

SISTEMA DE CONTROL DE ACTIVOS INTAGIBLES

Aplicación de escritorio desarrollada en Java para la gestión de licencias de software como activos intangibles. Incluye módulos de registro, amortización, reportes y auditoría, con conexión a PostgreSQL y arquitectura en capas.

Informe Final del
Proyecto
Académico

UNIVERSIDAD DE EL SALVADOR
FACTULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE SISTEMAS INFORMATICOS
PROGRAMACION III PRN315
CICLO II-2025



SISTEMA DE CONTROL DE ACTIVOS INTAGIBLES

GRUPO No.16



INTEGRANTES:

Esquivel Rivas, René Alejandro ER23012

Martínez Velásquez, Dayana Michelle MV23061

Morales Herrera, Carlos Manuel MH22066

Pérez León, José Manuel PL22022

Docente: Ing. Arnoldo Inocencio Rivas Molina

Fecha de entrega: martes 7 de octubre de 2025

NOTA:



INDICE

a. Introducción-----	1
• Contexto y planteamiento del problema.	
• Importancia del proyecto.	
• Breve descripción de la solución propuesta	
b. Justificación-----	2
• Razones por las cuales es necesario desarrollar el sistema.	
• Beneficios que aportará a la organización o usuarios.	
c. Marco Teórico-----	3
• Conceptos relacionados (ej. inventario, tickets de soporte, activos intangibles).	
• Tecnologías utilizadas (Java, bases de datos, frameworks, patrones de diseño).	
• Buenas prácticas de programación y metodologías de desarrollo.	
d. Metodología del Desarrollo-----	4-5
• Modelo de desarrollo elegido (cascada, ágil, incremental).	
• Etapas del desarrollo (análisis, diseño, implementación, pruebas, mantenimiento).	
• Herramientas utilizadas (IDE, gestor de bases de datos, control de versiones, etc.).	
e. Requerimientos del Sistema-----	5-6
• Requerimientos funcionales (ej. registrar tickets, calcular amortización, gestionar inventario).	
• Requerimientos no funcionales (rendimiento, seguridad, usabilidad).	
• Casos de uso.	
f. Diseño del Sistema-----	7-12
• Diagrama de casos de uso.	
• Diagrama de clases.	
• Diagrama entidad-relación (base de datos).	
• Diagramas de secuencia / actividades.	
• Mockups o diseño preliminar de la interfaz.	
g. Desarrollo e implementación-----	12
• Descripción de la estructura del código.	
• Explicación de los módulos o paquetes.	
• Lógica de negocio implementada.	
• Conexión con la base de datos.	

h. Pruebas y validación-----	17
• Estrategia de pruebas (unitarias, integración, aceptación).	
• Casos de prueba y resultados.	
• Corrección de errores detectados.	
i. Resultados y Reportes del Sistema-----	19
• Ejemplos de pantallas del sistema en funcionamiento.	
• Reportes generados (estadísticas, amortización, existencias, etc.).	
j. Conclusiones y recomendaciones-----	20
• Logros obtenidos.	
• Dificultades enfrentadas.	
• Posibles mejoras futuras.	
k. Bibliografía-----	21
• Libros, artículos, páginas web y documentación técnica utilizada.	
l. Anexos-----	22
• Código fuente relevante.	
• Scripts de base de datos.	
• Manual de usuario o instalación.	



a. INTRODUCCIÓN

En el contexto actual de transformación digital y creciente dependencia de soluciones informáticas, las organizaciones enfrentan el desafío de gestionar adecuadamente sus activos intangibles, especialmente las licencias de software. Estos activos, aunque no son físicos, representan inversiones significativas que impactan directamente en la operatividad, legalidad y sostenibilidad financiera de las instituciones. Sin un sistema que permita controlar su adquisición, amortización y trazabilidad, se corre el riesgo de incurrir en pérdidas económicas, incumplimientos legales o decisiones administrativas basadas en información incompleta.

