratedKeys();
130.                                if (rsMovimiento.next()) {
131.                                    int idMovimiento = rsMovimiento.getInt(1);
132.
133.                                    PreparedStatement pstmtRevaluacion =
134.                                         conn.prepareStatement(
135.                                             "INSERT INTO revaluaciones (id_movimiento,
136.                                             valor_anterior, valor_revaluado, depreciacion_anterior, nueva_depreciacion,
137.                                             impacto_total) " +
138.                                                 "VALUES (?, ?, ?, ?, ?, ?, ?)"
139.                                         );
140.                                    pstmtRevaluacion.setInt(1, idMovimiento);
141.                                    pstmtRevaluacion.setDouble(2, valorAnterior);
142.                                    pstmtRevaluacion.setDouble(3, valorRevaluado);
143.                                    pstmtRevaluacion.setDouble(4, depreciacionAnterior);
144.                                    pstmtRevaluacion.setDouble(5, depreciacionMensual);
145.                                    pstmtRevaluacion.setDouble(6, impactoTotal);
146.                                    pstmtRevaluacion.executeUpdate();
147.
148.                                }
149.
150.                                PreparedStatement pstmtUpdateActivo = conn.prepareStatement(
151.                                    "UPDATE activos SET costo_inicial = ?,
152.                                     depreciacion_mensual = ?, depreciacion_acumulada = ?, valor_residual = ? WHERE id =
153.                                     ?"
154.                                );
155.                                pstmtUpdateActivo.setDouble(1, valorRevaluado);
156.                                pstmtUpdateActivo.setDouble(2, depreciacionMensual);
157.                                pstmtUpdateActivo.setDouble(3, depreciacionAcumulada);
158.                                pstmtUpdateActivo.setDouble(4, valorResidual);
159.                                pstmtUpdateActivo.setInt(5, idActivo);
160.                                pstmtUpdateActivo.executeUpdate();
161.
162.                            } catch (SQLException e) {
163.                                e.printStackTrace();
164.                            }
165.
166.                            private double obtenerDepreciacionAnterior(Connection conn, int idActivo)
167.                            throws SQLException {
168.                                PreparedStatement pstmt = conn.prepareStatement("SELECT
169.                                     depreciacion_mensual FROM activos WHERE id = ?");
170.                                pstmt.setInt(1, idActivo);
171.                                ResultSet rs = pstmt.executeQuery();
172.                                return rs.next() ? rs.getDouble("depreciacion_mensual") : 0.0;
173.                            }
174.
175.                            public double calcularImpactoPorcentual(double valorAnterior, double
176.                                         valorRevaluado) {

```

```

167.             if (valorAnterior > 0) {
168.                 return ((valorRevaluado - valorAnterior) / Math.max(valorAnterior,
169.                             valorRevaluado) * 100);
170.             }
171.         }
172.
173.         private double obtenerValorAnterior(Connection conn, int idActivo) throws
174.             SQLException {
175.             PreparedStatement pstmt = conn.prepareStatement("SELECT costo_inicial
176.                 FROM activos WHERE id = ?");
177.             pstmt.setInt(1, idActivo);
178.             ResultSet rs = pstmt.executeQuery();
179.             return rs.next() ? rs.getDouble("costo_inicial") : 0.0;
180.         }
181.
182.         private double calcularDepreciacionMensual(double costoInicial, int
183.             idActivo, Connection conn) throws SQLException {
184.             String query = "SELECT vida_util FROM activos WHERE id = ?";
185.             PreparedStatement pstmt = conn.prepareStatement(query);
186.             pstmt.setInt(1, idActivo);
187.             ResultSet rs = pstmt.executeQuery();
188.             if (rs.next()) {
189.                 int vidaUtil = rs.getInt("vida_util");
190.                 if (vidaUtil == 0) return 0.0;
191.                 return costoInicial / (vidaUtil * 12);
192.             }
193.             return 0.0;
194.
195.         private double calcularDepreciacionAcumulada(double costoInicial, int
196.             idActivo, Connection conn) throws SQLException {
197.             String query = """
198.                 SELECT
199.                     TIMESTAMPDIFF(MONTH, DATE_ADD(fecha_adquisicion, INTERVAL 1
200.                         MONTH), CURDATE()) AS meses_transcurridos,
201.                     vida_util
202.                 FROM activos WHERE id = ?
203.             """;;
204.             PreparedStatement pstmt = conn.prepareStatement(query);
205.             pstmt.setInt(1, idActivo);
206.             ResultSet rs = pstmt.executeQuery();
207.             if (rs.next()) {
208.                 int mesesTranscurridos = rs.getInt("meses_transcurridos");
209.                 int vidaUtilMeses = rs.getInt("vida_util") * 12;
210.                 return Math.min(costoInicial / vidaUtilMeses * mesesTranscurridos,
211.                     costoInicial);
212.             }
213.             return 0.0;
214.         }
215.
216.         private double calcularValorResidual(double costoInicial, double
217.             depreciacionAcumulada) {
218.             return Math.max(0, costoInicial - depreciacionAcumulada);
219.         }
220.     }

```

Paquete: util; Clase: ConexionBD.java

```

1. package util;
2.
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.SQLException;

```

```

6.
7. public class ConexionBD {
8.     private static final String URL = "jdbc:mysql://localhost:3306/valorium_db";
9.     private static final String USER = "root";
10.    private static final String PASSWORD = "root";
11.
12.    public static Connection getConnection() throws SQLException {
13.        return DriverManager.getConnection(URL, USER, PASSWORD);
14.    }
15. }

```

Paquete: util; Clase: ImpactoTotalCellRenderer.java

```

1. package util;
2.
3. import java.awt.*;
4. import javax.swing.*;
5. import javax.swing.table.DefaultTableCellRenderer;
6.
7. public class ImpactoTotalCellRenderer extends DefaultTableCellRenderer {
8.     @Override
9.     public Component getTableCellRendererComponent(JTable table, Object value,
10.         boolean isSelected, boolean hasFocus, int row, int column) {
11.         Component cell = super.getTableCellRendererComponent(table, value,
12.             isSelected, hasFocus, row, column);
13.
14.         if (value != null) {
15.             try {
16.                 String porcentajeStr = value.toString().replace("%", "");
17.                 double porcentaje = Double.parseDouble(porcentajeStr);
18.
19.                 if (porcentaje >= 50) {
20.                     cell.setBackground(new Color(144, 238, 144));
21.                 } else if (porcentaje >= 20) {
22.                     cell.setBackground(new Color(173, 255, 47));
23.                 } else if (porcentaje >= 10) {
24.                     cell.setBackground(new Color(255, 255, 153));
25.                 } else if (porcentaje >= 0) {
26.                     cell.setBackground(new Color(255, 204, 153));
27.                 }
28.
29.                 cell.setForeground(Color.DARK_GRAY);
30.             } catch (NumberFormatException e) {
31.                 cell.setBackground(Color.WHITE);
32.                 cell.setForeground(Color.BLACK);
33.             }
34.         }
35.
36.         return cell;
37.     }
38. }
39. }

```

Paquete: util; Clase: NonEditableTableModel.java

```

1. package util;
2.
3. import javax.swing.table.DefaultTableModel;
4.
5. public class NonEditableTableModel extends DefaultTableModel {

```

```

6.
7.     @Override
8.     public boolean isCellEditable(int row, int column) {
9.         return false;
10.    }
11. }
```

Paquete: util; Clase: ValidadorActivo.java

```

1. package util;
2.
3. import com.toedter.calendar.JDateChooser;
4. import javax.swing.JComboBox;
5.
6. public class ValidadorActivo {
7.
8.     public static String validarNombre(String nombre) {
9.         if (nombre == null || nombre.length() < 3) {
10.             return "El nombre debe tener al menos 3 caracteres.";
11.        }
12.        return null;
13.    }
14.
15.    public static String validarCategoria(JComboBox<String> comboCategoria) {
16.        String categoria = (String) comboCategoria.getSelectedItem();
17.        if (categoria == null || categoria.isEmpty()) {
18.            return "Seleccione una categoría para el activo.";
19.        }
20.        return null;
21.    }
22.
23.    public static String validarCostoInicial(String costoInicialStr) {
24.        try {
25.            double costoInicial = Double.parseDouble(costoInicialStr);
26.            if (costoInicial <= 0) {
27.                return "El costo inicial debe ser mayor a 0.";
28.            }
29.            if (costoInicial > 100_000_000) {
30.                return "El costo inicial no puede ser mayor a S/ 100,000,000.";
31.            }
32.        } catch (NumberFormatException e) {
33.            return "Formato incorrecto para el costo inicial. Debe ser un número.";
34.        }
35.        return null;
36.    }
37.
38.    public static String validarFechaAdquisicion(JDateChooser dateChooser) {
39.        return ValidadorFechas.validarFechaJDateChooser(dateChooser);
40.    }
41. }
```

Paquete: util; Clase: ValidadorFechas.java

```

1. package util;
2.
3. import com.toedter.calendar.JDateChooser;
4. import java.text.SimpleDateFormat;
5. import java.time.LocalDate;
6. import java.util.Date;
7.
8. public class ValidadorFechas {
```

```

10.    public static String validarFecha(String fechaAdquisicion) {
11.        String[] partesFecha = fechaAdquisicion.split("-");
12.        if (partesFecha.length != 3) {
13.            return "Fecha en formato incorrecto. Use DD-MM-AAAA.";
14.        }
15.
16.        try {
17.            int dia = Integer.parseInt(partesFecha[0]);
18.            int mes = Integer.parseInt(partesFecha[1]);
19.            int anio = Integer.parseInt(partesFecha[2]);
20.
21.            LocalDate fechaIngresada = LocalDate.of(anio, mes, dia);
22.            LocalDate fechaActual = LocalDate.now();
23.
24.            if (anio < 1997 || fechaIngresada.isAfter(fechaActual)) {
25.                return "La fecha debe ser entre 1997 y la fecha actual.";
26.            }
27.
28.            if (!esDiaValido(dia, mes, anio)) {
29.                return "El día ingresado no es válido.";
30.            }
31.
32.            return null;
33.
34.        } catch (Exception e) {
35.            return "Fecha en formato incorrecto. Use DD-MM-AAAA.";
36.        }
37.    }
38.
39.    public static boolean esDiaValido(int dia, int mes, int anio) {
40.        int[] diasPorMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
41.        if (mes == 2 && esAnioBisiesto(anio)) {
42.            diasPorMes[1] = 29;
43.        }
44.        return dia > 0 && dia <= diasPorMes[mes - 1];
45.    }
46.
47.    public static boolean esAnioBisiesto(int anio) {
48.        return (anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0);
49.    }
50.
51.    public static String validarFechaJDateChooser(JDateChooser dateChooser) {
52.        Date fecha = dateChooser.getDate();
53.        if (fecha == null) {
54.            return "La fecha de adquisición no puede estar vacía.";
55.        }
56.
57.        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
58.        String fechaStr = dateFormat.format(fecha);
59.
60.        return validarFecha(fechaStr);
61.    }
}

```

Paquete: util; Clase: ValidadorVentas.java

```

1. package util;
2.
3. import javax.swing.*;
4. import java.util.regex.Pattern;
5.
6. public class ValidadorVentas {
7.
8.     public static boolean validarComprador(JTextField txtComprador) {
9.         if (txtComprador.getText().trim().isEmpty()) {

```

```

10.         JOptionPane.showMessageDialog(null, "El campo 'Comprador' no puede estar
11.           vacío.", "Error", JOptionPane.ERROR_MESSAGE);
12.       }
13.     return true;
14.   }
15.
16.   public static boolean validarRUCComprador(JTextField txtRUCComprador) {
17.     String ruc = txtRUCComprador.getText().trim();
18.     if (ruc.isEmpty() || !Pattern.matches("\d{11}", ruc)) {
19.       JOptionPane.showMessageDialog(null, "El RUC del comprador debe tener 11
20.         dígitos.", "Error", JOptionPane.ERROR_MESSAGE);
21.     }
22.     return true;
23.   }
24.
25.   public static boolean validarDetallesVenta(JTextArea txtDetalleVenta) {
26.     if (txtDetalleVenta.getText().trim().isEmpty()) {
27.       JOptionPane.showMessageDialog(null, "El campo 'Detalles de la Venta' no
28.         puede estar vacío.", "Error", JOptionPane.ERROR_MESSAGE);
29.     }
30.     return true;
31.   }
32.
33.   public static boolean validarPrecioVenta(JTextField txtPrecioVenta, double
34.     valorResidual) {
35.     String precioVentaStr = txtPrecioVenta.getText().trim();
36.     if (precioVentaStr.isEmpty()) {
37.       JOptionPane.showMessageDialog(null, "El campo 'Precio de Venta' no puede
38.         estar vacío.", "Error", JOptionPane.ERROR_MESSAGE);
39.     }
40.     try {
41.       double precioVenta = Double.parseDouble(precioVentaStr);
42.       if (precioVenta <= 0) {
43.         JOptionPane.showMessageDialog(null, "El 'Precio de Venta' debe ser un
44.           valor positivo.", "Error", JOptionPane.ERROR_MESSAGE);
45.       }
46.
47.       if (precioVenta >= valorResidual) {
48.         JOptionPane.showMessageDialog(null, "El 'Precio de Venta' debe ser
49.           menor al valor residual del activo.", "Error", JOptionPane.ERROR_MESSAGE);
50.       }
51.     } catch (NumberFormatException e) {
52.       JOptionPane.showMessageDialog(null, "El 'Precio de Venta' debe ser un
53.         número válido.", "Error", JOptionPane.ERROR_MESSAGE);
54.     }
55.
56.     return true;
57.   }
58.
59.   public static boolean validarFormularioVenta(JTextField txtComprador, JTextField
60.     txtRUCComprador, JTextArea txtDetalleVenta, JTextField txtPrecioVenta, double
61.     valorResidual) {
62.     return validarComprador(txtComprador) &&
63.           validarRUCComprador(txtRUCComprador) &&
64.           validarDetallesVenta(txtDetalleVenta) &&
65.           validarPrecioVenta(txtPrecioVenta, valorResidual);
66.   }

```

Paquete: util; Clase: VentanaUtils.java

```
1. package util;
2.
3. import javax.swing.*;
4. import java.awt.event.WindowEvent;
5. import java.awt.event.WindowStateListener;
6.
7. public class VentanaUtils {
8.
9.     public static void deshabilitarMinimizar(JFrame frame) {
10.         try {
11.             frame.addWindowStateListener(new WindowStateListener() {
12.                 @Override
13.                 public void windowStateChanged(WindowEvent e) {
14.                     if (e.getNewState() == JFrame.ICONIFIED) {
15.                         frame.setExtendedState(JFrame.NORMAL);
16.                     }
17.                 }
18.             });
19.
20.         } catch (Exception e) {
21.             JOptionPane.showMessageDialog(frame, "Error al deshabilitar minimizar: "
+ e.getMessage(),
22.                                         "Error", JOptionPane.ERROR_MESSAGE);
23.         }
24.     }
25. }
```

Paquete: vista; Clase: AnalisisDeBajasFrame.java

```
1. package vista;
2.
3. import org.jfree.chart.ChartFactory;
4. import org.jfree.chart.ChartPanel;
5. import org.jfree.chart.JFreeChart;
6. import org.jfree.chart.plot.CategoryPlot;
7. import org.jfree.chart.renderer.category.BarRenderer;
8. import org.jfree.data.category.DefaultCategoryDataset;
9.
10. import javax.swing.*;
11. import javax.swing.border.EmptyBorder;
12. import javax.swing.table.DefaultTableModel;
13. import java.awt.*;
14. import java.sql.Connection;
15. import java.sql.PreparedStatement;
16. import java.sql.ResultSet;
17. import java.sql.SQLException;
18. import java.util.List;
19. import modelo.ActivoFijo;
20. import util.ConexionBD;
21. import util.VentanaUtils;
22.
23. public class AnalisisDeBajasFrame extends JFrame {
24.
25.     public AnalisisDeBajasFrame(List<ActivoFijo> activos) {
26.         setTitle("Reporte de Activos en Baja");
27.         setSize(1000, 700);
28.         setLocationRelativeTo(null);
29.         VentanaUtils.deshabilitarMinimizar(this);
30.         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
31.         setResizable(false);
```

```

32.         setLayout(new BorderLayout(10, 10));
33.
34.         List<ActivoFijo> activosFiltrados = activos.stream()
35.             .filter(a -> !a.getEstado().equalsIgnoreCase("Renovado"))
36.             .toList();
37.
38.         // Encabezado
39.         JPanel encabezado = crearPanelEncabezado(activosFiltrados);
40.         add(encabezado, BorderLayout.NORTH);
41.
42.         // Tabla de activos
43.         JTable tablaActivos = crearTablaActivos(activosFiltrados);
44.         JScrollPane scrollPane = new JScrollPane(tablaActivos);
45.         scrollPane.setBorder(new EmptyBorder(10, 10, 10, 10));
46.         add(scrollPane, BorderLayout.CENTER);
47.
48.         // Panel de gráficos
49.         JTabbedPane panelGraficos = crearPanelGraficos(activosFiltrados);
50.         add(panelGraficos, BorderLayout.SOUTH);
51.     }
52.
53.     private JPanel crearPanelEncabezado(List<ActivoFijo> activos) {
54.         JPanel panel = new JPanel(new GridLayout(1, 3, 10, 10));
55.         panel.setBorder(new EmptyBorder(10, 10, 10, 10));
56.
57.         double perdidaTotal = activos.stream()
58.             .filter(a -> a.getEstado().equalsIgnoreCase("Baja") ||
a.getEstado().equalsIgnoreCase("Baja por venta"))
59.             .mapToDouble(a -> a.getCostoInicial() - a.calcularValorResidual())
60.             .sum();
61.
62.         double montoRecuperado = activos.stream()
63.             .filter(a -> a.getEstado().equalsIgnoreCase("Baja por venta"))
64.             .mapToDouble(a -> obtenerPrecioVentaDesdeBD(a.getNumero()))
65.             .sum();
66.
67.         double perdidaNeta = perdidaTotal - montoRecuperado;
68.
69.         // Etiquetas
70.         JLabel labelPerdidaTotal = new JLabel("Pérdida Bruta: S/ " +
String.format("%.2f", perdidaTotal), JLabel.CENTER);
71.         JLabel labelMontoRecuperado = new JLabel("Monto Recuperado: S/ " +
String.format("%.2f", montoRecuperado), JLabel.CENTER);
72.         JLabel labelPerdidaNeta = new JLabel("Pérdida Neta: S/ " +
String.format("%.2f", Math.max(perdidaNeta, 0)), JLabel.CENTER);
73.
74.         labelPerdidaTotal.setFont(new Font("Arial", Font.BOLD, 16));
75.         labelMontoRecuperado.setFont(new Font("Arial", Font.BOLD, 16));
76.         labelPerdidaNeta.setFont(new Font("Arial", Font.BOLD, 16));
77.
78.         panel.add(labelPerdidaTotal);
79.         panel.add(labelMontoRecuperado);
80.         panel.add(labelPerdidaNeta);
81.
82.         return panel;
83.     }
84.
85.     private JTable crearTablaActivos(List<ActivoFijo> activos) {
86.         String[] columnas = {"ID", "Nombre", "Categoría", "Estado", "Costo Inicial",
"Depreciación Acumulada", "Valor Residual", "Pérdida Neta"};
87.         DefaultTableModel modelo = new DefaultTableModel(columnas, 0) {
88.             @Override
89.             public boolean isCellEditable(int row, int column) {
90.                 return false;
91.             }
92.         };

```

```

93.         for (ActivoFijo activo : activos) {
94.             if (activo.getEstado().equalsIgnoreCase("Baja") ||
95.                 activo.getEstado().equalsIgnoreCase("Baja por venta")) {
96.                     double depreciacionAcumulada =
97.                         activo.calcularDepreciacionAcumulada();
98.                     double valorResidual = activo.calcularValorResidual();
99.                     double perdidaNeta = activo.getCostoInicial() - valorResidual;
100.
101.                     modelo.addRow(new Object[]{
102.                         activo.getNumero(),
103.                         activo.getNombre(),
104.                         activo.getCategoría(),
105.                         activo.getEstado(),
106.                         String.format("%.2f", activo.getCostoInicial()),
107.                         String.format("%.2f", depreciacionAcumulada),
108.                         String.format("%.2f", valorResidual),
109.                         String.format("%.2f", Math.max(0, perdidaNeta))
110.                     });
111.                 }
112.             JTable tabla = new JTable(modelo);
113.             tabla.getTableHeader().setReorderingAllowed(false);
114.             return tabla;
115.         }
116.     private JTabbedPane crearPanelGraficos(List<ActivoFijo> activos) {
117.         JTabbedPane panel = new JTabbedPane();
118.         panel.addTab("Pérdidas por Categoría", crearGraficoBarras(activos));
119.
120.         boolean hayBajasPorVenta = activos.stream()
121.             .anyMatch(a -> a.getEstado().equalsIgnoreCase("Baja por
122. venta"));
123.
124.         if (hayBajasPorVenta) {
125.             panel.addTab("Análisis de Ventas", crearGraficoVentas(activos));
126.         }
127.
128.         return panel;
129.     }
130.     private JPanel crearGraficoBarras(List<ActivoFijo> activos) {
131.         DefaultCategoryDataset dataset = new DefaultCategoryDataset();
132.         for (ActivoFijo activo : activos) {
133.             if (activo.getEstado().equalsIgnoreCase("Baja") ||
134.                 activo.getEstado().equalsIgnoreCase("Baja por venta")) {
135.                     double perdidaNeta = activo.getCostoInicial() -
136.                         activo.calcularValorResidual();
137.                     if (perdidaNeta > 0) {
138.                         dataset.addValue(perdidaNeta, activo.getCategoría(),
139.                             activo.getNombre());
140.                     }
141.                 }
142.             }
143.             JFreeChart chart = ChartFactory.createBarChart(
144.                 "Pérdidas por Categoría",
145.                 "Categoría",
146.                 "Monto (S/)",
147.                 dataset,
148.                 org.jfree.chart.plot.PlotOrientation.VERTICAL,
149.                 true,
150.                 true,
151.                 false
152.             );
153.             chart.getTitle().setMargin(20, 0, 20, 0);

```