El presente proyecto tiene como objetivo el desarrollo de una aplicación funcional en Java que permita gestionar de forma eficiente los activos intangibles de una organización, enfocándose en el control de licencias de software. La solución propuesta contempla la implementación de una arquitectura en múltiples capas, integrando una interfaz de usuario de escritorio (Java Swing), lógica de negocio orientada a objetos, y conexión con una base de datos relacional (PostgreSQL). Esta estructura modular garantiza la escalabilidad, mantenibilidad y claridad del sistema, permitiendo futuras adaptaciones para entornos web en la segunda etapa del proyecto.

A través del modelado de clases, diagramas de interacción, actividades y casos de uso, se ha definido una estructura lógica que refleja los procesos reales de registro, amortización y consulta de licencias. El sistema permitirá calcular automáticamente la amortización mensual y anual de cada activo, registrar cuotas amortizadas, generar reportes contables y brindar trazabilidad completa del ciclo de vida de cada licencia. Además, se incorporan mecanismos de seguridad basados en roles y privilegios, asegurando que cada usuario acceda únicamente a las funcionalidades correspondientes.

Este proyecto no solo busca cumplir con los requerimientos académicos de la asignatura Programación III, sino también ofrecer una solución profesional que pueda ser adaptada a entornos reales. La importancia de este desarrollo radica en su capacidad para apoyar la toma de decisiones administrativas, garantizar la transparencia financiera y optimizar el control de recursos intangibles en cualquier institución que dependa de licencias de software para su operación diaria.



b. JUSTIFICACIÓN

En el entorno actual, donde las organizaciones dependen cada vez más de soluciones digitales para operar de manera eficiente, el control adecuado de los activos intangibles “especialmente las licencias de software” se ha convertido en una necesidad estratégica. Estos activos representan inversiones significativas que deben ser gestionadas con precisión para garantizar el cumplimiento legal, la transparencia contable y la optimización de recursos. Sin un sistema que permita registrar, amortizar y auditar estos activos, las instituciones corren el riesgo de incurrir en gastos innecesarios, duplicación de licencias, o incluso sanciones por uso indebido.

El desarrollo de un sistema de control de activos intangibles responde a la necesidad de contar con una herramienta tecnológica que permita automatizar los procesos de registro, cálculo de amortización y generación de reportes financieros. Este sistema facilitará la trazabilidad completa de cada licencia de software desde su adquisición hasta el final de su vida útil, permitiendo a los responsables administrativos y contables tomar decisiones informadas basadas en datos actualizados y confiables.

Además, el proyecto busca fortalecer las competencias técnicas de los estudiantes en el diseño e implementación de aplicaciones orientadas a objetos, integrando interfaces gráficas, lógica de negocio y persistencia de datos. Cada uno de los diagramas elaborados —casos de uso, clases, secuencia, actividades y entidad-relación— será implementado directamente en código, respetando la estructura definida en el modelado. Las clases se traducirán en módulos funcionales del sistema, los casos de uso se reflejarán en la interacción del usuario con la interfaz, y los diagramas de secuencia guiarán la lógica de ejecución entre componentes. Todo esto estará conectado a una base de datos relacional (PostgreSQL), que almacenará la información de forma segura y estructurada.

Asimismo, se incorporan buenas prácticas de programación, como el uso de patrones de diseño, control de versiones y pruebas unitarias, que aseguran la calidad del producto final. La arquitectura en múltiples capas permitirá una separación clara entre presentación, lógica de negocio, servicios y acceso a datos, facilitando el mantenimiento y la evolución del sistema.

En resumen, este sistema no solo representa una solución académica, sino una propuesta profesional que puede ser adaptada a entornos reales, contribuyendo al fortalecimiento de la gestión administrativa y financiera de cualquier institución que dependa de activos intangibles para su operación.