```

153.         CategoryPlot plot = chart.getCategoryPlot();
154.         BarRenderer renderer = (BarRenderer) plot.getRenderer();
155.
156.         Color[] colores = {
157.             new Color(112, 173, 71),
158.             new Color(255, 192, 0),
159.             new Color(237, 125, 49),
160.             new Color(91, 155, 213),
161.             new Color(193, 193, 193)
162.         };
163.
164.         for (int i = 0; i < dataset.getRowCount(); i++) {
165.             renderer.setSeriesPaint(i, colores[i % colores.length]);
166.         }
167.
168.         renderer.setBarPainter(new BarRenderer().getBarPainter());
169.         renderer.setMaximumBarWidth(0.1);
170.
171.         return new ChartPanel(chart);
172.     }
173.
174.     private JPanel crearGraficoVentas(List<ActivoFijo> activos) {
175.         DefaultCategoryDataset dataset = new DefaultCategoryDataset();
176.
177.         for (ActivoFijo activo : activos) {
178.             if (activo.getEstado().equalsIgnoreCase("Baja por venta")) {
179.                 double valorResidual = activo.calcularValorResidual();
180.                 double precioVenta =
181.                     obtenerPrecioVentaDesdeBD(activo.getNumero());
182.                 dataset.addValue(valorResidual, "Valor Residual",
183.                     activo.getNombre());
184.                 dataset.addValue(precioVenta, "Precio de Venta",
185.                     activo.getNombre());
186.             }
187.         }
188.         JFreeChart chart = ChartFactory.createBarChart(
189.             "Análisis de Ventas",
190.             "Activo",
191.             "Monto ($)",
192.             dataset,
193.             org.jfree.chart.plot.PlotOrientation.VERTICAL,
194.             true,
195.             true,
196.             false
197.         );
198.         chart.getTitle().setMargin(20, 0, 20, 0);
199.
200.         CategoryPlot plot = chart.getCategoryPlot();
201.         BarRenderer renderer = (BarRenderer) plot.getRenderer();
202.
203.         Color[] colores = {
204.             new Color(91, 155, 213),
205.             new Color(112, 173, 71),
206.             new Color(255, 192, 0),
207.             new Color(237, 125, 49)
208.         };
209.
210.         for (int i = 0; i < dataset.getRowCount(); i++) {
211.             renderer.setSeriesPaint(i, colores[i % colores.length]);
212.         }
213.
214.         renderer.setMaximumBarWidth(0.1);
215.

```

```

216.         return new ChartPanel(chart);
217.     }
218.
219.     private double obtenerPrecioVentaDesdeBD(int idActivo) {
220.         double precioVenta = 0.0;
221.         String sql = "SELECT precio_venta FROM ventas WHERE id_activo = ?";
222.
223.         try (Connection conn = ConexionBD.getConnection();
224.              PreparedStatement pstmt = conn.prepareStatement(sql)) {
225.             pstmt.setInt(1, idActivo);
226.             try (ResultSet rs = pstmt.executeQuery()) {
227.                 if (rs.next()) {
228.                     precioVenta = rs.getDouble("precio_venta");
229.                 }
230.             }
231.         } catch (SQLException e) {
232.             e.printStackTrace();
233.         }
234.
235.         return precioVenta;
236.     }
237. }

```

Paquete: vista; Clase: FiltroFrame.java

```

1. package vista;
2.
3. import com.toedter.calendar.JDateChooser;
4. import util.ValidadorActivo;
5. import util.ValidadorFechas;
6. import util.VentanaUtils;
7.
8. import javax.swing.*;
9. import javax.swing.table.TableModel;
10. import java.awt.*;
11. import java.time.LocalDate;
12. import java.time.ZoneId;
13. import java.time.format.DateTimeFormatter;
14. import java.util.Collections;
15. import java.util.Date;
16. import java.util.HashMap;
17. import java.util.Map;
18.
19. public class FiltroFrame extends JFrame {
20.     private static final DateTimeFormatter FORMATO_FECHA =
21.     DateTimeFormatter.ofPattern("dd-MM-yyyy");
22.     private JTextField txtCostoMin, txtCostoMax;
23.     private JDateChooser fechaInicio, fechaFin;
24.     private JComboBox<String> comboCategoria, comboEstado;
25.     private JButton btnAplicar, btnReset;
26.
27.     private Map<String, Object> filtrosActuales;
28.
29.     public FiltroFrame(MainFrame mainFrame, Map<String, Object> filtrosGuardados) {
30.         setTitle("Filtrar Activos Fijos");
31.         setSize(800, 300);
32.         setLocationRelativeTo(null);
33.         VentanaUtils.deshabilitarMinimizar(this);
34.         setResizable(false);
35.         setLayout(new BorderLayout(10, 10));
36.
37.         filtrosActuales = filtrosGuardados != null ? new HashMap<>(filtrosGuardados)
: new HashMap<>();

```

```

38.     // Panel principal
39.     JPanel panelFiltros = new JPanel();
40.     panelFiltros.setLayout(new GridBagLayout());
41.     panelFiltros.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
42.     GridBagConstraints gbc = new GridBagConstraints();
43.     gbc.fill = GridBagConstraints.HORIZONTAL;
44.     gbc.insets = new Insets(10, 10, 10, 10);
45.     gbc.weightx = 1;
46.
47.     // Filtro: Costo Inicial
48.     gbc.gridx = 0; gbc.gridy = 0;
49.     panelFiltros.add(new JLabel("Costo Inicial (Mín):"), gbc);
50.     gbc.gridx = 1;
51.     txtCostoMin = new JTextField();
52.     panelFiltros.add(txtCostoMin, gbc);
53.
54.     gbc.gridx = 2;
55.     panelFiltros.add(new JLabel("Costo Inicial (Máx):"), gbc);
56.     gbc.gridx = 3;
57.     txtCostoMax = new JTextField();
58.     panelFiltros.add(txtCostoMax, gbc);
59.
60.     // Filtro: Fechas
61.     gbc.gridx = 0; gbc.gridy = 1;
62.     panelFiltros.add(new JLabel("Fecha Adquisición (Inicio):"), gbc);
63.     gbc.gridx = 1;
64.     fechaInicio = new JDateChooser();
65.     fechaInicio.setDateFormatString("dd-MM-yyyy");
66.     panelFiltros.add(fechaInicio, gbc);
67.
68.     gbc.gridx = 2;
69.     panelFiltros.add(new JLabel("Fecha Adquisición (Fin):"), gbc);
70.     gbc.gridx = 3;
71.     fechaFin = new JDateChooser();
72.     fechaFin.setDateFormatString("dd-MM-yyyy");
73.     panelFiltros.add(fechaFin, gbc);
74.
75.     // Filtro: Categoría
76.     gbc.gridx = 0; gbc.gridy = 2;
77.     panelFiltros.add(new JLabel("Categoría:"), gbc);
78.     gbc.gridx = 1;
79.     comboCategoria = new JComboBox<>(new String[]{"Todos", "Equipo de Computo",
    "Mobiliario", "Equipo de Laboratorio", "Vehículo", "Terreno"});
80.     panelFiltros.add(comboCategoria, gbc);
81.
82.     // Filtro: Estado
83.     gbc.gridx = 2;
84.     panelFiltros.add(new JLabel("Estado:"), gbc);
85.     gbc.gridx = 3;
86.     comboEstado = new JComboBox<>(new String[]{"Todos", "Alta", "Depreciado",
    "Baja", "Baja por venta"});
87.     panelFiltros.add(comboEstado, gbc);
88.
89.     // Botones
90.     JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));
91.     btnAplicar = new JButton("Aplicar");
92.     btnAplicar.addActionListener(e -> aplicarFiltros(mainFrame));
93.     btnReset = new JButton("Resetear");
94.     btnReset.addActionListener(e -> resetearFiltros(mainFrame));
95.
96.     panelBotones.add(btnAplicar);
97.     panelBotones.add(btnReset);
98.
99.     add(panelFiltros, BorderLayout.CENTER);
100.    add(panelBotones, BorderLayout.SOUTH);
101.

```

```

102.            cargarFiltros();
103.        }
104.
105.        private void aplicarFiltros(MainFrame mainFrame) {
106.            Map<String, Object> criterios = new HashMap<>();
107.
108.            try {
109.                RowSorter<? extends TableModel> sorter =
110.                    mainFrame.getTable().getRowSorter();
111.                int sortedColumnIndex = sorter.getSortKeys().isEmpty() ? -1 :
112.                    sorter.getSortKeys().get(0).getColumn();
113.                SortOrder sortOrder = sorter.getSortKeys().isEmpty() ?
114.                    SortOrder.UNSORTED : sorter.getSortKeys().get(0).getSortOrder();
115.
116.                if (!txtCostoMin.getText().isEmpty()) {
117.                    String errorCostoMin =
118.                        ValidadorActivo.validarCostoInicial(txtCostoMin.getText());
119.                    if (errorCostoMin != null) {
120.                        throw new IllegalArgumentException(errorCostoMin);
121.                    }
122.                    criterios.put("costoMin",
123.                        Double.parseDouble(txtCostoMin.getText()));
124.
125.                if (!txtCostoMax.getText().isEmpty()) {
126.                    String errorCostoMax =
127.                        ValidadorActivo.validarCostoInicial(txtCostoMax.getText());
128.                    if (errorCostoMax != null) {
129.                        throw new IllegalArgumentException(errorCostoMax);
130.                    }
131.                    criterios.put("costoMax",
132.                        Double.parseDouble(txtCostoMax.getText()));
133.
134.                    if (criterios.containsKey("costoMin") &&
135.                        criterios.containsKey("costoMax")) {
136.                        double costoMin = (double) criterios.get("costoMin");
137.                        double costoMax = (double) criterios.get("costoMax");
138.                        if (costoMin > costoMax) {
139.                            throw new IllegalArgumentException("El costo mínimo no
140.    puede ser mayor que el costo máximo.");
141.                        }
142.                    }
143.                    criterios.put("fechaInicio", inicio);
144.                }
145.                if (fechaFin.getDate() != null) {
146.                    LocalDate fin = convertirDateToLocalDate(fechaFin.getDate());
147.                    String fechaFinStr = convertirLocalDateToString(fin);
148.                    String errorFechaFin =
149.                        ValidadorFechas.validarFecha(fechaFinStr);
150.                    if (errorFechaFin != null) {
151.                        throw new IllegalArgumentException(errorFechaFin);
152.                    }
153.                    criterios.put("fechaFin", fin);
154.                }
155.

```

```

156.                if (criterios.containsKey("fechaInicio") &&
157.                    criterios.containsKey("fechaFin")) {
158.                        LocalDate inicio = (LocalDate) criterios.get("fechaInicio");
159.                        LocalDate fin = (LocalDate) criterios.get("fechaFin");
160.                        if (inicio.isAfter(fin)) {
161.                            throw new IllegalArgumentException("La fecha de inicio no
162.                                puede ser posterior a la fecha de fin.");
163.                        }
164.                if (!comboCategoria.getSelectedItem().toString().equals("Todos"))
165.                {
166.                    criterios.put("categoria",
167.                        comboCategoria.getSelectedItem().toString());
168.                }
169.                if (!comboEstado.getSelectedItem().toString().equals("Todos"))
170.                {
171.                    criterios.put("estado",
172.                        comboEstado.getSelectedItem().toString());
173.                }
174.                filtrosActuales.clear();
175.                filtrosActuales.putAll(criterios);
176.                mainFrame.actualizarFiltrosGuardados(filtrosActuales);
177.                mainFrame.filtrarActivosAvanzado(criterios);
178.                if (mainFrame.tablaEstaVacia()) {
179.                    JOptionPane.showMessageDialog(this,
180.                        "No se encontraron activos que coincidan con los
181.                        filtros seleccionados.",
182.                        "Sin Resultados",
183.                        JOptionPane.INFORMATION_MESSAGE);
184.                }
185.                mainFrame.configurarOrdenamiento();
186.                if (sortedColumnIndex != -1) {
187.                    sorter.setSortKeys(Collections.singletonList(new
188.                        RowSorter.SortKey(sortedColumnIndex, sortOrder)));
189.                }
190.
191.            } catch (IllegalArgumentException ex) {
192.                mostrarMensajeError(ex.getMessage());
193.            }
194.        }
195.
196.        private static LocalDate convertirDateALocalDate(Date date) {
197.            return date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
198.        }
199.
200.        private static String convertirLocalDateAString(LocalDate date) {
201.            return date.format(FORMATO_FECHA);
202.        }
203.
204.        private static Date convertirLocalDateADate(LocalDate localDate) {
205.            return
206.                Date.from(localDate.atStartOfDay(ZoneId.systemDefault()).toInstant());
207.        }
208.        public void resetearFiltros(MainFrame mainFrame) {
209.            txtCostoMin.setText("");
210.            txtCostoMax.setText("");
211.            fechaInicio.setDate(null);
212.            fechaFin.setDate(null);
213.            comboCategoria.setSelectedIndex(0);

```

```

214.         comboEstado.setSelectedIndex(0);
215.
216.         filtrosActuales.clear();
217.         mainFrame.actualizarFiltrosGuardados(new HashMap<>());
218.
219.         mainFrame.filtrarActivosAvanzado(new HashMap<>());
220.
221.         mainFrame.configurarOrdenamiento();
222.     }
223.
224.     private void cargarFiltros() {
225.         if (filtrosActuales.containsKey("costoMin")) {
226.             txtCostoMin.setText(String.valueOf(filtrosActuales.get("costoMin")));
227.         }
228.         if (filtrosActuales.containsKey("costoMax")) {
229.             txtCostoMax.setText(String.valueOf(filtrosActuales.get("costoMax")));
230.         }
231.         if (filtrosActuales.containsKey("fechaInicio")) {
232.             Object fechaInicioObj = filtrosActuales.get("fechaInicio");
233.             if (fechaInicioObj instanceof LocalDate) {
234.                 fechaInicio.setDate(convertirLocalDateADate((LocalDate)
235.                     fechaInicioObj));
236.             } else if (fechaInicioObj instanceof Date) {
237.                 fechaInicio.setDate((Date) fechaInicioObj);
238.             }
239.         }
240.         if (filtrosActuales.containsKey("fechaFin")) {
241.             Object fechaFinObj = filtrosActuales.get("fechaFin");
242.             if (fechaFinObj instanceof LocalDate) {
243.                 fechaFin.setDate(convertirLocalDateADate((LocalDate)
244.                     fechaFinObj));
245.             } else if (fechaFinObj instanceof Date) {
246.                 fechaFin.setDate((Date) fechaFinObj);
247.             }
248.         }
249.         if (filtrosActuales.containsKey("categoria")) {
250.             comboCategoria.setSelectedItem(filtrosActuales.get("categoria"));
251.         }
252.         if (filtrosActuales.containsKey("estado")) {
253.             comboEstado.setSelectedItem(filtrosActuales.get("estado"));
254.         }
255.     }
256.     private void mostrarMensajeError(String mensaje) {
257.         JOptionPane.showMessageDialog(this, mensaje, "Error",
258.             JOptionPane.ERROR_MESSAGE);
259.     }

```

Paquete: vista; Clase: InventarioFrame.java

```

1. package vista;
2.
3. import modelo.ActivoFijo;
4. import org.jfree.chart.ChartFactory;
5. import org.jfree.chart.ChartPanel;
6. import org.jfree.chart.JFreeChart;
7. import org.jfree.chart.labels.StandardPieSectionLabelGenerator;
8. import org.jfree.chart.plot.PiePlot;
9. import org.jfree.data.general.DefaultPieDataset;
10. import util.NonEditableTableModel;
11. import util.VentanaUtils;

```

```

12.
13. import javax.swing.*;
14. import java.awt.*;
15. import java.util.List;
16. import java.util.Map;
17. import java.util.stream.Collectors;
18. import java.text.NumberFormat;
19. import java.util.Locale;
20.
21. public class InventarioFrame extends JFrame {
22.     private List<ActivoFijo> inventario;
23.     private JTable tablaInventario;
24.     private JComboBox<String> filtroCombo;
25.     private JPanel contenedorGrafico;
26.     private JPanel panelResumen;
27.
28.     public InventarioFrame(List<ActivoFijo> inventarioOriginal) {
29.         this.inventario = inventarioOriginal.stream()
30.             .filter(activo -> !activo.getEstado().equalsIgnoreCase("Renovado"))
31.             .collect(Collectors.toList());
32.
33.         setTitle("Inventario de Activos");
34.         setSize(1400, 800);
35.         setResizable(false);
36.         setLocationRelativeTo(null);
37.         VentanaUtils.deshabilitarMinimizar(this);
38.         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
39.
40.         initComponents();
41.         setVisible(true);
42.     }
43.
44.     private void initComponents() {
45.         JPanel panelPrincipal = new JPanel(new GridBagLayout());
46.         GridBagConstraints gbc = new GridBagConstraints();
47.
48.         // Tabla de inventario
49.         NonEditableTableModel modeloTabla = crearModeloTabla();
50.         llenarModeloTabla(modeloTabla);
51.
52.         tablaInventario = new JTable(modeloTabla);
53.         tablaInventario.getTableHeader().setReorderingAllowed(false);
54.         JScrollPane scrollTabla = new JScrollPane(tablaInventario);
55.
56.         gbc.gridx = 0;
57.         gbc.gridy = 0;
58.         gbc.gridheight = 2;
59.         gbc.weightx = 0.7;
60.         gbc.weighty = 1.0;
61.         gbc.fill = GridBagConstraints.BOTH;
62.         panelPrincipal.add(scrollTabla, gbc);
63.
64.         // Panel de resumen
65.         panelResumen = new JPanel(new GridBagLayout());
66.         panelResumen.setBackground(Color.WHITE);
67.         GridBagConstraints gbcResumen = new GridBagConstraints();
68.         gbcResumen.insets = new Insets(10, 10, 10, 10);
69.         gbcResumen.fill = GridBagConstraints.HORIZONTAL;
70.         gbcResumen.gridx = 0;
71.         gbcResumen.weightx = 1;
72.
73.         agregarResumenPanel(gbcResumen);
74.
75.         // Filtro
76.         filtroCombo = new JComboBox<>(new String[]{"Categoría", "Estado", "Años"});
77.         filtroCombo.setPreferredSize(new Dimension(150, 30));

```

```

78.         filtroCombo.setAlignmentX(Component.CENTER_ALIGNMENT);
79.         gbcResumen.gridx = 3;
80.         gbcResumen.fill = GridBagConstraints.NONE;
81.         panelResumen.add(filtroCombo, gbcResumen);
82.
83.         // Gráfico
84.         contenedorGrafico = new JPanel(new BorderLayout());
85.         contenedorGrafico.setBackground(Color.WHITE);
86.         actualizarGrafico("Categoría");
87.
88.         gbcResumen.gridx = 4;
89.         gbcResumen.fill = GridBagConstraints.BOTH;
90.         gbcResumen.weighty = 1;
91.         panelResumen.add(contenedorGrafico, gbcResumen);
92.
93.         gbc.gridx = 1;
94.         gbc.gridy = 0;
95.         gbc.gridheight = 2;
96.         gbc.weightx = 0.3;
97.         gbc.fill = GridBagConstraints.BOTH;
98.         panelPrincipal.add(panelResumen, gbc);
99.
100.        filtroCombo.addActionListener(e -> actualizarGrafico((String)
filtroCombo.getSelectedItem()));
101.        add(panelPrincipal);
102.    }
103.
104.
105.    private NonEditableTableModel crearModeloTabla() {
106.        NonEditableTableModel modelo = new NonEditableTableModel();
107.        modelo.addColumn("Código Patrimonial");
108.        modelo.addColumn("Nombre");
109.        modelo.addColumn("Categoría");
110.        modelo.addColumn("Estado");
111.        modelo.addColumn("Fecha de Adquisición");
112.        modelo.addColumn("Costo Inicial");
113.        modelo.addColumn("Depreciación Mensual");
114.        modelo.addColumn("Depreciación Acumulada");
115.        modelo.addColumn("Valor Residual");
116.        return modelo;
117.    }
118.
119.    private void llenarModeloTabla(NonEditableTableModel modeloTabla) {
120.        NumberFormat numberFormat = NumberFormat.getNumberInstance(Locale.US);
121.
122.        for (ActivoFijo activo : inventario) {
123.            modeloTabla.addRow(new Object[]{
124.                activo.getCodigoPatrimonial(),
125.                activo.getNombre(),
126.                activo.getCategoría(),
127.                activo.getEstado(),
128.                activo.getFechaAdquisicion(),
129.                numberFormat.format(activo.getCostoInicial()),
130.                numberFormat.format(activo.calcularDepreciacionMensual()),
131.                numberFormat.format(activo.calcularDepreciacionAcumulada())
132.            },
133.            numberFormat.format(activo.calcularValorResidual())
134.        });
135.    }
136.
137.    private void agregarResumenPanel(GridBagConstraints gbcResumen) {
138.        int totalActivos = inventario.size();
139.
140.        NumberFormat numberFormat = NumberFormat.getNumberInstance(Locale.US);

```

```

141.             String costoTotalFormatted =
    numberFormat.format(inventario.stream().map.ToDouble(ActivoFijo::getCostoInicial).sum(
    ));
142.             String valorLibrosTotalFormatted =
    numberFormat.format(inventario.stream().map.ToDouble(ActivoFijo::calcularValorResidual
    ).sum()));
143.
144.             JLabel lblTotalActivos = new JLabel(
145.                 String.format("<html><div style='text-align: center; font-
    size: 16px;'>Total de Activos</div><div style='text-align: center; font-size: 24px;
    font-weight: bold;'>%d</div></html>", totalActivos),
146.                     SwingConstants.CENTER
147.                 );
148.
149.             JLabel lblCostoTotal = new JLabel(
150.                 String.format("<html><div style='text-align: center; font-
    size: 16px;'>Costo Inicial Total</div><div style='text-align: center; font-size:
    24px; font-weight: bold;'>$/%s</div></html>", costoTotalFormatted),
151.                     SwingConstants.CENTER
152.                 );
153.
154.             JLabel lblValorLibrosTotal = new JLabel(
155.                 String.format("<html><div style='text-align: center; font-
    size: 16px;'>Valor en Libros Total</div><div style='text-align: center; font-size:
    24px; font-weight: bold;'>$/%s</div></html>", valorLibrosTotalFormatted),
156.                     SwingConstants.CENTER
157.                 );
158.
159.             gbcResumen.gridx = 0;
160.             panelResumen.add(lblTotalActivos, gbcResumen);
161.             gbcResumen.gridx = 1;
162.             panelResumen.add(lblCostoTotal, gbcResumen);
163.             gbcResumen.gridx = 2;
164.             panelResumen.add(lblValorLibrosTotal, gbcResumen);
165.         }
166.
167.         private void actualizarGrafico(String criterio) {
168.             contenedorGrafico.removeAll();
169.             ChartPanel nuevoGrafico = crearGraficoTorta(criterio);
170.             contenedorGrafico.add(nuevoGrafico, BorderLayout.CENTER);
171.             contenedorGrafico.revalidate();
172.             contenedorGrafico.repaint();
173.         }
174.
175.         private ChartPanel crearGraficoTorta(String criterio) {
176.             DefaultPieDataset dataset = new DefaultPieDataset();
177.             Map<String, Long> agrupamiento = inventario.stream()
178.                 .collect(Collectors.groupingBy(
179.                     criterio.equals("Categoría") ?
    ActivoFijo::getCategoría :
    criterio.equals("Estado") ?
    ActivoFijo::getEstado :
    activo -
    activo.getFechaAdquisición().split("-")[2],
    Collectors.counting()
    ));
180.
181.             for (Map.Entry<String, Long> entry : agrupamiento.entrySet()) {
182.                 dataset.setValue(entry.getKey(), entry.getValue());
183.             }
184.
185.             JFreeChart chart = ChartFactory.createPieChart(
186.                 "Distribución de Activos por " + criterio,
187.                 dataset,
188.                 false,
189.                 true,

```

```

194.           false
195.       );
196.
197.       PiePlot plot = (PiePlot) chart.getPlot();
198.       plot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}
199.   ({1})"));
200.       plot.setBackgroundPaint(Color.WHITE);
201.       plot.setOutlineVisible(false);
202.
203.       Color[] colores = {
204.           new Color(169, 209, 142),
205.           new Color(255, 192, 0),
206.           new Color(112, 173, 71),
207.           new Color(237, 125, 49),
208.           new Color(91, 155, 213),
209.           new Color(193, 193, 193)
210.       };
211.
212.       int index = 0;
213.       for (Object key : dataset.getKeys()) {
214.           if (index < colores.length) {
215.               plot.setSectionPaint(key.toString(), colores[index]);
216.           } else {
217.               plot.setSectionPaint(key.toString(), Color.LIGHT_GRAY);
218.           }
219.           index++;
220.       }
221.
222.       return new ChartPanel(chart);
223.   }
224. }

```

Paquete: vista; Clase: LoginFrame.java

```

1. package vista;
2.
3. import controlador.Main;
4. import controlador.Autenticacion;
5. import modelo.Usuario;
6.
7. import javax.swing.*;
8. import java.awt.*;
9. import java.awt.event.FocusAdapter;
10. import java.awt.event.FocusEvent;
11. import java.awt.event.ActionEvent;
12. import java.awt.event.ActionListener;
13.
14. public class LoginFrame extends JFrame {
15.
16.     private Autenticacion autenticacion = new Autenticacion();
17.     private Usuario usuarioAutenticado;
18.     private int intentosRestantes = 3;
19.
20.     private JTextField TextUsuario;
21.     private JPasswordField TxtPass;
22.     private JButton jButtonIngresar;
23.     private JLabel jLabel1;
24.     private JLabel jLabel2;
25.     private JLabel logo;
26.
27.     public LoginFrame() {
28.         initComponents();
29.     }
30. }

```

```

31.     private void initComponents() {
32.         // Panel principal
33.         JPanel jPanel1 = new JPanel();
34.         jPanel1.setBackground(new Color(37, 51, 60));
35.         jPanel1.setLayout(null);
36.
37.         // Logo
38.         logo = new JLabel();
39.         logo.setIcon(redimensionarLogo("Valorium.png", 85, 73));
40.         logo.setBounds(120, 20, 120, 120);
41.         jPanel1.add(logo);
42.
43.         // Etiqueta de "Usuario"
44.         jLabel2 = new JLabel("Usuario");
45.         jLabel2.setFont(new Font("Segoe UI", Font.BOLD, 18));
46.         jLabel2.setForeground(new Color(255, 255, 255));
47.         jLabel2.setBounds(40, 150, 140, 30);
48.         jPanel1.add(jLabel2);
49.
50.         // Campo de texto para ingresar el usuario
51.         TextUsuario = new JTextField();
52.         TextUsuario.setBounds(40, 190, 250, 40);
53.         TextUsuario.setBackground(new Color(43, 47, 51));
54.         TextUsuario.setForeground(Color.WHITE);
55.         TextUsuario.setBorder(BorderFactory.createLineBorder(new Color(103, 113,
121), 2));
56.         TextUsuario.setCaretColor(Color.WHITE);
57.         TextUsuario.setOpaque(true);
58.         agregarEfectoFoco(TextUsuario);
59.         jPanel1.add(TextUsuario);
60.
61.         // Etiqueta de "Contraseña"
62.         jLabel1 = new JLabel("Contraseña");
63.         jLabel1.setFont(new Font("Segoe UI", Font.BOLD, 18));
64.         jLabel1.setForeground(new Color(255, 255, 255));
65.         jLabel1.setBounds(40, 250, 140, 30);
66.         jPanel1.add(jLabel1);
67.
68.         // Campo de texto para la contraseña
69.         TxtPass = new JPasswordField();
70.         TxtPass.setBounds(40, 290, 250, 40);
71.         TxtPass.setBackground(new Color(43, 47, 51));
72.         TxtPass.setForeground(Color.WHITE);
73.         TxtPass.setBorder(BorderFactory.createLineBorder(new Color(103, 113, 121),
2));
74.         TxtPass.setCaretColor(Color.WHITE);
75.         TxtPass.setOpaque(true);
76.         agregarEfectoFoco(TxtPass);
77.         jPanel1.add(TxtPass);
78.
79.         // Botón de "Iniciar sesión"
80.         jButtonIngresar = new JButton("Iniciar sesión");
81.         jButtonIngresar.setFont(new Font("Segoe UI", Font.BOLD, 18));
82.         jButtonIngresar.setBackground(new Color(108, 215, 159));
83.         jButtonIngresar.setForeground(Color.WHITE); // Texto blanco
84.         jButtonIngresar.setBounds(85, 360, 160, 40);
85.         jButtonIngresar.setFocusPainted(false);
86.         jButtonIngresar.setBorder(BorderFactory.createLineBorder(new Color(108, 215,
159), 2));
87.         jButtonIngresar.setBorder(BorderFactory.createEmptyBorder());
88.         agregarEfectoHover(jButtonIngresar);
89.         jPanel1.add(jButtonIngresar);
90.
91.         jButtonIngresar.addActionListener(new ActionListener() {
92.             public void actionPerformed(ActionEvent evt) {
93.                 jButtonIngresarActionPerformed(evt);

```

```

94.         }
95.     });
96.
97.     TextUsuario.addActionListener(new ActionListener() {
98.         public void actionPerformed(ActionEvent evt) {
99.             jButtonIngresar.doClick();
100.        }
101.    });
102.
103.    TxtPass.addActionListener(new ActionListener() {
104.        public void actionPerformed(ActionEvent evt) {
105.            jButtonIngresar.doClick();
106.        }
107.    });
108.
109.    this.add(jPanel1);
110.    this.setTitle("Valorium - Iniciar sesión");
111.    this.setSize(340, 500);
112.    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
113.    this.setLocationRelativeTo(null);
114.    this.setResizable(false);
115. }
116.
117. private void jButtonIngresarActionPerformed(ActionEvent evt) {
118.     String nombreUsuario = TextUsuario.getText();
119.     String contrasena = new String(TxtPass.getPassword());
120.
121.     if (nombreUsuario.isEmpty() || contrasena.isEmpty()) {
122.         JOptionPane.showMessageDialog(this, "Debe completar todos los
campos.", "Advertencia", JOptionPane.WARNING_MESSAGE);
123.         return;
124.     }
125.
126.     usuarioAutenticado = autenticacion.autenticarUsuario(nombreUsuario,
contrasena);
127.
128.     if (usuarioAutenticado != null) {
129.         Main.usuarioAutenticado = usuarioAutenticado;
130.         this.dispose();
131.         Main.iniciarMainFrame();
132.     } else {
133.         TxtPass.setText("");
134.         intentosRestantes--;
135.         if (intentosRestantes > 0) {
136.             JOptionPane.showMessageDialog(this, "Usuario o contraseña
incorrectos. Intentos restantes: " + intentosRestantes, "Error",
JOptionPane.ERROR_MESSAGE);
137.         } else {
138.             JOptionPane.showMessageDialog(this, "Se ha excedido el número
de intentos. El programa se cerrará.", "Error", JOptionPane.ERROR_MESSAGE);
139.             System.exit(0);
140.         }
141.     }
142. }
143.
144.     private ImageIcon redimensionarLogo(String ruta, int width, int height) {
145.         ImageIcon logoIcon = new
ImageIcon(getClass().getClassLoader().getResource(ruta));
146.         Image logoImage = logoIcon.getImage().getScaledInstance(width, height,
Image.SCALE_SMOOTH);
147.         return new ImageIcon(logoImage);
148.     }
149.
150.     private void agregarEfectoFoco(JTextField campo) {
151.         campo.setBorder(BorderFactory.createLineBorder(new Color(103, 113,
121), 2));

```

```

152.             campo.addFocusListener(new FocusAdapter() {
153.                 @Override
154.                 public void focusGained(FocusEvent e) {
155.                     campo.setBorder(BorderFactory.createLineBorder(new Color(108,
156.                         215, 159), 2));
157.                 }
158.
159.                 @Override
160.                 public void focusLost(FocusEvent e) {
161.                     campo.setBorder(BorderFactory.createLineBorder(new Color(103,
162.                         113, 121), 2));
163.                 }
164.             }
165.
166.             private void agregarEfectoHover(JButton boton) {
167.                 boton.addMouseListener(new java.awt.event.MouseAdapter() {
168.                     public void mouseEntered(java.awt.event.MouseEvent evt) {
169.                         boton.setBackground(new Color(87, 171, 127));
170.                     }
171.
172.                     public void mouseExited(java.awt.event.MouseEvent evt) {
173.                         boton.setBackground(new Color(108, 215, 159));
174.                     }
175.                 });
176.             }
177.         }

```

Paquete: vista; Clase: MainFrame.java

```

1. package vista;
2.
3. import controlador.ActivoControlador;
4. import controlador.Main;
5. import modelo.*;
6. import org.apache.poi.ss.usermodel.Font;
7. import servicio.ActivoServicio;
8. import util.ConexionBD;
9. import util.ValidadorActivo;
10. import util.NonEditableTableModel;
11.
12. import org.apache.poi.ss.usermodel.*;
13. import org.apache.poi.xssf.usermodel.XSSFWorkbook;
14.
15. import java.awt.Color;
16. import java.awt.event.MouseAdapter;
17. import java.awt.event.MouseEvent;
18. import java.io.File;
19. import java.io.FileOutputStream;
20. import javax.swing.JFileChooser;
21. import com.toedter.calendar.JDateChooser;
22.
23. import java.io.IOException;
24. import java.sql.Connection;
25. import java.sql.PreparedStatement;
26. import java.sql.ResultSet;
27. import java.sql.SQLException;
28. import java.time.format.TextStyle;
29. import java.util.*;
30. import java.text.SimpleDateFormat;
31. import javax.swing.*;
32. import javax.swing.event.DocumentEvent;
33. import javax.swing.event.DocumentListener;

```

```

34. import java.awt.*;
35. import java.time.LocalDate;
36. import java.time.format.DateTimeFormatter;
37. import java.util.List;
38. import java.util.stream.Collectors;
39. import javax.swing.table.*;
40.
41. import static modelo.ActivoFijo.determinarVidaUtil;
42.
43. public class MainFrame extends JFrame {
44.     private NonEditableTableModel model;
45.     private JTable table;
46.     public JTextField txtBuscar;
47.     private JTextField txtNombre;
48.     private JComboBox<String> comboCategoria;
49.     private JDateChooser dateChooserFechaAdquisicion;
50.     private JTextField txtVidaUtil;
51.     private JTextField txtCostoInicial;
52.     private JTextField txtDepreciacionMensual;
53.     private JTextField txtDepreciacionAcumulada;
54.     private JTextField txtValorNeto;
55.     private JTextArea txtAreaMensajes;
56.     private JPanel panelTotales;
57.     private JButton btnModificar;
58.     private JButton btnRegistrar;
59.     private JButton btnDarBaja;
60.     private JButton btnVender;
61.     private JComboBox<String> comboTipoBaja;
62.
63.     private JButton btnCambios;
64.     private JButton btnProyectarDepreciacion;
65.     private JButton btnRevaluar;
66.     private JButton btnBuscar;
67.     private JButton btnFiltrar;
68.     private JButton btnExportar;
69.     private JButton btnCerrarSesion;
70.     private JButton btnInventario;
71.     boolean enModoEdicion = false;
72.     private List<ActivoFijo> listaBaseDatos;
73.     private ActivoControlador activoControlador;
74.     private int indiceActivoEnEdicion = -1;
75.     private double totalCostoInicial = 0.0;
76.     private double totalDepreciacionMensual = 0.0;
77.     private double totalDepreciacionAcumulada = 0.0;
78.     private double totalValorResidual = 0.0;
79.     private int numeroDeActivos = 0;
80.     private ActivoServicio activoServicio;
81.     private String nombreOriginal;
82.     private String categoriaOriginal;
83.     private NotificacionFrame notificacionFrame;
84.     private int columnaActualOrdenada = -1;
85.     private SortOrder ordenActual = SortOrder.UNSORTED;
86.     private Map<String, Object> filtrosGuardados = new HashMap<>();
87.     private FiltroFrame filtroFrame;
88.     private JButton btnRenovar;
89.     private JButton btnUbicacionActivos;
90.     private JButton btnRecargar;
91.     private JButton btnResetearBusqueda;
92.     private JButton btnAnalisisDeBajas;
93.     private List<ActivoFijo> listaFiltrada;
94.     private JComboBox<Sede> comboSede;
95.     public List<ActivoFijo> getListaBaseDatos() {
96.         return listaBaseDatos;
97.     }
98.
99.     public MainFrame() {

```

```

100.         activoServicio = new ActivoServicio(this);
101.         activoControlador = new ActivoControlador(activoServicio);
102.         listaBaseDatos = activoServicio.cargarActivosDesdeBD();
103.         notificacionFrame = new NotificacionFrame(this);
104.
105.         setTitle("Valorium");
106.         setExtendedState(JFrame.MAXIMIZED_BOTH);
107.         setResizable(false);
108.         setDefaultCloseOperation(EXIT_ON_CLOSE);
109.         setLayout(new BorderLayout(10, 10));
110.
111.         configurarUI();
112.         cargarSedes();
113.         cargarActivosEnTabla();
114.         configurarOrdenamiento();
115.         setVisible(true);
116.         mostrarNotificacionesPendientes();
117.     }
118.
119.     private void configurarUI() {
120.         // Configuración de la tabla
121.         model = new NonEditableTableModel();
122.
123.         model.addColumn("Nº");
124.         model.addColumn("Código Patrimonial");
125.         model.addColumn("Nombre");
126.         model.addColumn("Categoría");
127.         model.addColumn("Estado");
128.         model.addColumn("Fecha de Adquisición");
129.         model.addColumn("Vida Útil");
130.         model.addColumn("Costo Inicial");
131.         model.addColumn("Depreciación Mensual");
132.         model.addColumn("Depreciación Acumulada");
133.         model.addColumn("Valor Residual");
134.
135.         table = new JTable(model);
136.         table.getTableHeader().setReorderingAllowed(false);
137.         JScrollPane scrollPane = new JScrollPane(table);
138.
139.         // Panel para el lateral derecho (lista de activos)
140.         JPanel panelDerecho = new JPanel();
141.         panelDerecho.setLayout(new BorderLayout());
142.
143.         // Panel para el título
144.         JPanel panelTitulo = new JPanel(new FlowLayout(FlowLayout.CENTER));
145.         JLabel lblTitulo = new JLabel("Lista de Activos Registrados");
146.         lblTitulo.setFont(new java.awt.Font("Arial", java.awt.Font.BOLD, 20));
147.         panelTitulo.add(Box.createVerticalStrut(50));
148.         panelTitulo.add(lblTitulo);
149.
150.         JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.RIGHT));
151.         btnModificar = new JButton("Modificar");
152.         btnModificar.addActionListener(e -> modificarActivo());
153.         panelBotones.add(btnModificar);
154.         panelBotones.add(Box.createHorizontalStrut(10));
155.         btnDarBaja = new JButton("Dar de Baja");
156.         btnDarBaja.addActionListener(e -> darDeBajaActivo());
157.         panelBotones.add(btnDarBaja);
158.         panelBotones.add(Box.createHorizontalStrut(0));
159.         comboTipoBaja = new JComboBox<>(new String[]{
160.             "Obsolescencia", "Siniestro", "Fin de vida útil"
161.         });
162.         panelBotones.add(comboTipoBaja);
163.         panelBotones.add(Box.createHorizontalStrut(10));
164.         btnVender = new JButton("Vender");
165.         btnVender.addActionListener(e -> venderActivo());

```

```

166.         panelBotones.add(btnVender);
167.
168.         // Panel para los botones de búsqueda
169.         JPanel panelBuscar = new JPanel(new FlowLayout(FlowLayout.LEFT));
170.         txtBuscar = new JTextField(15);
171.         panelBuscar.add(txtBuscar);
172.         txtBuscar.getDocument().addDocumentListener(new DocumentListener() {
173.             @Override
174.             public void insertUpdate(DocumentEvent e) {
175.                 filtrarActivos(txtBuscar.getText());
176.             }
177.
178.             @Override
179.             public void removeUpdate(DocumentEvent e) {
180.                 filtrarActivos(txtBuscar.getText());
181.             }
182.
183.             @Override
184.             public void changedUpdate(DocumentEvent e) {
185.                 filtrarActivos(txtBuscar.getText());
186.             }
187.         });
188.         panelBuscar.add(Box.createHorizontalStrut(0));
189.         btnResetearBusqueda = new JButton("X");
190.         btnResetearBusqueda.addActionListener(e -> {
191.             txtBuscar.setText("");
192.         });
193.         panelBuscar.add(btnResetearBusqueda);
194.         panelBuscar.add(Box.createHorizontalStrut(10));
195.         btnBuscar = new JButton("Buscar");
196.         btnBuscar.addActionListener(e -> {
197.             String textoBusqueda = txtBuscar.getText();
198.             if (!textoBusqueda.isEmpty()) {
199.                 seleccionarPrimeraCoincidencia(textoBusqueda);
200.             } else {
201.                 mostrarMensaje("Por favor, ingrese un texto para buscar.");
202.             }
203.         });
204.         panelBuscar.add(btnBuscar);
205.         panelBuscar.add(Box.createHorizontalStrut(10));
206.         btnFiltrar = new JButton("Filtrar");
207.         panelBuscar.add(btnFiltrar);
208.         btnFiltrar.addActionListener(e -> abrirFiltroFrame());
209.         panelBuscar.add(Box.createHorizontalStrut(10));
210.         btnExportar = new JButton("Exportar reporte");
211.         btnExportar.addActionListener(e -> exportarReporte());
212.         panelBuscar.add(btnExportar);
213.         panelBuscar.add(Box.createHorizontalStrut(10));
214.         btnRecargar = new JButton("O");
215.         btnRecargar.addActionListener(e -> recargarDatos());
216.         panelBuscar.add(btnRecargar);
217.         panelBuscar.add(Box.createHorizontalStrut(10));
218.
219.         // Panel que contiene los botones de búsqueda como los de acción
220.         JPanel panelAcciones = new JPanel(new BorderLayout());
221.         panelAcciones.add(panelBuscar, BorderLayout.WEST);
222.         panelAcciones.add(panelBotones, BorderLayout.EAST);
223.
224.         // Panel que contiene tanto el título como las acciones
225.         JPanel panelTituloBotones = new JPanel();
226.         panelTituloBotones.setLayout(new BoxLayout(panelTituloBotones,
227.             BoxLayout.Y_AXIS));
228.         panelTituloBotones.add(panelTitulo);
229.         panelTituloBotones.add(panelAcciones);
230.         panelTituloBotones.add(Box.createVerticalStrut(10));

```

```

231.         panelDerecho.add(panelTituloBotones, BorderLayout.NORTH);
232.         panelDerecho.add(scrollPane, BorderLayout.CENTER);
233.
234.         // Panel para los totales y el botón
235.         JPanel panelTotalesConBoton = new JPanel();
236.         panelTotalesConBoton.setLayout(new BorderLayout());
237.         panelTotales = new JPanel();
238.         panelTotales.add(new JLabel("Número de Activos: " + numeroDeActivos +
239.             " | "));
240.         panelTotales.add(new JLabel("Total de Costo Inicial: S/" +
241.             formatearNumero(totalCostoInicial) + " | "));
242.         panelTotales.add(new JLabel("Total de Depreciación Mensual: S/" +
243.             formatearNumero(totalDepreciacionMensual) + " | "));
244.         panelTotales.add(new JLabel("Total de Depreciación Acumulada: S/" +
245.             formatearNumero(totalDepreciacionAcumulada) + " | "));
246.         panelTotales.add(new JLabel("Total de Valor Residual: S/" +
247.             formatearNumero(totalValorResidual)));
248.
249.         // Panel de logo y registro
250.         JPanel panelRegistro = new JPanel();
251.         panelRegistro.setLayout(new BoxLayout(panelRegistro,
252.             BoxLayout.Y_AXIS));
253.         JLabel lblLogo = new JLabel();
254.         ImageIcon logoIcon = new
255.             ImageIcon(getClass().getClassLoader().getResource("Valorium.png"));
256.         Image scaledImage = logoIcon.getImage().getScaledInstance(171, 145,
257.             Image.SCALE_SMOOTH);
258.         lblLogo.setIcon(new ImageIcon(scaledImage));
259.         lblLogo.setAlignmentX(Component.CENTER_ALIGNMENT);
260.         lblLogo.setBorder(BorderFactory.createEmptyBorder(50, 0, 0, 0));
261.         panelRegistro.add(lblLogo);
262.
263.         // Panel de datos
264.         JPanel panelDatos = new JPanel(new GridBagLayout());
265.         GridBagConstraints gbc = new GridBagConstraints();
266.         gbc.fill = GridBagConstraints.HORIZONTAL;
267.         gbc.insets = new Insets(5, 5, 5, 5);
268.         Dimension textFieldSize = new Dimension(200, 25);
269.
270.         txtNombre = new JTextField();
271.         txtNombre.setPreferredSize(textFieldSize);
272.         txtNombre.addActionListener(e -> activarBoton());
273.         comboCategoria = new JComboBox<>(new String[]{"Equipo de Computo",
274.             "Mobiliario", "Equipo de Laboratorio", "Vehiculo", "Terreno"});
275.         comboCategoria.setPreferredSize(textFieldSize);
276.         dateChooserFechaAdquisicion = new JDateChooser();
277.         dateChooserFechaAdquisicion.setDateFormatString("dd-MM-yyyy");
278.         dateChooserFechaAdquisicion.setPreferredSize(textFieldSize);
279.         dateChooserFechaAdquisicion.getDateEditor().getUiComponent().addKeyList
280.             tener(new java.awt.event.KeyAdapter() {
281.                 public void keyPressed(java.awt.event.KeyEvent evt) {
282.                     if (evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER) {
283.                         activarBoton();
284.                     }
285.                 }
286.             });
287.             txtVidaUtil = new JTextField();
288.             txtVidaUtil.setPreferredSize(textFieldSize);
289.             txtCostoInicial = new JTextField();
290.             txtCostoInicial.setPreferredSize(textFieldSize);
291.             txtCostoInicial.addActionListener(e -> activarBoton());
292.             comboSede = new JComboBox<>();

```