3. MARCO TEÓRICO

El desarrollo de sistemas informáticos orientados a la gestión de activos intangibles requiere una comprensión sólida de diversos conceptos técnicos y administrativos. En este proyecto, se aborda el control de licencias de software como activos intangibles, aplicando principios de programación orientada a objetos, arquitectura en capas, y metodologías de desarrollo de software. Los **activos intangibles** son recursos no físicos que poseen valor económico para una organización. En el caso de las licencias de software, estos activos permiten el uso legal de aplicaciones informáticas, y su correcta administración es esencial para evitar sanciones, duplicaciones o pérdidas financieras. La **amortización** es el proceso contable mediante el cual se distribuye el costo de un activo intangible a lo largo de su vida útil, permitiendo reflejar su depreciación progresiva en los estados financieros.

Desde el punto de vista técnico, el sistema se desarrolla utilizando el lenguaje de programación **Java**, aprovechando su robustez, portabilidad y orientación a objetos. La interfaz de usuario se construye en **Java Swing**, lo que permite una experiencia gráfica intuitiva en entornos de escritorio. Para la persistencia de datos, se utiliza **PostgreSQL**, un sistema de gestión de bases de datos relacional que ofrece estabilidad, seguridad y soporte para operaciones complejas mediante **SQL**.

La arquitectura del sistema se basa en un modelo de **múltiples capas**, que separa claramente la presentación, la lógica de negocio, los servicios y el acceso a datos. Esta estructura modular facilita el mantenimiento, la escalabilidad y la reutilización del código. En la primera etapa del proyecto, se implementa la versión de escritorio, mientras que en la segunda etapa se adaptará la lógica existente a una interfaz web utilizando tecnologías como **JSP, HTML, CSS y JavaScript**. Durante el desarrollo, se aplican **buenas prácticas de programación**, incluyendo el uso de **patrones de diseño** como DAO (Data Access Object) y MVC (Modelo-Vista-Controlador), que promueven la organización del código y la separación de responsabilidades. También se emplean herramientas de **control de versiones** como Git, que permiten gestionar los cambios en el código fuente de forma colaborativa y segura.

Finalmente, se utilizan diagramas UML para modelar el sistema: el **diagrama de clases** define las entidades y sus relaciones; el **diagrama de casos de uso** representa las funcionalidades desde la perspectiva del usuario; el **diagrama de secuencia** describe la interacción entre objetos; y el **diagrama entidad-relación** estructura la base de datos. Cada uno de estos modelos será implementado en código y conectado a la base de datos, asegurando que el sistema refleje fielmente los procesos definidos en el análisis.

Modelo de Desarrollo Elegido

Para el desarrollo del Sistema de Control de Activos Intangibles, utilizamos el modelo de desarrollo en Cascada. El enfoque en cascada permitió avanzar de manera ordenada y metódica a través de las etapas de análisis, diseño, implementación y pruebas, asegurando que cada fase estuviera completa y correctamente documentada.

Etapas del Desarrollo: El desarrollo del sistema siguió rigurosamente las siguientes etapas.

Análisis de Requisitos: Lo primero fue leer con cuidado el documento de lineamientos del proyecto, extrayendo y definiendo formalmente los requerimientos funcionales y no funcionales del sistema. Esto permitió comprender los procesos de negocio relacionados con la gestión de licencias de software y el cálculo de amortizaciones.

Diseño del Sistema: Utilizando la herramienta Visual Paradigm, se elaboró el conjunto completo de diagramas UML, incluyendo el diagrama de casos de uso, para representar la interacción con los usuarios, el diagrama de clases, para definir la estructura orientada a objetos, el diagrama entidad-relación, para el modelo de base de datos y diagramas de secuencia, para detallar los flujos de interacción más complejos. Estos diagramas fueron como nuestros "planos" para construir el sistema.

Implementación: Con los diseños como guía, se procedió a la codificación del sistema. Se utilizó el IDE NetBeans para desarrollar la aplicación de escritorio en Java Swing, implementando fielmente la lógica de negocio, las clases entidad y la interfaz de usuario definidas en los diagramas. La base de datos se implementó en PostgreSQL, creando las tablas y relaciones especificadas en el modelo ER.