```

347.                 activoServicio.contarActivosPorEstado("Baja por venta");
348.                 if (activosRenovables > 0) {
349.                     RenovacionFrame renovacionFrame = new RenovacionFrame(new
350.                         ActivoServicio(), this);
351.                     renovacionFrame.setVisible(true);
352.                 } else {
353.                     mostrarMensaje("No hay activos disponibles para renovar.");
354.                 }
355.             });
356.             btnAnalisisDeBajas = new JButton("Análisis de Bajas");
357.             btnAnalisisDeBajas.addActionListener(e -> {
358.                 int activosBaja = activoServicio.contarActivosPorEstado("Baja") +
359.                     activoServicio.contarActivosPorEstado("Baja por venta");
360.
361.                 if (activosBaja > 0) {
362.                     List<ActivoFijo> activos =
363.                         activoControlador.obtenerInventario();
364.                     new AnalisisDeBajasFrame(activos).setVisible(true);
365.                 } else {
366.                     mostrarMensaje("No hay activos en baja para analizar.");
367.                 }
368.             });
369.             panelBotonesAccion.add(btnAnalisisDeBajas);
370.             btnUbicacionActivos = new JButton("Ubicación de Activos");
371.             btnUbicacionActivos.addActionListener(e -> {
372.                 List<Sede> sedes = activoServicio.cargarSedesDesdeBD();
373.                 List<ActivoFijo> activos = activoServicio.cargarActivosDesdeBD();
374.                 if (!sedes.isEmpty() && !activos.isEmpty()) {
375.                     new UbicacionActivosFrame(activoServicio);
376.                 } else if (sedes.isEmpty()) {
377.                     mostrarMensaje("No hay sedes registradas en el sistema.");
378.                 } else if (activos.isEmpty()) {
379.                     mostrarMensaje("No hay activos registrados en el sistema.");
380.                 }
381.             });
382.             Color rojoPastel = new Color(255, 153, 153);
383.             panelBotonesAccion.add(btnUbicacionActivos);
384.             btnCerrarSesion = new JButton("Cerrar sesión");
385.             btnCerrarSesion.setBackground(rojoPastel);
386.             btnCerrarSesion.addActionListener(e -> cerrarSesion());
387.             panelBotonesAccion.add(btnCerrarSesion);
388.
389.             panelRegistro.add(panelBotonesAccion);
390.             panelRegistro.add(Box.createVerticalStrut(20));
391.
392.             // Panel para mostrar mensajes
393.             JPanel panelMensajes = new JPanel();
394.             panelMensajes.setLayout(new BorderLayout());
395.
396.             // JTextArea para los mensajes
397.             txtAreaMensajes = new JTextArea(1, 30);
398.             txtAreaMensajes.setEditable(false);
399.             txtAreaMensajes.setLineWrap(true);
400.             txtAreaMensajes.setWrapStyleWord(true);
401.             txtAreaMensajes.setBackground(new java.awt.Color(245, 245, 245));
402.
403.             // JScrollPane para permitir el desplazamiento
404.             JScrollPane scrollPaneMensajes = new JScrollPane(txtAreaMensajes);
405.             scrollPaneMensajes.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
406.
407.             // Título de mensajes
408.             JPanel panelTituloMensajes = new JPanel(new

```

```

409.         panelTituloMensajes.add(lblTituloMensajes);
410.
411.         panelMensajes.add(panelTituloMensajes, BorderLayout.NORTH);
412.         panelMensajes.add(scrollPaneMensajes, BorderLayout.CENTER);
413.         panelMensajes.setAlignmentX(Component.CENTER_ALIGNMENT);
414.
415.         panelRegistro.add(panelMensajes);
416.
417.         // Usuario, fecha y versión
418.         JPanel panelInfo = new JPanel(new FlowLayout(FlowLayout.CENTER));
419.         Usuario usuario = Main.usuarioAutenticado;
420.         JLabel lblUsuario = new JLabel("Usuario: " +
421.             usuario.getNombreUsuario());
422.         panelInfo.add(lblUsuario);
423.         panelInfo.add(Box.createHorizontalGlue());
424.
425.         LocalDate fechaActual = LocalDate.now();
426.         DateTimeFormatter formatter =
427.             DateTimeFormatter.ofPattern("dd/MM/yyyy");
428.         String fechaFormatada = fechaActual.format(formatter);
429.         JLabel lblFecha = new JLabel("Fecha: " + fechaFormatada);
430.         panelInfo.add(lblFecha);
431.         panelInfo.add(Box.createHorizontalGlue());
432.         panelInfo.add(lblVersion);
433.
434.         panelRegistro.add(panelInfo);
435.         add(panelRegistro, BorderLayout.WEST);
436.         add(panelDerecho, BorderLayout.CENTER);
437.     }
438.
439.     private void cargarSedes() {
440.         comboSede.removeAllItems();
441.         List<Sede> sedes = activoServicio.cargarSedesDesdeBD();
442.         for (Sede sede : sedes) {
443.             comboSede.addItem(sede);
444.         }
445.     }
446.
447.     public void recargarDatos() {
448.         txtBuscar.setText("");
449.         getFiltroFrame().resetearFiltros(this);
450.
451.         listaBaseDatos = activoServicio.cargarActivosDesdeBD()
452.             .stream()
453.             .filter(activo ->
454.                 !activo.getEstado().equalsIgnoreCase("Renovado"))
455.             .collect(Collectors.toList());
456.
457.         NonEditableTableModel nuevoModel = new NonEditableTableModel();
458.         nuevoModel.setColumnIdentifiers(new String[]{
459.             "Nº", "Código Patrimonial", "Nombre", "Categoría", "Estado",
460.             "Fecha de Adquisición", "Vida Útil", "Costo Inicial",
461.             "Depreciación Mensual", "Depreciación Acumulada", "Valor
462.             Residual"
463.         });
464.
465.         for (ActivoFijo activo : listaBaseDatos) {
466.             nuevoModel.addRow(new Object[]{
467.                 activo.getNumero(),
468.                 activo.getCodigoPatrimonial(),
469.                 activo.getNombre(),
470.                 activo.getCategoría(),
471.                 activo.getEstado(),
472.                 activo.getFechaAdquisicion(),
473.                 activo.getCostoInicial(),
474.                 activo.getDepreciacionMensual(),
475.                 activo.getDepreciacionAcumulada(),
476.                 activo.getValorResidual()
477.             });
478.         }
479.     }

```

```

471.                 activo.getVidaUtil()),
472.                 String.format("%.2f", activo.getCostoInicial()),
473.                 String.format("%.2f",
474.                     activo.calcularDepreciacionMensual()),
475.                     String.format("%.2f",
476.                         activo.calcularDepreciacionAcumulada()),
477.                         String.format("%.2f", activo.calcularValorResidual())
478.                     });
479.                 }
480.             table.setModel(nuevoModel);
481.             model = nuevoModel;
482.             TableRowSorter<TableModel> sorter = new
483.             TableRowSorter<>(table.getModel());
484.             configurarComparadores(sorter);
485.             table.setRowSorter(sorter);
486.             table.revalidate();
487.             table.repaint();
488.             actualizarVistaDeTotales();
489.         }
490.     }
491.
492.     private void configurarComparadores(TableRowSorter<TableModel> sorter) {
493.         sorter.setComparator(0, Comparator.comparing(o -> {
494.             try {
495.                 return Integer.parseInt(o.toString());
496.             } catch (NumberFormatException e) {
497.                 return Integer.MAX_VALUE;
498.             }
499.         })); // N°
500.
501.         sorter.setComparator(1, Comparator.comparing(o -> {
502.             String[] partes = o.toString().split("-");
503.             try {
504.                 int anio = Integer.parseInt(partes[0]);
505.                 int consecutivo = partes.length > 1 ?
506.                     Integer.parseInt(partes[1]) : 0;
507.                 return anio * 10000 + consecutivo;
508.             } catch (NumberFormatException e) {
509.                 return Integer.MAX_VALUE;
510.             }
511.         })); // Código Patrimonial
512.         sorter.setComparator(2, Comparator.comparing(o ->
513.             o.toString().toLowerCase())); // Nombre
514.         sorter.setComparator(3, Comparator.comparing(o ->
515.             o.toString().toLowerCase())); // Categoría
516.         sorter.setComparator(4, Comparator.comparing(o ->
517.             o.toString().toLowerCase())); // Estado
518.         sorter.setComparator(5, Comparator.comparing(o -> {
519.             try {
520.                 DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-
521. MM-yyyy");
522.                 return LocalDate.parse(o.toString(), formatter);
523.             } catch (Exception e) {
524.                 return LocalDate.MIN;
525.             }
526.         })); // Fecha de Adquisición
527.         sorter.setComparator(6, Comparator.comparingInt(o -> {
528.             try {

```

```

529.             return Integer.parseInt(o.toString());
530.         } catch (NumberFormatException e) {
531.             return Integer.MAX_VALUE;
532.         }
533.     ); // Vida Útil
534.
535.     sorter.setComparator(7, Comparator.comparingDouble(o -> {
536.         try {
537.             return Double.parseDouble(o.toString().replace(",",
538.                 "").replace("S/", ""));
539.         } catch (NumberFormatException e) {
540.             return Double.MAX_VALUE;
541.         }
542.     )); // Costo Inicial
543.
544.     sorter.setComparator(8, Comparator.comparingDouble(o -> {
545.         try {
546.             return Double.parseDouble(o.toString().replace(",",
547.                 "").replace("S/", ""));
548.         } catch (NumberFormatException e) {
549.             return Double.MAX_VALUE;
550.         }
551.     )); // Depreciación Mensual
552.
553.     sorter.setComparator(9, Comparator.comparingDouble(o -> {
554.         try {
555.             return Double.parseDouble(o.toString().replace(",",
556.                 "").replace("S/", ""));
557.         } catch (NumberFormatException e) {
558.             return Double.MAX_VALUE;
559.         }
560.     )); // Depreciación Acumulada
561.
562.     sorter.setComparator(10, Comparator.comparingDouble(o -> {
563.         try {
564.             return Double.parseDouble(o.toString().replace(",",
565.                 "").replace("S/", ""));
566.         } catch (NumberFormatException e) {
567.             return Double.MAX_VALUE;
568.         }
569.     )); // Valor Residual
570.
571.     public void configurarOrdenamiento() {
572.         table.setAutoCreateRowSorter(true);
573.         TableRowSorter<TableModel> sorter = new
574.             TableRowSorter<>(table.getModel());
575.         sorter.setComparator(0, Comparator.comparingInt(o -> {
576.             try {
577.                 return Integer.parseInt(o.toString());
578.             } catch (NumberFormatException e) {
579.                 return Integer.MAX_VALUE;
580.             }
581.         )); // N°
582.         sorter.setComparator(1, Comparator.comparing(o -> {
583.             String[] partes = o.toString().split("-");
584.             try {
585.                 int anio = Integer.parseInt(partes[0]);
586.                 int consecutivo = partes.length > 1 ?
587.                     Integer.parseInt(partes[1]) : 0;
588.                 return anio * 10000 + consecutivo;
589.             } catch (NumberFormatException e) {
590.                 return Integer.MAX_VALUE;
591.             }
592.         }));
593.     }

```

```

589.        }
590.        )); // Código Patrimonial
591.
592.        sorter.setComparator(2, Comparator.comparing(o ->
593.            o.toString().toLowerCase())); // Nombre
594.        sorter.setComparator(3, Comparator.comparing(o ->
595.            o.toString().toLowerCase())); // Categoría
596.        sorter.setComparator(4, Comparator.comparing(o ->
597.            o.toString().toLowerCase())); // Estado
598.        sorter.setComparator(5, Comparator.comparing(o -> {
599.            try {
600.                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-
601. MM-yyyy");
602.                return LocalDate.parse(o.toString(), formatter);
603.            } catch (Exception e) {
604.                return LocalDate.MIN;
605.            }
606.        )); // Fecha de Adquisición
607.        sorter.setComparator(6, Comparator.comparingInt(o -> {
608.            try {
609.                return Integer.parseInt(o.toString());
610.            } catch (NumberFormatException e) {
611.                return Integer.MAX_VALUE;
612.            }
613.        )); // Vida Útil
614.
615.        sorter.setComparator(7, Comparator.comparingDouble(o -> {
616.            try {
617.                return Double.parseDouble(o.toString().replace(",",
618.                    "").replace("S/", ""));
619.            } catch (NumberFormatException e) {
620.                return Double.MAX_VALUE;
621.            }
622.        )); // Costo Inicial
623.
624.        sorter.setComparator(8, Comparator.comparingDouble(o -> {
625.            try {
626.                return Double.parseDouble(o.toString().replace(",",
627.                    "").replace("S/", ""));
628.            } catch (NumberFormatException e) {
629.                return Double.MAX_VALUE;
630.            }
631.        )); // Depreciación Mensual
632.
633.        sorter.setComparator(9, Comparator.comparingDouble(o -> {
634.            try {
635.                return Double.parseDouble(o.toString().replace(",",
636.                    "").replace("S/", ""));
637.            } catch (NumberFormatException e) {
638.                return Double.MAX_VALUE;
639.            }
640.        )); // Depreciación Acumulada
641.
642.        sorter.setComparator(10, Comparator.comparingDouble(o -> {
643.            try {
644.                return Double.parseDouble(o.toString().replace(",",
645.                    "").replace("S/", ""));
646.            } catch (NumberFormatException e) {
646.                return Double.MAX_VALUE;
646.            }
646.        )); // Valor Residual

```

```

647.         table.setRowSorter(sorter);
648.
649.         table.getTableHeader().addMouseListener(new MouseAdapter() {
650.             @Override
651.             public void mouseClicked(MouseEvent e) {
652.                 int columnIndex = table.columnAtPoint(e.getPoint());
653.                 manejarOrdenamiento(columnIndex);
654.             }
655.         });
656.     }
657.
658.     private void manejarOrdenamiento(int columnIndex) {
659.         DefaultRowSorter<?, ?> sorter = (DefaultRowSorter<?, ?>) table.getRowSorter();
660.         List<? extends RowSorter.SortKey> sortKeys = sorter.getSortKeys();
661.
662.         // Ciclo: UNSORTED → ASCENDING → DESCENDING → UNSORTED
663.         if (columnaActualOrdenada != columnIndex) {
664.             ordenActual = SortOrder.ASCENDING;
665.         } else {
666.             switch (ordenActual) {
667.                 case UNSORTED:
668.                     ordenActual = SortOrder.ASCENDING;
669.                     break;
670.                 case ASCENDING:
671.                     ordenActual = SortOrder.DESCENDING;
672.                     break;
673.                 case DESCENDING:
674.                     ordenActual = SortOrder.UNSORTED;
675.                     break;
676.             }
677.         }
678.
679.         columnaActualOrdenada = columnIndex;
680.
681.         if (ordenActual == SortOrder.UNSORTED) {
682.             sorter.setSortKeys(null);
683.         } else {
684.             sorter.setSortKeys(Collections.singletonList(new
685.             RowSorter.SortKey(columnIndex, ordenActual)));
686.         }
687.
688.         public FiltroFrame getFiltroFrame() {
689.             if (filtroFrame == null) {
690.                 filtroFrame = new FiltroFrame(this, filtrosGuardados);
691.             }
692.             return filtroFrame;
693.         }
694.
695.         private void abrirFiltroFrame() {
696.             FiltroFrame filtroFrame = new FiltroFrame(this, filtrosGuardados);
697.             filtroFrame.setVisible(true);
698.         }
699.
700.         public void actualizarFiltrosGuardados(Map<String, Object> filtros) {
701.             this.filtrosGuardados = new HashMap<>(filtros);
702.         }
703.
704.         public void filtrarActivosAvanzado(Map<String, Object> criterios) {
705.             if (criterios.isEmpty()) {
706.                 listaFiltrada = listaBaseDatos.stream()
707.                     .filter(activo ->
708.                         !activo.getEstado().equalsIgnoreCase("Renovado"))
709.                         .collect(Collectors.toList());
709.             } else {

```

```

710.             listaFiltrada = activoServicio.filtrarActivos(criterios).stream()
711.                     .filter(activo ->
712.                         !activo.getEstado().equalsIgnoreCase("Renovado"))
713.                     .collect(Collectors.toList());
714.         }
715.     }
716.
717.     public JTable getTable() {
718.         return table;
719.     }
720.
721.     private void filtrarActivos(String textoBusqueda) {
722.         textoBusqueda = textoBusqueda.toLowerCase();
723.         NonEditableTableModel modeloFiltrado = new NonEditableTableModel();
724.         modeloFiltrado.setColumnIdentifiers(new String[]{
725.             "Nº", "Código Patrimonial", "Nombre", "Categoría", "Estado",
726.             "Fecha de Adquisición", "Vida Útil", "Costo Inicial",
727.             "Depreciación Mensual", "Depreciación Acumulada", "Valor
    Residual"
728.         });
729.
730.         List<ActivoFijo> baseBusqueda = listaFiltrada != null ? listaFiltrada
: listaBaseDatos;
731.
732.         for (ActivoFijo activo : baseBusqueda.stream()
733.             .filter(a -> !a.getEstado().equalsIgnoreCase("Renovado"))
734.             .collect(Collectors.toList())) {
735.
736.             if (activo.getNombre().toLowerCase().contains(textoBusqueda) ||
737.                 activo.getCodigoPatrimonial().toLowerCase().contains(texto
    Busqueda)) {
738.
739.                 modeloFiltrado.addRow(new Object[]{
740.                     activo.getNumero(),
741.                     activo.getCodigoPatrimonial(),
742.                     activo.getNombre(),
743.                     activo.getCategoría(),
744.                     activo.getEstado(),
745.                     activo.getFechaAdquisicion(),
746.                     activo.getVidaUtil(),
747.                     String.format("%.2f", activo.getCostoInicial()),
748.                     String.format("%.2f",
    activo.calcularDepreciacionMensual()),
749.                     String.format("%.2f",
    activo.calcularDepreciacionAcumulada()),
750.                     String.format("%.2f", activo.calcularValorResidual())
751.                 });
752.             }
753.         }
754.
755.         table.setModel(modeloFiltrado);
756.
757.         TableRowSorter<TableModel> sorter = new
    TableRowSorter<>(table.getModel());
758.         configurarComparadores(sorter);
759.         table.setRowSorter(sorter);
760.     }
761.
762.     private void cargarActivosEnTabla(List<ActivoFijo> activos) {
763.         NonEditableTableModel modelo = new NonEditableTableModel();
764.         modelo.setColumnIdentifiers(new String[]{
765.             "ID", "Código Patrimonial", "Nombre", "Categoría", "Estado",
766.             "Fecha de Adquisición", "Vida Útil", "Costo Inicial",
767.             "Depreciación Mensual", "Depreciación Acumulada", "Valor
    Residual"

```

```

768.         });
769.
770.         totalCostoInicial = 0.0;
771.         totalDepreciacionMensual = 0.0;
772.         totalDepreciacionAcumulada = 0.0;
773.         totalValorResidual = 0.0;
774.         numeroDeActivos = 0;
775.
776.         List<ActivoFijo> activosFiltrados = activos.stream()
777.             .filter(activo ->
778.                 !activo.getEstado().equalsIgnoreCase("Renovado"))
779.             .collect(Collectors.toList());
780.
781.         for (ActivoFijo activo : activosFiltrados) {
782.             double depreciacionMensual = activo.calcularDepreciacionMensual();
783.             double depreciacionAcumulada =
784.                 activo.calcularDepreciacionAcumulada();
785.             double valorResidual = activo.calcularValorResidual();
786.
787.             modelo.addRow(new Object[]{
788.                 activo.getNumero(),
789.                 activo.getCodigoPatrimonial(),
790.                 activo.getNombre(),
791.                 activo.getCategoría(),
792.                 activo.getEstado(),
793.                 activo.getFechaAdquisicion(),
794.                 activo.getVidaUtil(),
795.                 String.format("%.2f", activo.getCostoInicial()),
796.                 String.format("%.2f", depreciacionMensual),
797.                 String.format("%.2f", depreciacionAcumulada),
798.                 String.format("%.2f", valorResidual)
799.             });
800.
801.             totalCostoInicial += activo.getCostoInicial();
802.             totalDepreciacionMensual += depreciacionMensual;
803.             totalDepreciacionAcumulada += depreciacionAcumulada;
804.             totalValorResidual += valorResidual;
805.             numeroDeActivos++;
806.
807.             table.setModel(modelo);
808.             actualizarVistaDeTotales();
809.
810.         private void seleccionarPrimeraCoincidencia(String textoBusqueda) {
811.             textoBusqueda = textoBusqueda.toLowerCase();
812.             for (int i = 0; i < table.getRowCount(); i++) {
813.                 int idActivo = (int) table.getValueAt(i, 0);
814.                 String nombre = table.getValueAt(i, 2).toString().toLowerCase();
815.
816.                 if (String.valueOf(idActivo).contains(textoBusqueda) ||
817.                     nombre.contains(textoBusqueda)) {
818.                     table.setRowSelectionInterval(i, i);
819.                     table.scrollRectToVisible(table.getCellRect(i, 0, true));
820.                     return;
821.                 }
822.
823.                 mostrarMensaje("No se encontraron coincidencias para: " +
824.                     textoBusqueda);
825.
826.             public int obtenerIndicePorNúmero(int númeroActivo) {
827.                 for (int i = 0; i < listaBaseDatos.size(); i++) {
828.                     if (listaBaseDatos.get(i).getNúmero() == númeroActivo) {
829.                         return i;

```

```

830.         }
831.     }
832.     return -1;
833. }
834.
835. public void cargarActivosEnTabla() {
836.     model.setRowCount(0);
837.
838.     totalCostoInicial = 0.0;
839.     totalDepreciacionMensual = 0.0;
840.     totalDepreciacionAcumulada = 0.0;
841.     totalValorResidual = 0.0;
842.
843.     List<ActivoFijo> activosFiltrados = listaBaseDatos.stream()
844.         .filter(a -> !a.getEstado().equalsIgnoreCase("Renovado"))
845.         .collect(Collectors.toList());
846.
847.     for (ActivoFijo activo : activosFiltrados) {
848.         double depreciacionMensual = activo.calcularDepreciacionMensual();
849.         double depreciacionAcumulada =
850.             activo.calcularDepreciacionAcumulada();
851.         double valorResidual = activo.calcularValorResidual();
852.
853.         model.addRow(new Object[]{
854.             activo.getNumero(),
855.             activo.getCodigoPatrimonial(),
856.             activo.getNombre(),
857.             activo.getCategoría(),
858.             activo.getEstado(),
859.             activo.getFechaAdquisicion(),
860.             activo.getVidaUtil(),
861.             String.format("%.2f", activo.getCostoInicial()),
862.             String.format("%.2f", depreciacionMensual),
863.             String.format("%.2f", depreciacionAcumulada),
864.             String.format("%.2f", valorResidual)
865.         });
866.         totalCostoInicial += activo.getCostoInicial();
867.         totalDepreciacionMensual += depreciacionMensual;
868.         totalDepreciacionAcumulada += depreciacionAcumulada;
869.         totalValorResidual += valorResidual;
870.     }
871.
872.     numeroDeActivos = activosFiltrados.size();
873.     actualizarVistaDeTotales();
874. }
875.
876. private void agregarActivo() {
877.     String nombre = txtNombre.getText();
878.
879.     Date fechaAdquisicion = dateChooserFechaAdquisicion.getDate();
880.     if (fechaAdquisicion == null) {
881.         mostrarMensaje("Por favor seleccione una fecha de adquisición.");
882.         return;
883.     }
884.
885.     SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
886.     String fechaAdquisicionStr;
887.     try {
888.         fechaAdquisicionStr = dateFormat.format(fechaAdquisicion);
889.     } catch (Exception e) {
890.         mostrarMensaje("Error al formatear la fecha: " + e.getMessage());
891.         return;
892.     }
893.
894.     String errorNombre = ValidadorActivo.validarNombre(nombre);

```

```

895.             if (errorNombre != null) {
896.                 mostrarMensaje(errorNombre);
897.                 return;
898.             }
899.
900.             String errorCategoria =
901.                 ValidadorActivo.validarCategoria(comboCategoria);
902.                 if (errorCategoria != null) {
903.                     mostrarMensaje(errorCategoria);
904.                     return;
905.                 }
906.
907.             String errorFecha =
908.                 ValidadorActivo.validarFechaAdquisicion(dateChooserFechaAdquisicion);
909.                 if (errorFecha != null) {
910.                     mostrarMensaje(errorFecha);
911.                     return;
912.                 }
913.
914.                 Sede sedeSeleccionada = (Sede) comboSede.getSelectedItem();
915.                 if (sedeSeleccionada == null) {
916.                     mostrarMensaje("Por favor seleccione una sede.");
917.                     return;
918.                 }
919.
920.                 try {
921.                     double costoInicial =
922.                         limpiarFormatoCostoInicial(txtCostoInicial.getText());
923.                         if (costoInicial == -1) return;
924.
925.                     String categoria = (String) comboCategoria.getSelectedItem();
926.
927.                     ActivoFijo nuevoActivo = new ActivoFijo(
928.                         nombre, categoria, costoInicial, fechaAdquisicionStr,
929.                         "Alta",
930.                         sedeSeleccionada
931.
932.                         double depreciacionMensual =
933.                             nuevoActivo.calcularDepreciacionMensual();
934.                             double depreciacionAcumulada =
935.                               nuevoActivo.calcularDepreciacionAcumulada();
936.                               double valorResidual = nuevoActivo.calcularValorResidual();
937.
938.                               GuardarActivoResultado resultado =
939.                                 activoServicio.guardarActivoEnBD(
940.                                     nombre, categoria, nuevoActivo.getEstado(),
941.                                     fechaAdquisicionStr,
942.                                     determinarVidaUtil(categoria), costoInicial,
943.                                     depreciacionMensual,
944.                                     depreciacionAcumulada, valorResidual,
945.                                     sedeSeleccionada.getId()
946.                                     );
947.
948.                                     if (resultado == null) {
949.                                         mostrarMensaje("Error al registrar el activo en la base de
950. datos.");
951.                                         return;
952.                                     }
953.
954.                                     nuevoActivo.setCodigoPatrimonial(resultado.getCodigoPatrimonial())
955. ;
956.                                     nuevoActivo.setId(resultado.getId());
957.                                     nuevoActivo.setNumero(listaBaseDatos.size() + 1);
958.
959.                                     listaBaseDatos.add(nuevoActivo);

```

```

949.         model.addRow(new Object[]{
950.             resultado.getId(),
951.             resultado.getCodigoPatrimonial(),
952.             nombre,
953.             categoria,
954.             nuevoActivo.getEstado(),
955.             fechaAdquisicionStr,
956.             nuevoActivo.getVidaUtil(),
957.             String.format("%.2f", costoInicial),
958.             String.format("%.2f", depreciacionMensual),
959.             String.format("%.2f", depreciacionAcumulada),
960.             String.format("%.2f", valorResidual)
961.         });
962.
963.         totalCostoInicial += costoInicial;
964.         totalDepreciacionMensual += depreciacionMensual;
965.         totalDepreciacionAcumulada += depreciacionAcumulada;
966.         totalValorResidual += valorResidual;
967.         numeroDeActivos++;
968.
969.         activoServicio.registrarActivo(resultado.getId());
970.         mostrarMensaje("Activo registrado correctamente.");
971.         getFiltroFrame().resetearFiltros(this);
972.         txtBuscar.setText("");
973.         actualizarVistaDeTotales();
974.         limpiarCampos();
975.     } catch (Exception e) {
976.         mostrarMensaje("Error al registrar el activo: " + e.getMessage());
977.     }
978. }
979.
980.     private void modificarActivo() {
981.         if (!enModoEdicion) {
982.             int filaVista = table.getSelectedRow();
983.             if (filaVista == -1) {
984.                 mostrarMensaje("Por favor seleccione un activo para
modificar.");
985.                 return;
986.             }
987.
988.             int numeroActivo = (int) table.getValueAt(filaVista, 0);
989.             int indiceModelo = listaBaseDatos.stream()
990.                 .filter(a -> a.getNumero() == numeroActivo)
991.                 .findFirst()
992.                 .map(listaBaseDatos::indexOf)
993.                 .orElse(-1);
994.
995.             if (indiceModelo == -1) {
996.                 mostrarMensaje("No se encontró el activo en el modelo de
datos.");
997.                 return;
998.             }
999.
1000.             ActivoFijo activoSeleccionado = listaBaseDatos.get(indiceModelo);
1001.             if (activoSeleccionado.getEstado().equals("Depreciado") ||
activoSeleccionado.getEstado().equals("Baja") ||
activoSeleccionado.getEstado().equals("Baja por venta")) {
1002.                 mostrarMensaje("No se puede modificar un activo que está en
estado 'Depreciado' o 'Baja'.");
1003.                 return;
1004.             }
1005.
1006.             nombreOriginal = activoSeleccionado.getNombre();
1007.             categoriaOriginal = activoSeleccionado.getCategoría();
1008.
1009.             txtNombre.setText(activoSeleccionado.getNombre());

```

```

1010.         comboCategoria.setSelectedItem(activoSeleccionado.getCategoria());
1011.
1012.         dateChooserFechaAdquisicion.setEnabled(false);
1013.         txtCostoInicial.setEnabled(false);
1014.         comboSede.setEnabled(false);
1015.
1016.         enModoEdicion = true;
1017.         indiceActivoEnEdicion = indiceModelo;
1018.         btnModificar.setText("Guardar Cambios");
1019.
1020.         btnRegistrar.setEnabled(false);
1021.         btnDarBaja.setEnabled(false);
1022.         btnVender.setEnabled(false);
1023.         comboTipoBaja.setEnabled(false);
1024.         btnCambios.setEnabled(false);
1025.         btnProyectarDepreciacion.setEnabled(false);
1026.         btnInventario.setEnabled(false);
1027.         txtBuscar.setEnabled(false);
1028.         btnBuscar.setEnabled(false);
1029.         btnExportar.setEnabled(false);
1030.         btnCerrarSesion.setEnabled(false);
1031.         btnResetearBusqueda.setEnabled(false);
1032.         btnRecargar.setEnabled(false);
1033.         btnRevaluar.setEnabled(false);
1034.         btnRenovar.setEnabled(false);
1035.         btnFiltrar.setEnabled(false);
1036.         btnAnalisisDeBajas.setEnabled(false);
1037.         btnUbicacionActivos.setEnabled(false);
1038.
1039.     } else {
1040.         String nuevoNombre = txtNombre.getText();
1041.         String nuevaCategoria = (String) comboCategoria.getSelectedItem();
1042.
1043.         String errorNombre = ValidadorActivo.validarNombre(nuevoNombre);
1044.         if (errorNombre != null) {
1045.             mostrarMensaje(errorNombre);
1046.             return;
1047.         }
1048.
1049.         String errorCategoria =
1050.             ValidadorActivo.validarCategoria(comboCategoria);
1051.         if (errorCategoria != null) {
1052.             mostrarMensaje(errorCategoria);
1053.             return;
1054.         }
1055.         boolean categoriaCambiada =
1056.             !categoriaOriginal.equals(nuevaCategoria);
1057.         if (!categoriaCambiada && nombreOriginal.equals(nuevoNombre)) {
1058.             salirModoEdicion();
1059.             mostrarMensaje("No se realizaron cambios. El activo se
mantiene igual.");
1060.         } else {
1061.             int numeroActivo = (int)
1062.                 table.getValueAt(table.getSelectedRow(), 0);
1063.                 ActivoFijo activoSeleccionado = listaBaseDatos.stream()
1064.                     .filter(a -> a.getNumero() == numeroActivo)
1065.                     .findFirst()
1066.                     .orElse(null);
1067.
1068.             if (activoSeleccionado == null) {
1069.                 mostrarMensaje("Error: No se encontró el activo en la base
de datos.");
1070.             }

```

```

1071.             activoSeleccionado.setNombre(nuevoNombre);
1072.             activoSeleccionado.setCategoria(nuevaCategoria);
1073.
1074.             if (categoriaCambiada) {
1075.                 activoSeleccionado.setVidaUtil(nuevaCategoria);
1076.                 double nuevaDepreciacionMensual =
1077.                     activoSeleccionado.calcularDepreciacionMensual();
1078.                 double nuevaDepreciacionAcumulada =
1079.                     activoSeleccionado.calcularDepreciacionAcumulada();
1080.                 double nuevoValorResidual =
1081.                     activoSeleccionado.calcularValorResidual();
1082.
1083.                     // Actualizar totales
1084.                     totalCostoInicial -= activoSeleccionado.getCostoInicial();
1085.                     totalDepreciacionMensual =
1086.                         activoSeleccionado.calcularDepreciacionMensual();
1087.                     totalDepreciacionAcumulada =
1088.                         activoSeleccionado.calcularDepreciacionAcumulada();
1089.                     totalValorResidual =
1090.                         activoSeleccionado.calcularValorResidual();
1091.
1092.                     activoServicio.modificarActivoEnBD(
1093.                         activoSeleccionado.getNumero(),
1094.                         nuevoNombre,
1095.                         nuevaCategoria,
1096.                         activoSeleccionado.getEstado(),
1097.                         activoSeleccionado.getFechaAdquisicion(),
1098.                         activoSeleccionado.getVidaUtil(),
1099.                         activoSeleccionado.getCostoInicial(),
1100.                         activoSeleccionado.calcularDepreciacionMensual(),
1101.                         activoSeleccionado.calcularDepreciacionAcumulada(),
1102.                         activoSeleccionado.calcularValorResidual()
1103.                     );
1104.
1105.                     int filaVista = table.getSelectedRow();
1106.                     model.setValueAt(activoSeleccionado.getCodigoPatrimonial(),
1107.                         filaVista, 1);
1108.                     model.setValueAt(nuevoNombre, filaVista, 2);
1109.                     model.setValueAt(nuevaCategoria, filaVista, 3);
1110.                     model.setValueAt(activoSeleccionado.getEstado(), filaVista,
1111.                         4);
1112.                     model.setValueAt(activoSeleccionado.getVidaUtil(), filaVista,
1113.                         6);
1114.                     model.setValueAt(String.format("%.2f",
1115.                         activoSeleccionado.getCostoInicial()), filaVista, 7);
1116.                     model.setValueAt(String.format("%.2f",
1117.                         activoSeleccionado.calcularDepreciacionMensual()), filaVista, 8);
1118.                     model.setValueAt(String.format("%.2f",
1119.                         activoSeleccionado.calcularDepreciacionAcumulada()), filaVista, 9);
1120.                     model.setValueAt(String.format("%.2f",
1121.                         activoSeleccionado.calcularValorResidual()), filaVista, 10);
1122.
1123.                     salirModoEdicion();
1124.                     String detallesModificacion = "\n"
1125.                         + (!nombreOriginal.equals(nuevoNombre) ? "Nombre: '" +
1126.                             nombreOriginal + "' -> '" + nuevoNombre + "'\n' : "")
1127.                         + (!categoriaOriginal.equals(nuevaCategoria) ?
1128.                             "Categoria: '" + categoriaOriginal + "' -> '" + nuevaCategoria + "'\n' : "");
1129.                     activoServicio.modificarActivo(activoSeleccionado.getNumero(),
1130.                         detallesModificacion);
1131.                     mostrarMensaje("Activo modificado correctamente.");

```

```

1121.             getFiltroFrame().resetearFiltros(this);
1122.             txtBuscar.setText("");
1123.             recargarDatos();
1124.         }
1125.     }
1126. }
1127.
1128.     private void salirModoEdicion() {
1129.         enModoEdicion = false;
1130.         indiceActivoEnEdicion = -1;
1131.         btnModificar.setText("Modificar");
1132.
1133.         btnRegistrar.setEnabled(true);
1134.         btnDarBaja.setEnabled(true);
1135.         btnVender.setEnabled(true);
1136.         comboTipoBaja.setEnabled(true);
1137.         btnCambios.setEnabled(true);
1138.         btnProyectarDepreciacion.setEnabled(true);
1139.         btnInventario.setEnabled(true);
1140.         txtBuscar.setEnabled(true);
1141.         btnBuscar.setEnabled(true);
1142.         btnExportar.setEnabled(true);
1143.         btnCerrarSesion.setEnabled(true);
1144.         btnResetearBusqueda.setEnabled(true);
1145.         btnRecargar.setEnabled(true);
1146.         btnRevaluar.setEnabled(true);
1147.         btnRenovar.setEnabled(true);
1148.         btnFiltrar.setEnabled(true);
1149.         btnAnalisisDeBajas.setEnabled(true);
1150.         btnUbicacionActivos.setEnabled(true);
1151.
1152.         dateChooserFechaAdquisicion.setEnabled(true);
1153.         txtCostoInicial.setEnabled(true);
1154.         comboSede.setEnabled(true);
1155.
1156.         limpiarCampos();
1157.     }
1158.
1159.     private void darDeBajaActivo() {
1160.         int filaVista = table.getSelectedRow();
1161.         if (filaVista == -1) {
1162.             mostrarMensaje("Seleccione un activo para dar de baja.");
1163.             return;
1164.         }
1165.
1166.         int numeroActivo = (int) table.getValueAt(filaVista, 0);
1167.
1168.         ActivoFijo activoSeleccionado = listaBaseDatos.stream()
1169.             .filter(a -> a.getNumero() == numeroActivo)
1170.             .findFirst()
1171.             .orElse(null);
1172.
1173.         if (activoSeleccionado == null) {
1174.             mostrarMensaje("Error: No se encontró el activo en la base de
datos.");
1175.             return;
1176.         }
1177.
1178.         String tipoBaja = (String) comboTipoBaja.getSelectedItem();
1179.
1180.         if (tipoBaja == null || tipoBaja.isEmpty()) {
1181.             mostrarMensaje("Seleccione un tipo de baja.");
1182.             return;
1183.         }
1184.

```

```

1185.             if (tipoBaja.equals("Fin de vida útil") &&
1186.                 !activoSeleccionado.getEstado().equals("Depreciado")) {
1187.                     mostrarMensaje("El activo no está depreciado y no puede darse de
1188.                         baja por 'Fin de vida útil'.");
1189.                 return;
1190.             }
1191.             if (activoSeleccionado.getEstado().equals("Depreciado") ||
1192.                 activoSeleccionado.getEstado().equals("Baja") ||
1193.                 activoSeleccionado.getEstado().equals("Baja por venta")) {
1194.                     mostrarMensaje("No se puede dar de baja un activo que está en
1195.                         estado 'Depreciado' o 'Baja'.");
1196.                 return;
1197.             }
1198.             String motivoBaja;
1199.             do {
1200.                 motivoBaja = JOptionPane.showInputDialog(this, "Ingrese el motivo
1201.                     de la baja:");
1202.                 if (motivoBaja == null) {
1203.                     return;
1204.                 }
1205.                 motivoBaja = motivoBaja.trim();
1206.                 if (motivoBaja.isEmpty()) {
1207.                     JOptionPane.showMessageDialog(this, "El motivo de baja no
1208.                         puede estar vacío.", "Advertencia", JOptionPane.WARNING_MESSAGE);
1209.                 }
1210.             } while (motivoBaja.isEmpty());
1211.             activoSeleccionado.darDeBaja(motivoBaja, tipoBaja);
1212.             activoServicio.actualizarEstadoActivo(numeroActivo, "Baja");
1213.             activoServicio.darDeBaja(numeroActivo, motivoBaja, tipoBaja);
1214.             model.setValueAt("Baja", filaVista, 4); // Columna 4: Estado
1215.             mostrarMensaje("Activo dado de baja como: " + tipoBaja);
1216.             getFiltroFrame().resetearFiltros(this);
1217.             txtBuscar.setText("");
1218.             actualizarVistaDeTotales();
1219.             limpiarCampos();
1220.         }
1221.     }
1222.     private void venderActivo() {
1223.         int filaVista = table.getSelectedRow();
1224.         if (filaVista == -1) {
1225.             mostrarMensaje("Debe seleccionar un activo para vender.");
1226.             return;
1227.         }
1228.         int numeroActivo = (int) table.getValueAt(filaVista, 0);
1229.         ActivoFijo activoSeleccionado = listaBaseDatos.stream()
1230.             .filter(a -> a.getNumero() == numeroActivo)
1231.             .findFirst()
1232.             .orElse(null);
1233.         if (activoSeleccionado == null) {
1234.             mostrarMensaje("Error: No se encontró el activo en la base de
1235.                 datos.");
1236.             return;
1237.         }
1238.         if (activoSeleccionado.getEstado().equalsIgnoreCase("Baja") ||
1239.             activoSeleccionado.getEstado().equalsIgnoreCase("Baja por
1240.                 venta")) {

```

```

1243.             mostrarMensaje("No se puede vender un activo que ya está en estado
1244.                 de baja.");
1245.             }
1246.
1247.             double valorResidual = activoSeleccionado.calcularValorResidual();
1248.             if (valorResidual <= 0) {
1249.                 mostrarMensaje("No se puede vender un activo con valor residual
1250.                     0.");
1251.             }
1252.
1253.             String nombre = activoSeleccionado.getNombre();
1254.             String categoria = activoSeleccionado.getCategoría();
1255.             String fechaAdquisicion = activoSeleccionado.getFechaAdquisicion();
1256.             double costoInicial = activoSeleccionado.getCostoInicial();
1257.
1258.             VentaFrame ventaFrame = new VentaFrame(this, numeroActivo, nombre,
1259.                 categoria, fechaAdquisicion, costoInicial, valorResidual);
1260.             ventaFrame.setVisible(true);
1261.         }
1262.
1263.         private void activarBoton() {
1264.             if (enModoEdicion) {
1265.                 btnModificar.doClick();
1266.             } else {
1267.                 btnRegistrar.doClick();
1268.             }
1269.
1270.             private double limpiarFormatoCostoInicial(String costoInicialStr) {
1271.                 String valorLimpio = costoInicialStr.replaceAll("[^\d.]", "");
1272.
1273.                 int ultimoPunto = valorLimpio.lastIndexOf('.');
1274.                 if (ultimoPunto != -1) {
1275.                     valorLimpio = valorLimpio.substring(0, ultimoPunto).replace(".",",
1276.                         "") + valorLimpio.substring(ultimoPunto);
1277.                 }
1278.
1279.                 try {
1280.                     double costoInicial = Double.parseDouble(valorLimpio);
1281.
1282.                     if (costoInicial <= 0) {
1283.                         mostrarMensaje("El costo inicial debe ser un número mayor a
1284.                             0.");
1285.                     }
1286.
1287.                     return costoInicial;
1288.                 } catch (NumberFormatException e) {
1289.                     mostrarMensaje("Formato incorrecto para el costo inicial.");
1290.                     return -1;
1291.                 }
1292.
1293.             private void agregarCampo(JPanel panel, GridBagConstraints gbc, int fila,
1294.                 String label, Component componente) {
1295.                 gbc.gridx = 0;
1296.                 gbc.gridy = fila;
1297.                 panel.add(new JLabel(label), gbc);
1298.
1299.                 gbc.gridx = 1;
1300.                 panel.add(componente, gbc);
1301.
1302.             private void limpiarCampos() {

```

```

1303.         txtNombre.setText("");
1304.         comboCategoria.setSelectedIndex(0);
1305.         comboSede.setSelectedIndex(0);
1306.         dateChooserFechaAdquisicion.setDate(null);
1307.         txtVidaUtil.setText("");
1308.         txtCostoInicial.setText("");
1309.     }
1310.
1311.     public void actualizarEstadoActivoEnTabla(int idActivo, String
1312.         nuevoEstado) {
1312.         for (int i = 0; i < model.getRowCount(); i++) {
1313.             if ((int) model.getValueAt(i, 0) == idActivo) {
1314.                 model.setValueAt(nuevoEstado, i, 4);
1315.                 break;
1316.             }
1317.         }
1318.     }
1319.
1320.     public boolean tablaEstaVacia() {
1321.         return table.getRowCount() == 0;
1322.     }
1323.
1324.     public static String formatearNumero(double numero) {
1325.         if (numero >= 1_000_000_000.0) { // Billones
1326.             return String.format("%.2f T", numero / 1_000_000_000.0);
1327.         } else if (numero >= 1_000_000_000) { // Miles de millones
1328.             return String.format("%.2f B", numero / 1_000_000_000);
1329.         } else if (numero >= 1_000_000) { // Millones
1330.             return String.format("%.2f M", numero / 1_000_000);
1331.         } else if (numero >= 1_000) { // Miles
1332.             return String.format("%.2f K", numero / 1_000);
1333.         } else {
1334.             return String.format("%.2f", numero);
1335.         }
1336.     }
1337.
1338.     public void actualizarVistaDeTotales() {
1339.         panelTotales.removeAll();
1340.         numeroDeActivos = model.getRowCount();
1341.         panelTotales.add(new JLabel("Número de Activos: " + numeroDeActivos +
1342.             " | "));
1342.         panelTotales.add(new JLabel("Total de Costo Inicial: S/" +
1343.             formatearNumero(totalCostoInicial) + " | "));
1343.         panelTotales.add(new JLabel("Total de Depreciación Mensual: S/" +
1344.             formatearNumero(totalDepreciacionMensual) + " | "));
1344.         panelTotales.add(new JLabel("Total de Depreciación Acumulada: S/" +
1345.             formatearNumero(totalDepreciacionAcumulada) + " | "));
1345.         panelTotales.add(new JLabel("Total de Valor Residual: S/" +
1346.             formatearNumero(totalValorResidual)));
1346.         panelTotales.revalidate();
1347.         panelTotales.repaint();
1348.     }
1349.
1350.     public void mostrarMensaje(String mensaje) {
1351.         SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
1352.         String horaActual = sdf.format(new Date());
1353.
1354.         txtAreaMensajes.append("[" + horaActual + "] " + mensaje + "\n");
1355.
1356.         txtAreaMensajes.setCaretPosition(txtAreaMensajes.getDocument().getLength());
1357.     }
1358.
1359.     private void verHistorial() {
1360.         List<String> detallesMovimientos =
1360.             activoServicio.cargarHistorialMovimientos();

```