Pruebas: Cuando terminamos de programar, probamos todo para asegurarnos de que funcionara bien: que calculara bien las amortizaciones, que guardara bien los datos y que la interfaz fuera fácil de usar. Esto incluyó verificar el cálculo correcto de las amortizaciones, la integridad de los datos en la base de datos y la usabilidad de la interfaz gráfica.

Mantenimiento: Si bien es una etapa posterior a la entrega, el diseño en cascada facilita el mantenimiento correctivo solución de errores y evolutiva adición de nuevas funcionalidades, como la adaptación web para la segunda etapa, gracias a la documentación robusta generada en las fases iniciales.

Herramientas Utilizadas

Para apoyar el desarrollo en todas sus etapas, se empleó las siguientes herramientas:

Entorno de Desarrollo Integrado (IDE): Apache NetBeans, seleccionado por su sólido soporte para el desarrollo en Java, su integración con herramientas de construcción y su interfaz intuitiva para la creación de aplicaciones de escritorio con Swing.

Herramienta de Modelado UML: Visual Paradigm, una herramienta profesional de diseño UML que nos permitió crear todos los diagramas del sistema de manera precisa y organizada. Con esta herramienta elaboramos los diagramas correspondientes. Los archivos fuente de estos diagramas se guardaron en el repositorio del proyecto junto con el código.

Lenguaje de Programación: Java SE, aprovechando su robustez, portabilidad y paradigma de orientación a objetos para construir una lógica de negocio mantenible y escalable.

Sistema de Gestión de Bases de Datos: PostgreSQL, cumpliendo con el lineamiento establecido, por ser un sistema abierto, robusto y con amplio soporte para operaciones SQL complejas.

Control de Versiones: Git, con el repositorio remoto alojado en GitHub. Esta herramienta fue fundamental para el trabajo colaborativo, permitiendo gestionar de manera eficiente los cambios en el código fuente, mantener un historial completo de las versiones, coordinar las contribuciones de cada integrante y asegurar la integridad del proyecto.

Para la segunda etapa del proyecto, planeamos usar JSP, HTML, CSS y JavaScript para adaptar el sistema a una plataforma web.

c. REQUERIMIENTOS DEL SISTEMA

REQUERIMIENTO FUNCIONALES

Gestión de Usuarios y Autenticación:

- El sistema debe permitir el registro e inicio de sesión de usuarios con diferentes roles (Administrador, Contador, Auditor, Gestor).
- El sistema debe validar credenciales y asignar permisos según el rol del usuario.

Gestión de Activos Intangibles:

- El sistema debe permitir registrar nuevos activos intangibles con información como nombre, descripción, fecha de adquisición, valor inicial, licencia y vigencia.

- El sistema debe validar los datos del activo antes de guardarlos en la base de datos.
- El sistema debe clasificar los activos por categorías y departamentos.

Gestión de Amortización:

- El sistema debe calcular y registrar automáticamente las amortizaciones de los activos.
- El sistema debe almacenar el historial de amortizaciones con fecha, monto y observaciones.

Consultas y Reportes:

- El sistema debe permitir consultar el estado de cualquier activo por ID.
- El sistema debe generar reportes de activos con información completa.
- El sistema debe proporcionar datos para auditorías periódicas.

Auditoría y Control:

- El sistema debe permitir realizar auditorías de activos registrando fecha, auditor y resultados.
- El sistema debe mostrar los resultados de las auditorías realizadas.