```

1361.         if (detallesMovimientos.isEmpty()) {
1362.             mostrarMensaje("No hay movimientos registrados.");
1363.             return;
1364.         }
1365.     }
1366.
1367.     StringBuilder historialTexto = new StringBuilder();
1368.     for (String detalle : detallesMovimientos) {
1369.         historialTexto.insert(0, detalle + "\n");
1370.     }
1371.
1372.     String historialFinal = historialTexto.toString().replaceAll("\n+$",
1373.     "");
1374.     JTextArea areaTexto = new JTextArea(historialFinal, 20, 50);
1375.     areaTexto.setLineWrap(true);
1376.     areaTexto.setWrapStyleWord(true);
1377.     areaTexto.setEditable(false);
1378.     JScrollPane scrollPane = new JScrollPane(areaTexto);
1379.
1380.     areaTexto.setCaretPosition(areaTexto.getDocument().getLength());
1381.
1382.     JOptionPane.showMessageDialog(null, scrollPane, "Historial de
cambios", JOptionPane.INFORMATION_MESSAGE);
1383. }
1384.
1385.     private void exportarReporte() {
1386.         String nombreArchivo = "reporte_valorium.xlsx";
1387.
1388.         if (table.getRowCount() == 0) {
1389.             mostrarMensaje("No hay activos disponibles en la tabla para
exportar.");
1390.             return;
1391.         }
1392.
1393.         try {
1394.             Workbook workbook = new XSSFWorkbook();
1395.             Sheet sheet = workbook.createSheet("Activos");
1396.
1397.             // Estilo para el encabezado
1398.             CellStyle headerStyle = workbook.createCellStyle();
1399.             org.apache.poi.ss.usermodel.Font headerFont =
workbook.createFont();
1400.             headerFont.setBold(true);
1401.             headerStyle.setFont(headerFont);
1402.             headerStyle.setFillForegroundColor(IndexedColors.GREY_25_PERCENT.g
etIndex());
1403.             headerStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);
1404.
1405.             // Estilo para los datos
1406.             CellStyle dataStyle = workbook.createCellStyle();
1407.             dataStyle.setBorderBottom(BorderStyle.THIN);
1408.             dataStyle.setBorderTop(BorderStyle.THIN);
1409.             dataStyle.setBorderLeft(BorderStyle.THIN);
1410.             dataStyle.setBorderRight(BorderStyle.THIN);
1411.             dataStyle.setAlignment(HorizontalAlignment.RIGHT);
1412.
1413.             // Fila de encabezado
1414.             Row headerRow = sheet.createRow(0);
1415.             String[] columnas = {"Nº", "Código Patrimonial", "Nombre",
"Categoría", "Estado", "Fecha de Adquisición",
1416.                         "Vida Útil (años)", "Costo Inicial", "Depreciación
Mensual", "Depreciación Acumulada",
1417.                         "Valor Residual"};
1418.             for (int i = 0; i < columnas.length; i++) {
1419.                 Cell cell = headerRow.createCell(i);

```

```

1420.                 cell.setCellValue(columnas[i]);
1421.                 cell.setCellStyle(headerStyle);
1422.             }
1423.
1424.             int rowNum = 1;
1425.
1426.             for (int i = 0; i < table.getRowCount(); i++) {
1427.                 Row row = sheet.createRow(rowNum++);
1428.
1429.                 for (int j = 0; j < table.getColumnCount(); j++) {
1430.                     Object valorCelda = table.getValueAt(i, j);
1431.                     Cell cell = row.createCell(j);
1432.
1433.                     if (valorCelda instanceof Number) {
1434.                         cell.setCellValue(((Number)
1435.                             valorCelda).doubleValue());
1436.                     } else if (valorCelda != null) {
1437.                         cell.setCellValue(valorCelda.toString());
1438.                     }
1439.                     cell.setCellStyle(dataStyle);
1440.                 }
1441.
1442.                 for (int i = 0; i < columnas.length; i++) {
1443.                     sheet.autoSizeColumn(i);
1444.                 }
1445.
1446.                 JFileChooser fileChooser = new JFileChooser();
1447.                 fileChooser.setDialogTitle("Guardar reporte");
1448.                 fileChooser.setSelectedFile(new File(nombreArchivo));
1449.                 int userSelection = fileChooser.showSaveDialog(this);
1450.
1451.                 if (userSelection == JFileChooser.APPROVE_OPTION) {
1452.                     File archivoSeleccionado = fileChooser.getSelectedFile();
1453.
1454.                     try (FileOutputStream fileOut = new
1455.                         FileOutputStream(archivoSeleccionado)) {
1456.                         workbook.write(fileOut);
1457.                         JOptionPane.showMessageDialog(this, "Reporte exportado
correctamente a: " + archivoSeleccionado.getAbsolutePath());
1458.                         mostrarMensaje("Reporte exportado correctamente.");
1459.                     }
1460.                     workbook.close();
1461.
1462.                 } catch (Exception ex) {
1463.                     ex.printStackTrace();
1464.                     JOptionPane.showMessageDialog(this, "Error al exportar el reporte:
" + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
1465.                     mostrarMensaje("Error al exportar el reporte.");
1466.                 }
1467.             }
1468.
1469.             private void proyectarDepreciacion() {
1470.                 int filaVista = table.getSelectedRow();
1471.                 if (filaVista == -1) {
1472.                     mostrarMensaje("Por favor seleccione un activo para proyectar la
depreciación.");
1473.                     return;
1474.                 }
1475.
1476.                 int numeroActivo = (int) table.getValueAt(filaVista, 0);
1477.                 int indiceModelo = obtenerIndicePorNumero(numeroActivo);
1478.                 if (indiceModelo == -1) {
1479.                     mostrarMensaje("No se encontró el activo en el modelo de datos.");
1480.                     return;

```

```

1481.         }
1482.         ActivoFijo activoSeleccionado = listaBaseDatos.get(indiceModelo);
1484.
1485.         if ("Terreno".equalsIgnoreCase(activoSeleccionado.getCategoría()) ||
1486.             activoSeleccionado.getVidaUtil() == 0) {
1486.             mostrarMensaje("No se puede proyectar la depreciación de un
1487.             terreno.");
1487.             return;
1488.         }
1489.
1490.         int vidaUtil = activoSeleccionado.getVidaUtil();
1491.         double depreciacionMensual =
1491.             activoSeleccionado.calcularDepreciacionMensual();
1492.
1493.         LocalDate fechaInicio =
1493.             LocalDate.parse(activoSeleccionado.getFechaAdquisicion(),
1493.                 DateTimeFormatter.ofPattern("dd-MM-yyyy")).plusMonths(1);
1494.         LocalDate fechaFin = fechaInicio.plusYears(vidaUtil);
1495.
1496.         String[] columnNames = {"Mes", "Año", "Depreciación Mensual",
1496.             "Depreciación Acumulada", "Valor en Libros"};
1497.         Object[][] data = new Object[vidaUtil * 12][5];
1498.
1499.         double depreciacionAcumulada = 0;
1500.         double valorEnLibros = activoSeleccionado.getCostoInicial();
1501.
1502.         for (int año = 0; año < vidaUtil; año++) {
1503.             for (int mes = 0; mes < 12; mes++) {
1504.                 LocalDate fechaActual = fechaInicio.plusMonths(año * 12 +
1504.                     mes);
1505.
1506.                 if (fechaActual.isBefore(fechaFin) ||
1506.                     fechaActual.isEqual(fechaFin)) {
1507.                     String mesNombre =
1507.                         fechaActual.getMonth().getDisplayName(TextStyle.FULL, new Locale("es", "ES"));
1508.                     data[año * 12 + mes][0] = mesNombre;
1509.                     data[año * 12 + mes][1] = fechaActual.getYear();
1510.                     data[año * 12 + mes][2] = String.format("S/ %.2f",
1510.                         depreciacionMensual);
1511.
1512.                     depreciacionAcumulada += depreciacionMensual;
1513.                     data[año * 12 + mes][3] = String.format("S/ %.2f",
1513.                         depreciacionAcumulada);
1514.
1515.                     valorEnLibros -= depreciacionMensual;
1516.                     data[año * 12 + mes][4] = String.format("S/ %.2f",
1516.                         Math.max(0, valorEnLibros));
1517.                 }
1518.             }
1519.         }
1520.
1521.         JTable tableProyeccion = new JTable(data, columnNames) {
1522.             @Override
1523.             public boolean isCellEditable(int row, int column) {
1524.                 return false;
1525.             }
1526.         };
1527.         tableProyeccion.getTableHeader().setReorderingAllowed(false);
1528.         tableProyeccion.setFillsViewportHeight(true);
1529.         tableProyeccion.setPreferredScrollableViewportSize(new Dimension(800,
1529.             400));
1530.
1531.         JScrollPane scrollPane = new JScrollPane(tableProyeccion);
1532.
1533.         JButton btnExportar = new JButton("Exportar a Excel");

```

```

1534.         btnExportar.addActionListener(e ->
1535.             exportarProyeccionAExcel(activoSeleccionado, data, columnNames));
1536.         JDialog dialog = new JDialog((Frame) null, "Proyección de Depreciación
1537.             para '" + activoSeleccionado.getNombre() + "'", true);
1538.             dialog.setLayout(new BorderLayout());
1539.             dialog.add(scrollPane, BorderLayout.CENTER);
1539.
1540.             JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.CENTER));
1541.             panelBotones.add(btnExportar);
1542.             JButton btnCerrar = new JButton("Cerrar");
1543.             btnCerrar.addActionListener(e -> dialog.dispose());
1544.             panelBotones.add(btnCerrar);
1545.
1546.             dialog.add(panelBotones, BorderLayout.SOUTH);
1547.
1548.             dialog.pack();
1549.             dialog.setLocationRelativeTo(null);
1550.             dialog.setVisible(true);
1551.     }
1552.
1553.     private void exportarProyeccionAExcel(ActivoFijo activo, Object[][][] data,
1554.     String[] columnNames) {
1555.         String nombreArchivo = activo.getNombre() +
1556.             "_Proyeccion_Depreciacion.xlsx";
1555.
1556.         try (Workbook workbook = new XSSFWorbook()) {
1557.             Sheet sheet = workbook.createSheet("Proyección");
1558.
1559.             // Estilos de encabezado
1560.             CellStyle headerStyle = workbook.createCellStyle();
1561.             Font headerFont = workbook.createFont();
1562.             headerFont.setBold(true);
1563.             headerStyle.setFont(headerFont);
1564.             headerStyle.setFillForegroundColor(IndexedColors.GREY_25_PERCENT.g
1565.                 etIndex());
1565.             headerStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);
1566.
1567.             Row headerRow = sheet.createRow(0);
1568.             for (int i = 0; i < columnNames.length; i++) {
1569.                 Cell cell = headerRow.createCell(i);
1570.                 cell.setCellValue(columnNames[i]);
1571.                 cell.setStyle(headerStyle);
1572.             }
1573.
1574.             for (int i = 0; i < data.length; i++) {
1575.                 Row row = sheet.createRow(i + 1);
1576.                 for (int j = 0; j < data[i].length; j++) {
1577.                     Cell cell = row.createCell(j);
1578.                     if (data[i][j] instanceof String) {
1579.                         cell.setCellValue((String) data[i][j]);
1580.                     } else if (data[i][j] instanceof Double) {
1581.                         cell.setCellValue((Double) data[i][j]);
1582.                     } else if (data[i][j] instanceof Integer) {
1583.                         cell.setCellValue((Integer) data[i][j]);
1584.                     }
1585.                 }
1586.             }
1587.
1588.             for (int i = 0; i < columnNames.length; i++) {
1589.                 sheet.autoSizeColumn(i);
1590.             }
1591.
1592.             JFileChooser fileChooser = new JFileChooser();
1593.             fileChooser.setDialogTitle("Guardar reporte");
1594.             fileChooser.setSelectedFile(new File(nombreArchivo));

```

```

1595.             int userSelection = fileChooser.showSaveDialog(null);
1596.
1597.             if (userSelection == JFileChooser.APPROVE_OPTION) {
1598.                 File archivoSeleccionado = fileChooser.getSelectedFile();
1599.                 try (FileOutputStream fileOut = new
1600.                     FileOutputStream(archivoSeleccionado)) {
1601.                         workbook.write(fileOut);
1602.                         JOptionPane.showMessageDialog(null, "Proyección exportada
correctamente a: " + archivoSeleccionado.getAbsolutePath());
1603.                     }
1604.                 } catch (IOException ex) {
1605.                     JOptionPane.showMessageDialog(null, "Error al exportar la
proyección: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
1606.                 }
1607.             }
1608.
1609.             public void mostrarNotificacionEmergente(String mensaje, int
idNotificacion) {
1610.                 String titulo = "Notificación de Activo";
1611.
1612.                 notificacionFrame.mostrarNotificacion(
1613.                     titulo,
1614.                     mensaje,
1615.                     idNotificacion,
1616.                     () -> {
1617.                         actualizarEstadoNotificacion(idNotificacion, "Listo");
1618.                     },
1619.                     () -> {
1620.                         actualizarEstadoNotificacion(idNotificacion, "Recordar");
1621.                     }
1622.                 );
1623.             }
1624.
1625.             private void actualizarEstadoNotificacion(int idNotificacion, String
nuevoEstado) {
1626.                 String sql = "UPDATE notificaciones SET estado = ?, fecha_creacion =
CURRENT_TIMESTAMP WHERE id = ?";
1627.
1628.                 try (Connection conn = ConexionBD.getConnection();
1629.                      PreparedStatement pstmt = conn.prepareStatement(sql)) {
1630.
1631.                     pstmt.setString(1, nuevoEstado);
1632.                     pstmt.setInt(2, idNotificacion);
1633.                     pstmt.executeUpdate();
1634.                 } catch (SQLException e) {
1635.                     e.printStackTrace();
1636.                 }
1637.             }
1638.
1639.             public void mostrarNotificacionesPendientes() {
1640.                 String sql = "SELECT id, mensaje FROM notificaciones WHERE estado IN
('Pendiente', 'Recordar')";
1641.
1642.                 try (Connection conn = ConexionBD.getConnection();
1643.                      PreparedStatement pstmt = conn.prepareStatement(sql);
1644.                      ResultSet rs = pstmt.executeQuery()) {
1645.
1646.                     while (rs.next()) {
1647.                         int idNotificacion = rs.getInt("id");
1648.                         String mensaje = rs.getString("mensaje");
1649.                         mostrarNotificacionEmergente(mensaje, idNotificacion);
1650.                     }
1651.                 } catch (SQLException e) {
1652.                     e.printStackTrace();
1653.                 }

```

```

1654.         }
1655.
1656.         private void cerrarSession() {
1657.             this.dispose();
1658.
1659.             SwingUtilities.invokeLater(() -> {
1660.                 LoginFrame loginFrame = new LoginFrame();
1661.                 loginFrame.setVisible(true);
1662.             });
1663.         }
1664.     }

```

Paquete: vista; Clase: NotificacionFrame.java

```

1. package vista;
2.
3. import javax.swing.*;
4. import java.awt.*;
5. import java.util.ArrayList;
6. import java.util.List;
7.
8. public class NotificacionFrame {
9.     private final JFrame ventanaPrincipal;
10.    private final List<JDialog> notificacionesActivas = new ArrayList<>();
11.    private final int anchoNotificacion = 350;
12.    private final int altoNotificacion = 100;
13.
14.    public NotificacionFrame(JFrame ventanaPrincipal) {
15.        this.ventanaPrincipal = ventanaPrincipal;
16.    }
17.
18.    public void mostrarNotificacion(String titulo, String mensaje, int
idNotificacion, Runnable accionListo, Runnable accionRecordar) {
19.        JDialog notificacion = new JDialog(ventanaPrincipal, false);
20.        notificacion.setUndecorated(true);
21.        notificacion.setSize(anchoNotificacion, altoNotificacion + 20);
22.
23.        JPanel panelContenido = new JPanel();
24.        panelContenido.setLayout(new BorderLayout());
25.        panelContenido.setBackground(new Color(245, 245, 245));
26.
27.        panelContenido.setBorder(BorderFactory.createCompoundBorder(
28.            BorderFactory.createLineBorder(new Color(200, 200, 200), 1),
29.            BorderFactory.createMatteBorder(5, 5, 5, 5, new Color(245, 245, 245))
30.        ));
31.
32.        JLabel labelTitulo = new JLabel(titulo);
33.        labelTitulo.setFont(new Font("Arial", Font.BOLD, 14));
34.        labelTitulo.setForeground(new Color(50, 50, 50));
35.        labelTitulo.setBorderStyle(BorderFactory.createEmptyBorder(5, 10, 0, 10));
36.
37.        JTextArea textAreaMensaje = new JTextArea(mensaje);
38.        textAreaMensaje.setFont(new Font("Arial", Font.PLAIN, 12));
39.        textAreaMensaje.setForeground(new Color(50, 50, 50));
40.        textAreaMensaje.setWrapStyleWord(true);
41.        textAreaMensaje.setLineWrap(true);
42.        textAreaMensaje.setOpaque(false);
43.        textAreaMensaje.setEditable(false);
44.        textAreaMensaje.setBorderStyle(BorderFactory.createEmptyBorder(5, 10, 10, 10));
45.
46.        JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 5));
47.        panelBotones.setBackground(new Color(245, 245, 245));
48.
49.        JButton btnRecordar = new JButton("Recordar");

```

```

50.         JButton btnListo = new JButton("Listo");
51.
52.         estilizarBoton(btnRecordar, new Color(240, 240, 240), new Color(100, 100,
53.             100));
54.         estilizarBoton(btnListo, new Color(33, 150, 243), Color.WHITE);
55.         btnRecordar.addActionListener(e -> {
56.             accionRecordar.run();
57.             desvanecerNotificacion(notificacion);
58.         });
59.
60.         btnListo.addActionListener(e -> {
61.             accionListo.run();
62.             desvanecerNotificacion(notificacion);
63.         });
64.
65.         panelBotones.add(btnRecordar);
66.         panelBotones.add(btnListo);
67.
68.         panelContenido.add(labelTitulo, BorderLayout.NORTH);
69.         panelContenido.add(textAreaMensaje, BorderLayout.CENTER);
70.         panelContenido.add(panelBotones, BorderLayout.SOUTH);
71.
72.         notificacion.add(panelContenido);
73.
74.         int x = ventanaPrincipal.getWidth() - anchoNotificacion - 30;
75.         int separacionVertical = 30;
76.         int margenSuperior = 175;
77.         int y = ventanaPrincipal.getHeight() - margenSuperior -
78.             (notificacionesActivas.size() * (altoNotificacion + separacionVertical));
79.         notificacion.setLocation(x, y);
80.
81.         notificacionesActivas.add(notificacion);
82.
83.         notificacion.setVisible(true);
84.
85.         Timer temporizador = new Timer(8000, e ->
86.             desvanecerNotificacion(notificacion));
87.         temporizador.setRepeats(false);
88.         temporizador.start();
89.     }
90.
91.     private void estilizarBoton(JButton boton, Color fondo, Color texto) {
92.         boton.setBackground(fondo);
93.         boton.setForeground(texto);
94.         boton.setFocusPainted(false);
95.         boton.setFont(new Font("Arial", Font.PLAIN, 12));
96.         boton.setBorder(BorderFactory.createCompoundBorder(
97.             BorderFactory.createLineBorder(new Color(200, 200, 200), 1),
98.             BorderFactory.createEmptyBorder(5, 15, 5, 15)
99.         ));
100.    }
101.
102.    private void desvanecerNotificacion(JDialog notificacion) {
103.        new Thread(() -> {
104.            try {
105.                for (float i = 1.0f; i > 0; i -= 0.1f) {
106.                    Thread.sleep(50);
107.                    notificacion.setOpacity(i);
108.                }
109.            } catch (InterruptedException ignored) {}
110.            SwingUtilities.invokeLater(() -> {
111.                notificacion.dispose();
112.                notificacionesActivas.remove(notificacion);
113.                actualizarPosicionesNotificaciones();
114.            });
115.        });

```

```

113.             }).start();
114.         }
115.
116.         private void actualizarPosicionesNotificaciones() {
117.             int y = ventanaPrincipal.getHeight() - 175;
118.             for (JDialog notificacion : notificacionesActivas) {
119.                 notificacion.setLocation(notificacion.getX(), y);
120.                 y -= (altoNotificacion + 10);
121.             }
122.         }
123.     }

```

Paquete: vista; Clase: RenovacionFrame.java

```

1. package vista;
2.
3. import modelo.ActivoFijo;
4. import servicio.ActivoServicio;
5. import util.VentanaUtils;
6.
7. import javax.swing.*;
8. import javax.swing.table.DefaultTableModel;
9. import java.awt.*;
10. import java.util.ArrayList;
11. import java.util.List;
12.
13. public class RenovacionFrame extends JFrame {
14.     private final MainFrame mainFrame;
15.     private final ActivoServicio activoServicio;
16.     private DefaultTableModel tableModel;
17.     private JTable activosTable;
18.     private JComboBox<String> cmbTipoActivo;
19.     private JTextField txtBuscarActivo;
20.     private JLabel lblTotalRenovados;
21.     private List<Object[]> estadoInicial = new ArrayList<>();
22.
23.     public RenovacionFrame(ActivoServicio servicio, MainFrame mainFrame) {
24.         this.activoServicio = servicio;
25.         this.mainFrame = mainFrame;
26.         initUI();
27.     }
28.
29.     private void initUI() {
30.         setTitle("Renovación de Activos");
31.         setSize(1000, 600);
32.         setLocationRelativeTo(null);
33.         VentanaUtils.deshabilitarMinimizar(this);
34.         setResizable(false);
35.         setLayout(new BorderLayout(10, 10));
36.
37.         // Panel superior
38.         JPanel filterPanel = new JPanel(new GridBagLayout());
39.         filterPanel.setBorder(BorderFactory.createEmptyBorder(15, 10, 15, 10));
40.         GridBagConstraints gbc = new GridBagConstraints();
41.         gbc.insets = new Insets(5, 5, 5, 5);
42.         gbc.fill = GridBagConstraints.HORIZONTAL;
43.
44.         JLabel lblTipoActivo = new JLabel("Tipo de Activo:");
45.         cmbTipoActivo = new JComboBox<>(new String[]{"Todos", "Equipo de Computo",
46.             "Mobiliario",
47.             "Equipo de Laboratorio", "Vehiculo", "Terreno"});
48.         txtBuscarActivo = new JTextField(20);
49.         txtBuscarActivo.setPreferredSize(new Dimension(200, 25));

```

```

50.         JButton btnReload = new JButton("X");
51.         btnReload.setPreferredSize(new Dimension(30, 25));
52.         btnReload.setBorder(BorderFactory.createLineBorder(Color.GRAY));
53.         btnReload.setFocusPainted(false);
54.         btnReload.setContentAreaFilled(false);
55.         btnReload.setCursor(new Cursor(Cursor.HAND_CURSOR));
56.
57.         JButton btnBuscar = new JButton("Buscar");
58.
59.         JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
60.         searchPanel.add(txtBuscarActivo);
61.         searchPanel.add(btnReload);
62.
63.         gbc.gridx = 0;
64.         gbc.gridy = 0;
65.         filterPanel.add(lblTipoActivo, gbc);
66.
67.         gbc.gridx = 1;
68.         gbc.gridy = 0;
69.         filterPanel.add(cmbTipoActivo, gbc);
70.
71.         gbc.gridx = 2;
72.         gbc.gridy = 0;
73.         filterPanel.add(searchPanel, gbc);
74.
75.         gbc.gridx = 3;
76.         gbc.gridy = 0;
77.         filterPanel.add(btnBuscar, gbc);
78.
79.         // Tabla de activos
80.         String[] columnNames = {
81.             "ID", "Nombre", "Categoría", "Estado", "Fecha Adquisición", "Costo
Inicial"
82.         };
83.         tableModel = new DefaultTableModel(columnNames, 0) {
84.             @Override
85.             public boolean isCellEditable(int row, int column) {
86.                 return false; // Ninguna celda editable
87.             }
88.         };
89.         activosTable = new JTable(tableModel);
90.         activosTable.getTableHeader().setReorderingAllowed(false);
91.
92.         JScrollPane tableScrollPane = new JScrollPane(activosTable);
93.
94.         // Panel inferior
95.         JPanel buttonPanel = new JPanel(new BorderLayout());
96.
97.         // Mensaje
98.         lblTotalRenovados = new JLabel("Total de activos renovados: 0");
99.         lblTotalRenovados.setBorder(BorderFactory.createEmptyBorder(5, 10, 5,
10));
100.
101.         // Botones
102.         JPanel botonesPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
103.         JButton btnGuardar = new JButton("Renovar");
104.         JButton btnCancelar = new JButton("Cancelar");
105.
106.         botonesPanel.add(btnGuardar);
107.         botonesPanel.add(btnCancelar);
108.
109.         buttonPanel.add(lblTotalRenovados, BorderLayout.WEST);
110.         buttonPanel.add(botonesPanel, BorderLayout.EAST);
111.
112.

```

```

113.         cmbTipoActivo.addActionListener(e ->
114.             filtrarPorTipoActivo(cmbTipoActivo.getSelectedItem().toString()));
115.         btnBuscar.addActionListener(e ->
116.             buscarActivo(txtBuscarActivo.getText()));
117.         btnReload.addActionListener(e -> {
118.             txtBuscarActivo.setText("");
119.             filtrarPorTipoActivo(cmbTipoActivo.getSelectedItem().toString());
120.         });
121.         btnGuardar.addActionListener(e -> renovarActivoSeleccionado());
122.         btnCancelar.addActionListener(e -> dispose());
123.         add(filterPanel, BorderLayout.NORTH);
124.         add(tableScrollPane, BorderLayout.CENTER);
125.         add(buttonPanel, BorderLayout.SOUTH);
126.         setDefaultCloseOperation(DISPOSE_ON_CLOSE);
127.         cargarDatosDesdeBD();
128.         actualizarTotalRenovados();
129.     }
130.
131.     private void actualizarTotalRenovados() {
132.         int totalRenovados =
133.             activoServicio.contarActivosPorEstado("Renovado");
134.         lblTotalRenovados.setText("Total de activos renovados: " +
135.             totalRenovados);
136.     }
137.
138.     private void cargarDatosDesdeBD() {
139.         tableModel.setRowCount(0);
140.         estadoInicial.clear();
141.
142.         List<ActivoFijo> activos = activoServicio.cargarActivosDesdeBD();
143.         for (ActivoFijo activo : activos) {
144.             if (activo.getEstado().equalsIgnoreCase("Depreciado") ||
145.                 activo.getEstado().equalsIgnoreCase("Baja") ||
146.                 activo.getEstado().equalsIgnoreCase("Baja por venta")) {
147.
148.                 Object[] row = {
149.                     activo.getNumero(),
150.                     activo.getNombre(),
151.                     activo.getCategoría(),
152.                     activo.getEstado(),
153.                     activo.getFechaAdquisicion(),
154.                     activo.getCostoInicial()
155.                 };
156.                 tableModel.addRow(row);
157.                 estadoInicial.addRow(row);
158.             }
159.         }
160.
161.         SwingUtilities.invokeLater(() -> {
162.             activosTable.revalidate();
163.             activosTable.repaint();
164.         });
165.     }
166.
167.     private void filtrarPorTipoActivo(String tipoActivo) {
168.         tableModel.setRowCount(0);
169.
170.         for (Object[] row : estadoInicial) {
171.             boolean tipoCoincide = tipoActivo.equals("Todos") ||
172.             row[2].equals(tipoActivo);
173.             if (tipoCoincide) {
174.                 tableModel.addRow(row);

```

```

174.             }
175.         }
176.
177.         SwingUtilities.invokeLater(() -> {
178.             activosTable.revalidate();
179.             activosTable.repaint();
180.         });
181.     }
182.
183.     private void buscarActivo(String query) {
184.         tableModel.setRowCount(0);
185.
186.         for (Object[] row : estadoInicial) {
187.             boolean coincideBusqueda =
188.                 row[1].toString().toLowerCase().contains(query.toLowerCase()) ||
189.                     row[0].toString().equals(query);
190.
191.             if (coincideBusqueda) {
192.                 tableModel.addRow(row);
193.             }
194.
195.         SwingUtilities.invokeLater(() -> {
196.             activosTable.revalidate();
197.             activosTable.repaint();
198.         });
199.     }
200.
201.     private void renovarActivoSeleccionado() {
202.         int selectedRow = activosTable.getSelectedRow();
203.         if (selectedRow == -1) {
204.             JOptionPane.showMessageDialog(this, "Seleccione un activo para
renovar.", "Advertencia", JOptionPane.WARNING_MESSAGE);
205.             return;
206.         }
207.
208.         int idActivo = Integer.parseInt(tableModel.getValueAt(selectedRow,
0).toString());
209.         String nombreActivo = tableModel.getValueAt(selectedRow,
1).toString();
210.         String motivo = null;
211.         boolean motivoValido = false;
212.
213.         while (!motivoValido) {
214.             motivo = JOptionPane.showInputDialog(this, "Ingrese el motivo de
la renovación:", "Motivo de Renovación", JOptionPane.PLAIN_MESSAGE);
215.
216.             if (motivo == null) {
217.                 return;
218.             }
219.
220.             if (!motivo.trim().isEmpty()) {
221.                 motivoValido = true;
222.             } else {
223.                 JOptionPane.showMessageDialog(this, "Debe ingresar un motivo
para renovar el activo.", "Advertencia", JOptionPane.WARNING_MESSAGE);
224.             }
225.         }
226.
227.         try {
228.             activoServicio.renovarActivo(idActivo, motivo);
229.             String mensajeRenovacion = String.format("Activo '%s' (ID: %d) ha
sido renovado con éxito.",
230.                                         nombreActivo, idActivo);
231.             JOptionPane.showMessageDialog(this, "Activo renovado con éxito.",
"Éxito", JOptionPane.INFORMATION_MESSAGE);

```

```

232.                     mainFrame.mostrarMensaje(mensajeRenovacion);
233.                     mainFrame.recargarDatos();
234.                     cargarDatosDesdeBD();
235.                     actualizarTotalRenovados();
236.                 } catch (Exception e) {
237.                     JOptionPane.showMessageDialog(this, "Error al renovar el activo: "
+ e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
238.                 }
239.             }
240.         }

```

Paquete: vista; Clase: RevaluacionFrame.java

```

1. package vista;
2.
3. import servicio.ActivoServicio;
4. import servicio.RevaluacionServicio;
5. import util.ImpactoTotalCellRenderer;
6. import util.VentanaUtils;
7.
8. import javax.swing.*;
9. import javax.swing.table.DefaultTableModel;
10. import java.awt.*;
11. import java.util.ArrayList;
12. import java.util.HashSet;
13. import java.util.List;
14. import java.util.Set;
15.
16. public class RevaluacionFrame extends JFrame {
17.     private final MainFrame mainFrame;
18.     private final ActivoServicio activoServicio;
19.     private DefaultTableModel tableModel;
20.     private final Set<Integer> filasBloqueadas = new HashSet<>();
21.     private boolean cambiosRealizados = false;
22.     private JTable activosTable;
23.     private JComboBox<String> cmbTipoActivo;
24.     private JTextField txtBuscarActivo;
25.     private List<Object[]> estadoInicial = new ArrayList<>();
26.
27.     public RevaluacionFrame(ActivoServicio servicio, MainFrame mainFrame) {
28.         this.activoServicio = servicio;
29.         this.mainFrame = mainFrame;
30.         initUI();
31.     }
32.
33.     private void initUI() {
34.         setTitle("Revaluación de Activos");
35.         setSize(1000, 600);
36.         setLocationRelativeTo(null);
37.         VentanaUtils.deshabilitarMinimizar(this);
38.         setResizable(false);
39.         setLayout(new BorderLayout(10, 10));
40.
41.         // Panel superior
42.         JPanel filterPanel = new JPanel(new GridBagLayout());
43.         filterPanel.setBorder(BorderFactory.createEmptyBorder(15, 10, 15, 10)); // /
Márgenes adicionales
44.         GridBagConstraints gbc = new GridBagConstraints();
45.         gbc.insets = new Insets(5, 5, 5, 5);
46.         gbc.fill = GridBagConstraints.HORIZONTAL;
47.
48.         JLabel lblTipoActivo = new JLabel("Tipo de Activo:");
49.         cmbTipoActivo = new JComboBox<>(new String[]{"Todos", "Equipo de Computo",
"Mobiliario",

```

```

50.             "Equipo de Laboratorio", "Vehiculo", "Terreno"));
51.
52.         // Campo de búsqueda
53.         txtBuscarActivo = new JTextField(20);
54.         txtBuscarActivo.setPreferredSize(new Dimension(200, 25));
55.
56.         JButton btnReload = new JButton("X");
57.         btnReload.setPreferredSize(new Dimension(30, 25));
58.         btnReload.setBorder(BorderFactory.createLineBorder(Color.GRAY));
59.         btnReload.setFocusPainted(false);
60.         btnReload.setContentAreaFilled(false);
61.         btnReload.setCursor(new Cursor(Cursor.HAND_CURSOR));
62.
63.         JButton btnBuscar = new JButton("Buscar");
64.
65.         JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
66.         searchPanel.add(txtBuscarActivo);
67.         searchPanel.add(btnReload);
68.
69.         gbc.gridx = 0;
70.         gbc.gridy = 0;
71.         filterPanel.add(lblTipoActivo, gbc);
72.
73.         gbc.gridx = 1;
74.         gbc.gridy = 0;
75.         filterPanel.add(cmbTipoActivo, gbc);
76.
77.         gbc.gridx = 2;
78.         gbc.gridy = 0;
79.         filterPanel.add(searchPanel, gbc);
80.
81.         gbc.gridx = 3;
82.         gbc.gridy = 0;
83.         filterPanel.add(btnBuscar, gbc);
84.
85.         // Tabla de activos
86.         String[] columnNames = {
87.             "ID", "Nombre", "Categoría", "Valor Anterior", "Valor Revaluado",
88.             "Depreciación Anterior", "Nueva Depreciación", "Cambio Absoluto",
89.             "Impacto Total"
90.         };
91.         tableModel = new DefaultTableModel(columnNames, 0) {
92.             @Override
93.             public boolean isCellEditable(int row, int column) {
94.                 return column == 4 && !filasBloqueadas.contains(row);
95.             }
96.         };
97.         activosTable = new JTable(tableModel);
98.         activosTable.getTableHeader().setReorderingAllowed(false);
99.         activosTable.getColumnModel().getColumn(8).setCellRenderer(new
    ImpactoTotalCellRenderer());
100.
101.        JScrollPane tableScrollPane = new JScrollPane(activosTable);
102.
103.        // Panel inferior
104.        JPanel buttonPanel = new JPanel(new BorderLayout());
105.
106.        // Mensaje guía
107.        JLabel lblGuia = new JLabel("Seleccione (con doble click) un campo en
    la columna 'Valor Revaluado' para revaluar.");
108.        lblGuia.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));
109.        lblGuia.setFont(new Font("SansSerif", Font.PLAIN, 12));
110.
111.        // Panel de botones
112.        JPanel botonesPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

```

```

113.         JButton btnCancelar = new JButton("Cancelar");
114.         botonesPanel.add(btnGuardar);
115.         botonesPanel.add(btnCancelar);
116.
117.         buttonPanel.add(lblGuia, BorderLayout.WEST);
118.         buttonPanel.add(botonesPanel, BorderLayout.EAST);
119.
120.         cmbTipoActivo.addActionListener(e ->
121.             filtrarPorTipoActivo(cmbTipoActivo.getSelectedItem().toString()));
122.             btnBuscar.addActionListener(e ->
123.                 buscarActivo(txtBuscarActivo.getText()));
124.                 btnReload.addActionListener(e -> {
125.                     txtBuscarActivo.setText("");
126.                     String categoriaSeleccionada =
127.                         cmbTipoActivo.getSelectedItem().toString();
128.                         filtrarPorTipoActivo(categoriaSeleccionada);
129.                         });
130.
131.             btnGuardar.addActionListener(e -> guardarRevaluaciones(activosTable));
132.             btnCancelar.addActionListener(e -> {
133.                 tableModel.setRowCount(0);
134.                 filasBloqueadas.clear();
135.                 dispose();
136.             });
137.
138.             add(filterPanel, BorderLayout.NORTH);
139.             add(tableScrollPane, BorderLayout.CENTER);
140.             add(buttonPanel, BorderLayout.SOUTH);
141.
142.             setDefaultCloseOperation(DISPOSE_ON_CLOSE);
143.             cargarDatosDesdeBD();
144.             cambiosRealizados = false;
145.
146.     private void filtrarPorTipoActivo(String tipoActivo) {
147.         tableModel.setRowCount(0);
148.         filasBloqueadas.clear();
149.         RevaluacionServicio revaluacionServicio = new RevaluacionServicio();
150.         List<Object[]> datos = revaluacionServicio.cargarDatosDesdeBD();
151.
152.         for (int i = 0; i < datos.size(); i++) {
153.             Object[] row = datos.get(i);
154.
155.             if (tipoActivo.equals("Todos") || row[2].equals(tipoActivo)) {
156.                 if (row[4] != null && !row[4].toString().trim().isEmpty()) {
157.                     try {
158.                         double valorAnterior = (double) row[3];
159.                         double valorRevaluado =
160.                             Double.parseDouble(row[4].toString());
161.                         row[7] = valorRevaluado - valorAnterior;
162.                         row[8] = String.format("%.2f%", new
163.                             RevaluacionServicio().calcularImpactoPorcentual(valorAnterior, valorRevaluado));
164.                         filasBloqueadas.add(tableModel.getRowCount());
165.                     } catch (NumberFormatException | ClassCastException e) {
166.                         row[7] = "";
167.                         row[8] = "";
168.                     }
169.                 } else {
170.                     row[7] = "";
171.                     row[8] = "";
172.                 }
173.             }

```

```

174.             tableModel.addRow(row);
175.         }
176.     }
177.
178.     SwingUtilities.invokeLater(() -> {
179.         activosTable.revalidate();
180.         activosTable.repaint();
181.     });
182. }
183.
184. private void buscarActivo(String query) {
185.     tableModel.setRowCount(0);
186.     filasBloqueadas.clear();
187.
188.     RevaluacionServicio revaluacionServicio = new RevaluacionServicio();
189.     List<Object[]> datos = revaluacionServicio.cargarDatosDesdeBD();
190.
191.     String categoriaSeleccionada = ((JComboBox<String>) ((JPanel)
192.         getContentPane().getComponent(0))
193.             .getComponent(1)).getSelectedItem().toString();
194.
195.     for (int i = 0; i < datos.size(); i++) {
196.         Object[] row = datos.get(i);
197.
198.         boolean coincideCategoria = categoriaSeleccionada.equals("Todos")
199.             || row[2].equals(categoriaSeleccionada);
200.
201.         boolean coincideBusqueda =
202.             row[1].toString().toLowerCase().contains(query.toLowerCase()) ||
203.             row[0].toString().equals(query);
204.
205.         if (coincideCategoria && coincideBusqueda) {
206.             if (row[4] != null && !row[4].toString().trim().isEmpty()) {
207.                 try {
208.                     double valorAnterior = (double) row[3];
209.                     double valorRevaluado =
210.                         Double.parseDouble(row[4].toString());
211.
212.                     row[7] = valorRevaluado - valorAnterior;
213.                     row[8] = String.format("%.2f%", new
214.                         RevaluacionServicio().calcularImpactoPorcentual(valorAnterior, valorRevaluado));
215.
216.                     filasBloqueadas.add(tableModel.getRowCount());
217.                 } catch (NumberFormatException | ClassCastException e) {
218.                     row[7] = "";
219.                     row[8] = "";
220.                 }
221.             }
222.         }
223.
224.         SwingUtilities.invokeLater(() -> {
225.             activosTable.revalidate();
226.             activosTable.repaint();
227.         });
228.     }
229.
230.     private void guardarRevaluaciones(JTable activosTable) {
231.         try {
232.             boolean cambiosGuardados = false;
233.             List<String> mensajesRevaluacion = new ArrayList<>();

```

```

234.
235.            for (int i = 0; i < tableModel.getRowCount(); i++) {
236.                if (filasBloqueadas.contains(i)) {
237.                    continue;
238.                }
239.
240.                String valorRevaluadoStr = tableModel.getValueAt(i, 4) != null
241.                    ? tableModel.getValueAt(i, 4).toString().trim()
242.                    : "";
243.
244.                if (valorRevaluadoStr.isEmpty()) {
245.                    continue;
246.                }
247.
248.                double valorRevaluado;
249.                try {
250.                    valorRevaluado = Double.parseDouble(valorRevaluadoStr);
251.
252.                    if (valorRevaluado <= 0) {
253.                        throw new IllegalArgumentException("El valor revaluado
en la fila " + (i + 1) + " debe ser mayor a 0.");
254.                    }
255.                } catch (NumberFormatException e) {
256.                    JOptionPane.showMessageDialog(this, "El valor revaluado en
la fila " + (i + 1) + " debe ser un número válido.", "Error",
JOptionPane.ERROR_MESSAGE);
257.                    return;
258.                }
259.
260.                double valorAnterior = (double) tableModel.getValueAt(i, 3);
261.
262.                if (valorRevaluado <= valorAnterior) {
263.                    JOptionPane.showMessageDialog(this, "El valor revaluado en
la fila " + (i + 1) + " debe ser mayor al valor anterior.", "Advertencia",
JOptionPane.WARNING_MESSAGE);
264.                    return;
265.                }
266.
267.                int idActivo = Integer.parseInt(tableModel.getValueAt(i,
0).toString());
268.                String nombreActivo = tableModel.getValueAt(i, 1).toString();
269.
270.                mensajesRevaluacion.add("Activo '" + nombreActivo + "' (ID: "
+ idActivo + ") revaluado de " +
String.format("%.2f", valorAnterior) + " a " +
String.format("%.2f", valorRevaluado) + ".");
271.
272.
273.                cambiosGuardados = true;
274.            }
275.
276.
277.            if (!cambiosGuardados) {
278.                JOptionPane.showMessageDialog(this, "No se ha revaluado ningún
activo. Por favor, realice cambios antes de guardar.", "Advertencia",
JOptionPane.WARNING_MESSAGE);
279.                return;
280.            }
281.
282.            String motivoGeneral = null;
283.            boolean motivoValido = false;
284.            while (!motivoValido) {
285.                motivoGeneral = JOptionPane.showInputDialog(this, "Ingrese el
motivo para esta revaluación:", "Motivo de Revaluación", JOptionPane.PLAIN_MESSAGE);
286.
287.                if (motivoGeneral == null) {
288.                    return;
289.                }

```

```

290.                     if (!motivoGeneral.trim().isEmpty()) {
291.                         motivoValido = true;
292.                     } else {
293.                         JOptionPane.showMessageDialog(this, "Debe ingresar un
294. motivo para guardar los cambios.", "Advertencia", JOptionPane.WARNING_MESSAGE);
295.                     }
296.                 }
297.
298.                 for (int i = 0; i < tableView.getModel().getRowCount(); i++) {
299.                     if (filasBloqueadas.contains(i)) {
300.                         continue;
301.                     }
302.
303.                     String valorRevaluadoStr = tableView.getModel().getValueAt(i, 4) != null
304.                         ? tableView.getModel().getValueAt(i, 4).toString().trim()
305.                         : "";
306.
307.                     if (valorRevaluadoStr.isEmpty()) {
308.                         continue;
309.                     }
310.
311.                     double valorRevaluado = Double.parseDouble(valorRevaluadoStr);
312.                     double valorAnterior = (double) tableView.getModel().getValueAt(i, 3);
313.
314.                     int idActivo = Integer.parseInt(tableView.getModel().getValueAt(i,
315.                         0).toString());
315.                     new RevaluacionServicio().guardarRevaluacionEnBD(idActivo,
316.                         valorRevaluado, motivoGeneral);
316.
317.                     filasBloqueadas.add(i);
318.                     tableView.setValueAt(valorRevaluado, i, 3);
319.                     tableView.setValueAt("", i, 4);
320.                 }
321.
322.                 SwingUtilities.invokeLater(() -> {
323.                     activosTable.revalidate();
324.                     activosTable.repaint();
325.                 });
326.                 cmbTipoActivo.setSelectedIndex(0);
327.                 txtBuscarActivo.setText("");
328.                 filtrarPorTipoActivo("Todos");
329.
330.                 if (mainFrame != null) {
331.                     mainFrame.recargarDatos();
332.                 }
333.
334.                 JOptionPane.showMessageDialog(this, "Revaluaciones guardadas
335. exitosamente.");
335.                 for (String mensaje : mensajesRevaluacion) {
336.                     mainFrame.mostrarMensaje(mensaje);
337.                 }
338.             } catch (Exception ex) {
339.                 JOptionPane.showMessageDialog(this, "Error al guardar las
340. revaluaciones: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
340.             }
341.         }
342.
343.         private void cargarDatosDesdeBD() {
344.             RevaluacionServicio revaluacionServicio = new RevaluacionServicio();
345.
346.             tableView.setRowCount(0);
347.             filasBloqueadas.clear();
348.             estadoInicial.clear();
349.             cambiosRealizados = false;
350.

```

```

351.             List<Object[]> datos = revaluacionServicio.cargarDatosDesdeBD();
352.
353.         for (int i = 0; i < datos.size(); i++) {
354.             Object[] row = datos.get(i);
355.
356.             if (row[4] != null && !row[4].toString().trim().isEmpty()) {
357.                 try {
358.                     double valorAnterior = (double) row[3];
359.                     double valorRevaluado =
360.                         Double.parseDouble(row[4].toString());
361.                     row[7] = String.format("%.2f", valorRevaluado -
362.                         valorAnterior);
363.                     row[8] = String.format("%.2f%%", new
364.                         RevaluacionServicio().calcularImpactoPorcentual(valorAnterior, valorRevaluado));
365.                     filasBloqueadas.add(i);
366.                 } catch (NumberFormatException | ClassCastException e) {
367.                     row[7] = "";
368.                     row[8] = "";
369.                 } else {
370.                     row[7] = "";
371.                     row[8] = "";
372.                 }
373.
374.                 tableViewModel.addRow(row);
375.                 estadoInicial.add(new Object[]{row[3], row[4]});
376.             }
377.
378.             SwingUtilities.invokeLater(() -> {
379.                 activosTable.revalidate();
380.                 activosTable.repaint();
381.             });
382.         }
383.     }

```

Paquete: vista; Clase: UbicacionActivosFrame.java