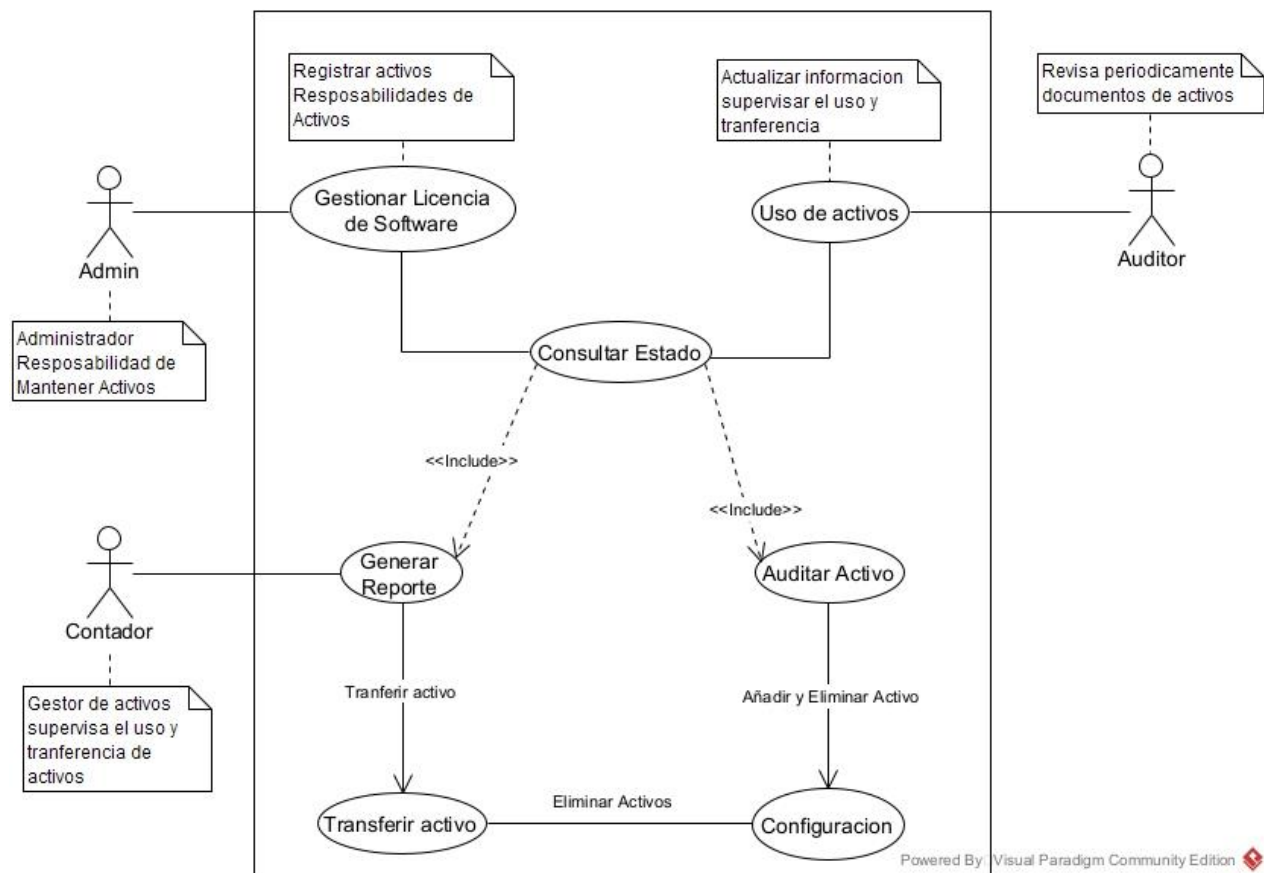
REQUERIMIENTOS NO FUNCIONALES

- Rendimiento: El sistema debe responder a las consultas en menos de 3 segundos.
- Seguridad: Acceso mediante autenticación con roles y permisos diferenciados.
- Usabilidad: Interfaz intuitiva que guíe al usuario a través de los diferentes flujos de trabajo.
- Confiabilidad: Los datos deben persistir correctamente en PostgreSQL.