```

1. package vista;
2.
3. import modelo.ActivoFijo;
4. import modelo.Sede;
5. import org.jfree.chart.ChartFactory;
6. import org.jfree.chart.ChartPanel;
7. import org.jfree.chart.JFreeChart;
8. import org.jfree.chart.labels.StandardPieSectionLabelGenerator;
9. import org.jfree.chart.plot.PiePlot;
10. import org.jfree.data.general.DefaultPieDataset;
11. import servicio.ActivoServicio;
12. import util.VentanaUtils;
13.
14. import javax.swing.*;
15. import javax.swing.border.EmptyBorder;
16. import java.awt.*;
17. import java.util.List;
18. import java.util.Map;
19. import java.util.stream.Collectors;
20.
21. public class UbicacionActivosFrame extends JFrame {
22.     private ActivoServicio activoServicio;
23.
24.     public UbicacionActivosFrame(ActivoServicio activoServicio) {
25.         this.activoServicio = activoServicio;

```

```

26.         setTitle("Distribución de Activos por Ubicación");
27.         setSize(1200, 700);
28.         setResizable(false);
29.         setLocationRelativeTo(null);
30.         VentanaUtils.deshabilitarMinimizar(this);
31.         setDefaultCloseOperation(DISPOSE_ON_CLOSE);
32.
33.         // Panel principal
34.         JPanel mainPanel = new JPanel(new BorderLayout());
35.         mainPanel.setBorder(new EmptyBorder(10, 10, 10, 10));
36.
37.         // Panel superior
38.         JPanel headerPanel = crearPanelContadores();
39.         mainPanel.add(headerPanel, BorderLayout.NORTH);
40.
41.         // Panel central
42.         JPanel centerPanel = new JPanel(new BorderLayout(10, 10));
43.
44.         // Panel de tarjetas
45.         JScrollPane cardsScrollPane = new JScrollPane(crearPanelTarjetas());
46.         cardsScrollPane.setPreferredSize(new Dimension(700, 0));
47.         centerPanel.add(cardsScrollPane, BorderLayout.CENTER);
48.
49.         // Panel del gráfico
50.         JPanel chartPanel = crearPanelGrafico();
51.         chartPanel.setPreferredSize(new Dimension(450, 0));
52.         centerPanel.add(chartPanel, BorderLayout.EAST);
53.
54.         mainPanel.add(centerPanel, BorderLayout.CENTER);
55.
56.         add(mainPanel);
57.         setVisible(true);
58.     }
59.
60.     private JPanel crearPanelContadores() {
61.         JPanel headerPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));
62.         headerPanel.setBackground(new Color(240, 240, 240));
63.
64.         Map<Sede, Long> activosPorSede = activoServicio.obtenerConteoPorSede();
65.
66.         if (activosPorSede.isEmpty()) {
67.             JLabel lblNoData = new JLabel("No hay datos de sedes disponibles");
68.             lblNoData.setFont(new Font("Arial", Font.BOLD, 16));
69.             headerPanel.add(lblNoData);
70.             return headerPanel;
71.         }
72.
73.         for (Map.Entry<Sede, Long> entry : activosPorSede.entrySet()) {
74.             JPanel counterPanel = new JPanel(new BorderLayout());
75.             counterPanel.setBackground(new Color(245, 245, 245));
76.             counterPanel.setBorder(BorderFactory.createLineBorder(new Color(200, 200,
77.                 200), 1));
77.             counterPanel.setPreferredSize(new Dimension(180, 80));
78.
79.             JLabel lblSede = new JLabel(entry.getKey().getNombre(),
80.                 SwingConstants.CENTER);
81.             lblSede.setForeground(new Color(54, 54, 54));
82.             lblSede.setFont(new Font("Arial", Font.BOLD, 14));
83.             lblSede.setBorder(BorderFactory.createEmptyBorder(5, 0, 10, 0));
84.
85.             JLabel lblConteo = new JLabel(String.valueOf(entry.getValue()),
86.                 SwingConstants.CENTER);
87.             lblConteo.setForeground(new Color(54, 162, 235));
88.             lblConteo.setFont(new Font("Arial", Font.BOLD, 36));
89.
90.             counterPanel.add(lblSede, BorderLayout.NORTH);

```

```

89.         counterPanel.add(lblConteo, BorderLayout.CENTER);
90.
91.         headerPanel.add(counterPanel);
92.     }
93.
94.     return headerPanel;
95. }
96.
97. private JPanel crearPanelTarjetas() {
98.     JPanel cardsPanel = new JPanel(new GridLayout(0, 2, 10, 10));
99.     cardsPanel.setBackground(Color.WHITE);
100.
101.     List<ActivoFijo> activos = activoServicio.cargarActivosDesdeBD();
102.
103.     Map<Sede, List<ActivoFijo>> activosPorSede = activos.stream()
104.         .filter(activo -> activo.getSede() != null)
105.         .collect(Collectors.groupingBy(ActivoFijo::getSede));
106.
107.     for (Map.Entry<Sede, List<ActivoFijo>> entry :
108.         activosPorSede.entrySet()) {
109.         List<ActivoFijo> activosEnSede = entry.getValue();
110.
111.         for (ActivoFijo activo : activosEnSede) {
112.             JPanel card = crearTarjetaActivo(activo);
113.             cardsPanel.add(card);
114.         }
115.
116.         JPanel containerPanel = new JPanel(new BorderLayout());
117.         containerPanel.setBackground(Color.WHITE);
118.
119.         JScrollPane scrollPane = new JScrollPane(cardsPanel);
120.         scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
121.         scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
122.         scrollPane.setBorder(BorderFactory.createEmptyBorder());
123.
124.         containerPanel.add(scrollPane, BorderLayout.CENTER);
125.
126.         return containerPanel;
127.     }
128.
129.     private JPanel crearTarjetaActivo(ActivoFijo activo) {
130.         JPanel cardPanel = new JPanel(new BorderLayout(10, 10));
131.         cardPanel.setBorder(BorderFactory.createCompoundBorder(
132.             BorderFactory.createLineBorder(new Color(200, 200, 200), 1),
133.             BorderFactory.createEmptyBorder(10, 10, 10, 10)
134.         ));
135.         cardPanel.setBackground(Color.WHITE);
136.         cardPanel.setPreferredSize(new Dimension(250, 150));
137.
138.         // Panel izquierdo
139.         JPanel iconPanel = new JPanel();
140.         iconPanel.setPreferredSize(new Dimension(80, 80));
141.         iconPanel.setBackground(new Color(255, 255, 255, 0));
142.         JLabel iconLabel = new JLabel();
143.         iconLabel.setHorizontalAlignment(SwingConstants.CENTER);
144.         iconLabel.setVerticalAlignment(SwingConstants.CENTER);
145.
146.         try {
147.             String iconPath = switch (activo.getCategoría().toLowerCase()) {
148.                 case "equipo de computo" -> "/computer_icon.png";
149.                 case "mobiliario" -> "/sofa_icon.png";
150.                 case "equipo de laboratorio" -> "/lab_icon.png";
151.                 case "vehiculo" -> "/car_icon.png";

```

```

152.             case "terreno" -> "/building_icon.png";
153.             default -> "/question_icon.png";
154.         };
155.
156.         ImageIcon originalIcon = new
157.             ImageIcon(getClass().getResource(iconPath));
158.             Image scaledImage = originalIcon.getImage().getScaledInstance(60,
159.                 60, Image.SCALE_SMOOTH);
160.             iconLabel.setIcon(new ImageIcon(scaledImage));
161.         } catch (NullPointerException e) {
162.             System.err.println("Icono no encontrado para categoría: " +
163.                 activo.getCategoría());
164.             iconLabel.setIcon(new
165.                 ImageIcon(getClass().getResource("/question_icon.png")));
166.             }
167.             iconPanel.add(iconLabel);
168.
169.             // Panel central
170.             JPanel infoPanel = new JPanel(new GridLayout(3, 1));
171.             infoPanel.setBackground(Color.WHITE);
172.
173.             JLabel lblNombre = new JLabel(activo.getNombre(),
174.                 SwingConstants.LEFT);
175.                 lblNombre.setFont(new Font("Arial", Font.BOLD, 14));
176.                 lblNombre.setForeground(new Color(54, 54, 54));
177.
178.             JLabel lblCodigo = new JLabel(activo.getCodigoPatrimonial(),
179.                 SwingConstants.LEFT);
180.                 lblCodigo.setFont(new Font("Arial", Font.PLAIN, 12));
181.                 lblCodigo.setForeground(new Color(120, 120, 120));
182.
183.             JLabel lblCategoria = new JLabel("Estado: " + activo.getEstado(),
184.                 SwingConstants.LEFT);
185.                 lblCategoria.setFont(new Font("Arial", Font.PLAIN, 12));
186.                 lblCategoria.setForeground(new Color(90, 90, 90));
187.
188.
189.             JPanel footerPanel = new JPanel(new BorderLayout());
190.             footerPanel.setBackground(Color.WHITE);
191.
192.             JLabel lblUbicacion = new JLabel("Ubicación: " +
193.                 activo.getSede().getNombre(), SwingConstants.LEFT);
194.                 lblUbicacion.setFont(new Font("Arial", Font.PLAIN, 12));
195.                 lblUbicacion.setForeground(new Color(150, 150, 150));
196.
197.             footerPanel.add(lblUbicacion, BorderLayout.WEST);
198.
199.             return cardPanel;
200.         }
201.
202.         private JPanel crearPanelGrafico() {
203.             JPanel chartContainer = new JPanel(new BorderLayout());
204.             chartContainer.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
205.                 10));
206.             Map<Sede, Long> activosPorSede =
207.                 activoServicio.obtenerConteoPorSede();

```

```

208.         DefaultPieDataset dataset = new DefaultPieDataset();
209.         for (Map.Entry<Sede, Long> entry : activosPorSede.entrySet()) {
210.             dataset.setValue(entry.getKey().getNombre(), entry.getValue());
211.         }
212.
213.         JFreeChart chart = ChartFactory.createPieChart(
214.             "Distribución de Activos por Ubicación",
215.             dataset,
216.             false,
217.             true,
218.             false
219.         );
220.
221.         PiePlot plot = (PiePlot) chart.getPlot();
222.         plot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}
223.             ({1})"));
224.         plot.setBackgroundPaint(new Color(0, 0, 0, 0));
225.
226.         Color[] colores = {
227.             new Color(169, 209, 142),
228.             new Color(255, 192, 0),
229.             new Color(112, 173, 71),
230.             new Color(237, 125, 49),
231.             new Color(91, 155, 213),
232.             new Color(193, 193, 193),
233.         };
234.
235.         int index = 0;
236.         for (Object key : dataset.getKeys()) {
237.             if (index < colores.length) {
238.                 plot.setSectionPaint(key.toString(), colores[index]);
239.             } else {
240.                 plot.setSectionPaint(key.toString(), Color.LIGHT_GRAY);
241.             }
242.             index++;
243.         }
244.         ChartPanel chartPanel = new ChartPanel(chart);
245.         chartContainer.add(chartPanel, BorderLayout.CENTER);
246.
247.         return chartContainer;
248.     }
249. }

```

Paquete: vista; Clase: VentaFrame.java

```

1. package vista;
2.
3. import servicio.ActivoServicio;
4. import util.ValidadorVentas;
5. import util.ConexionBD;
6. import util.VentanaUtils;
7.
8. import java.sql.*;
9.
10. import javax.swing.*;
11. import java.awt.*;
12. import java.awt.event.ActionEvent;
13. import java.time.LocalDate;
14. import java.time.format.DateTimeFormatter;
15. import javax.swing.event.DocumentEvent;
16. import javax.swing.event.DocumentListener;
17.
18. public class VentaFrame extends JFrame {

```

```

19.    private JTextField txtPrecioVenta;
20.    private JTextField txtComprador;
21.    private JTextField txtRUCComprador;
22.    private JTextField txtNumeroFactura;
23.    private JComboBox<String> cmbFormaPago;
24.    private JTextArea txtDetalleVenta;
25.    private JTextArea txtFacturaPreview;
26.    JButton btnVender, btnSalir;
27.
28.    private String nombre;
29.    private String categoria;
30.    private String fechaAdquisicion;
31.    private double costoInicial;
32.    private double valorResidual;
33.    private static int contadorFacturas = 1;
34.
35.    private MainFrame mainFrame;
36.    private int idActivo;
37.    private ActivoServicio activoServicio;
38.
39.    public VentaFrame(MainFrame mainFrame, int idActivo, String nombre, String
40.                      categoria, String fechaAdquisicion, double costoInicial, double valorResidual) {
41.        this.mainFrame = mainFrame;
42.        this.idActivo = idActivo;
43.        this.nombre = nombre;
44.        this.categoria = categoria;
45.        this.fechaAdquisicion = fechaAdquisicion;
46.        this.costoInicial = costoInicial;
47.        this.valorResidual = valorResidual;
48.        this.activoServicio = new ActivoServicio();
49.
50.        initComponents();
51.
52.        int numeroFactura = obtenerNumeroFactura();
53.        txtNumeroFactura.setText(LocalDate.now().format(DateTimeFormatter.ofPattern(
54.            "ddMMyyyy")) + "-" + String.format("%04d", numeroFactura));
55.
56.        agregarListenersParaPrevisualizacion();
57.        actualizarFacturaPreview();
58.        setVisible(true);
59.    }
60.
61.    private void initComponents() {
62.        setTitle("Formulario de Venta");
63.        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
64.        setLayout(new BorderLayout(10, 10));
65.        setResizable(false);
66.        VentanaUtils.deshabilitarMinimizar(this);
67.
68.        // Panel de entrada de datos
69.        JPanel panelDatos = new JPanel(new GridLayout(0, 2, 5, 5));
70.        panelDatos.setBorder(BorderFactory.createTitledBorder("Detalles de Venta"));
71.
72.        panelDatos.add(new JLabel("Número de Factura:"));
73.        panelDatos.add(txtNumeroFactura);
74.
75.        panelDatos.add(new JLabel("Activo:"));
76.        panelDatos.add(new JLabel(nombre));
77.
78.        panelDatos.add(new JLabel("Categoría:"));
79.        panelDatos.add(new JLabel(categoria));
80.
81.        panelDatos.add(new JLabel("Fecha de Adquisición:"));
82.

```

```

83.         panelDatos.add(new JLabel(fechaAdquisicion));
84.
85.         panelDatos.add(new JLabel("Valor Residual:"));
86.         panelDatos.add(new JLabel("S/ " + String.format("%.2f", valorResidual)));
87.
88.         panelDatos.add(new JLabel("Precio de Venta:"));
89.         txtPrecioVenta = new JTextField(String.format("%.2f", valorResidual));
90.         txtPrecioVenta.addActionListener(e -> btnVender.doClick());
91.         panelDatos.add(txtPrecioVenta);
92.
93.         panelDatos.add(new JLabel("Comprador:"));
94.         txtComprador = new JTextField();
95.         txtComprador.addActionListener(e -> btnVender.doClick());
96.         panelDatos.add(txtComprador);
97.
98.         panelDatos.add(new JLabel("RUC Comprador:"));
99.         txtRUCComprador = new JTextField();
100.        txtRUCComprador.addActionListener(e -> btnVender.doClick());
101.        panelDatos.add(txtRUCComprador);
102.
103.        panelDatos.add(new JLabel("Forma de Pago:"));
104.        cmbFormaPago = new JComboBox<>(new String[]{"Efectivo", "Tarjeta de Crédito", "Transferencia Bancaria", "Cheque"});
105.        panelDatos.add(cmbFormaPago);
106.
107.        panelDatos.add(new JLabel("Detalles de la Venta:"));
108.        txtDetalleVenta = new JTextArea(5, 20);
109.        txtDetalleVenta.setLineWrap(true);
110.        txtDetalleVenta.setWrapStyleWord(true);
111.        txtDetalleVenta.addKeyListener(new java.awt.event.KeyAdapter() {
112.            @Override
113.            public void keyPressed(java.awt.event.KeyEvent e) {
114.                if (e.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER) {
115.                    e.consume();
116.                    btnVender.doClick();
117.                }
118.            }
119.        });
120.        panelDatos.add(new JScrollPane(txtDetalleVenta));
121.
122.        add(panelDatos, BorderLayout.WEST);
123.
124.        // Panel de vista previa de factura
125.        txtFacturaPreview = new JTextArea();
126.        txtFacturaPreview.setEditable(false);
127.        txtFacturaPreview.setLineWrap(true);
128.        txtFacturaPreview.setWrapStyleWord(true);
129.        txtFacturaPreview.setBorder(BorderFactory.createTitledBorder("Vista Previa de Factura"));
130.        txtFacturaPreview.setFont(new Font("Monospaced", Font.PLAIN, 12));
131.        txtFacturaPreview.setPreferredSize(new Dimension(400, 500));
132.
133.        JScrollPane scrollFacturaPreview = new JScrollPane(txtFacturaPreview);
134.        scrollFacturaPreview.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
135.        scrollFacturaPreview.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
136.        add(scrollFacturaPreview, BorderLayout.CENTER);
137.
138.        // Panel de botones
139.        JPanel panelBotones = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
140.        btnVender = new JButton("Vender");
141.        btnSalir = new JButton("Salir");
142.
143.        panelBotones.add(btnVender);

```

```

144.         panelBotones.add(btnSalir);
145.         add(panelBotones, BorderLayout.SOUTH);
146.         btnVender.addActionListener(this::venderActivo);
147.         btnSalir.addActionListener(e -> dispose());
148.         setSize(850, 550);
149.         setLocationRelativeTo(null);
150.     }
151.
152.     private void venderActivo(ActionEvent e) {
153.         if (!ValidadorVentas.validarFormularioVenta(txtComprador,
154.             txtRUCComprador, txtDetalleVenta, txtPrecioVenta, valorResidual)) {
155.             return;
156.         }
157.
158.         int confirmacion = JOptionPane.showConfirmDialog(this, "¿Está seguro
159. de que desea vender este activo?", "Confirmar Venta", JOptionPane.YES_NO_OPTION);
160.         if (confirmacion == JOptionPane.NO_OPTION) {
161.             return;
162.         }
163.     }
164.
165.     try {
166.         activoServicio.actualizarEstadoActivo(idActivo, "Baja por venta");
167.
168.         int indiceActivo = mainFrame.obtenerIndicePorNumero(idActivo);
169.         if (indiceActivo != -1) {
170.             mainFrame.getListaBaseDatos().get(indiceActivo).setEstado("Baj
a por venta");
171.         }
172.
173.         double precioVenta = Double.parseDouble(txtPrecioVenta.getText());
174.         String detalleVenta = txtDetalleVenta.getText();
175.         String comprador = txtComprador.getText();
176.         String rucComprador = txtRUCComprador.getText();
177.         String formaPago = cmbFormaPago.getSelectedItem().toString();
178.         activoServicio.registrarVenta(idActivo, comprador, rucComprador,
179.             precioVenta, detalleVenta, formaPago);
180.         contadorFacturas++;
181.
182.         mainFrame.actualizarEstadoActivoEnTabla(idActivo, "Baja por
183.         venta");
184.         mainFrame.mostrarMensaje("Activo vendido exitosamente por S/ " +
185.             String.format("%.2f", precioVenta));
186.         mainFrame.getFiltroFrame().resetearFiltros(mainFrame);
187.         mainFrame.txtBuscar.setText("");
188.         mainFrame.actualizarVistaDeTotales();
189.         dispose();
190.
191.     } catch (NumberFormatException ex) {
192.         JOptionPane.showMessageDialog(this, "Error en el precio de venta.
193.         Por favor, ingrese un valor numérico.", "Error", JOptionPane.ERROR_MESSAGE);
194.     } catch (Exception ex) {
195.         JOptionPane.showMessageDialog(this, "Error al procesar la venta: "
+ ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
196.     }
197.
198.     private void actualizarFacturaPreview() {
199.         String nombreComercial = "Universidad Tecnológica del Perú";
200.         String rucEmisor = "R.U.C. 20462509236";
201.         String direccionSucursal = "Av. Arequipa, 265, Lima, Peru";
202.         String fechaEmision =
203.             LocalDate.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy")));
204.

```

```

201.         String factura = String.format(
202.             "----- FACTURA -----\\n" +
203.             "%s\\n" +
204.             "%s\\n" +
205.             "-----\\n" +
206.             "%s\\n" +
207.             "FACTURA ELECTRÓNICA\\n" +
208.             "Número de Factura: %s\\n" +
209.             "-----\\n" +
210.             "Fecha de Emisión: %s\\n" +
211.             "Comprador: %s\\n" +
212.             "RUC Comprador: %s\\n" +
213.             "-----\\n" +
214.             "Activo: %s\\n" +
215.             "Categoría: %s\\n" +
216.             "Fecha de Adquisición: %s\\n" +
217.             "Precio de Venta: S/ %s\\n" +
218.             "-----\\n" +
219.             "Forma de Pago: %s\\n" +
220.             "Detalles:\\n%s\\n" +
221.             "-----\\n" +
222.             "Gracias por su compra.\\n",
223.             nombreComercial,
224.             direccionSucursal,
225.             rucEmisor,
226.             txtNumeroFactura.getText(),
227.             fechaEmision,
228.             txtComprador.getText(),
229.             txtRUCComprador.getText(),
230.             nombre,
231.             categoria,
232.             fechaAdquisicion,
233.             txtPrecioVenta.getText(),
234.             cmbFormaPago.getSelectedItem(),
235.             txtDetalleVenta.getText()
236.         );
237.
238.         txtFacturaPreview.setText(factura);
239.     }
240.
241.     private void agregarListenersParaPrevisualizacion() {
242.         DocumentListener listener = new DocumentListener() {
243.             @Override
244.             public void insertUpdate(DocumentEvent e) {
245.                 actualizarFacturaPreview();
246.             }
247.
248.             @Override
249.             public void removeUpdate(DocumentEvent e) {
250.                 actualizarFacturaPreview();
251.             }
252.
253.             @Override
254.             public void changedUpdate(DocumentEvent e) {
255.                 actualizarFacturaPreview();
256.             }
257.         };
258.
259.         txtPrecioVenta.getDocument().addDocumentListener(listener);
260.         txtComprador.getDocument().addDocumentListener(listener);
261.         txtRUCComprador.getDocument().addDocumentListener(listener);
262.         txtDetalleVenta.getDocument().addDocumentListener(listener);
263.         cmbFormaPago.addActionListener(e -> actualizarFacturaPreview());
264.     }
265.
266.     private int obtenerNumeroFactura() {

```

```
267.         int numeroFactura = 1;
268.         String sql = "SELECT COUNT(*) AS total_ventas FROM ventas WHERE
269.             DATE(fecha_venta) = CURDATE()";
270.         try (Connection conn = ConexionBD.getConnection();
271.              PreparedStatement pstmt = conn.prepareStatement(sql);
272.              ResultSet rs = pstmt.executeQuery()) {
273.
274.             if (rs.next()) {
275.                 numeroFactura += rs.getInt("total_ventas");
276.             }
277.
278.         } catch (SQLException e) {
279.             e.printStackTrace();
280.         }
281.
282.         return numeroFactura;
283.     }
284. }
```