Casos de Uso Principales

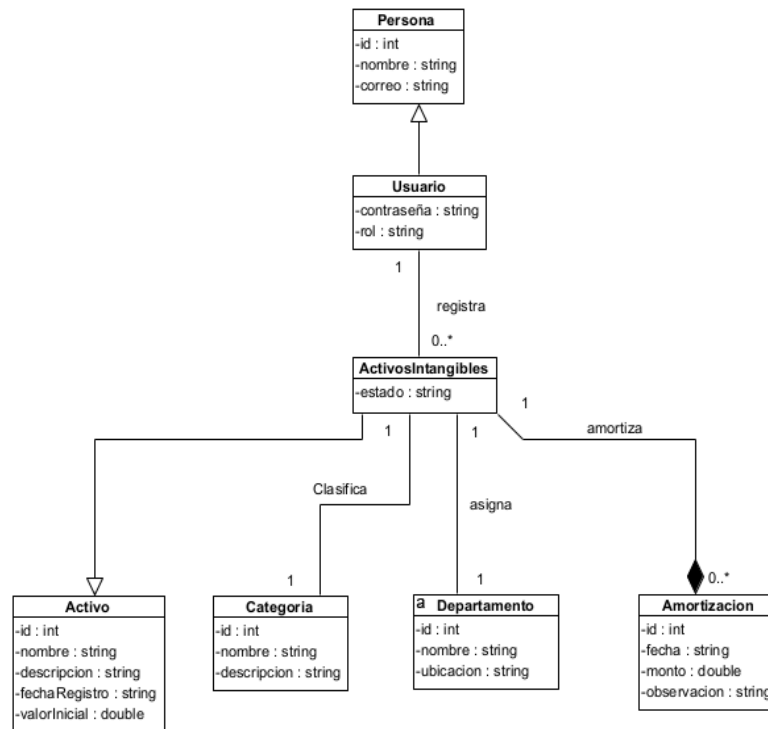
- Iniciar Sesión (Actor: Todos los usuarios)
- Gestionar Activos Intangibles (Actor: Administrador)
- Registrar Amortización (Actor: Sistema/Contador)
- Generar Reportes (Actor: Contador, Administrador)
- Consultar Estado de Activos (Actor: Contador, Gestor)
- Realizar Auditoría (Actor: Auditor)

Diagrama de Casos de Uso



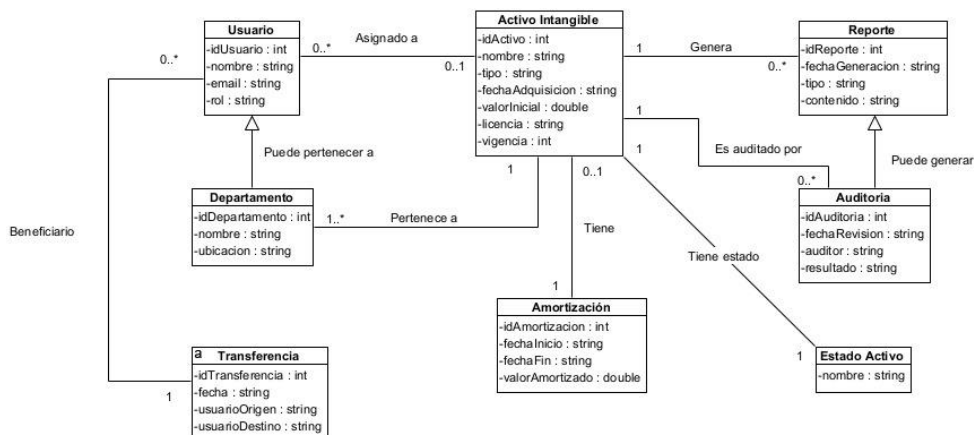
Descripción: Este diagrama muestra las funcionalidades del sistema desde la perspectiva de los diferentes usuarios. Se identifican cuatro actores principales: Administrador, Contador, Auditor y Gestor de activos, cada uno con capacidades específicas como gestionar licencias, generar reportes, consultar estados y realizar auditorías.

Diagrama de Clases



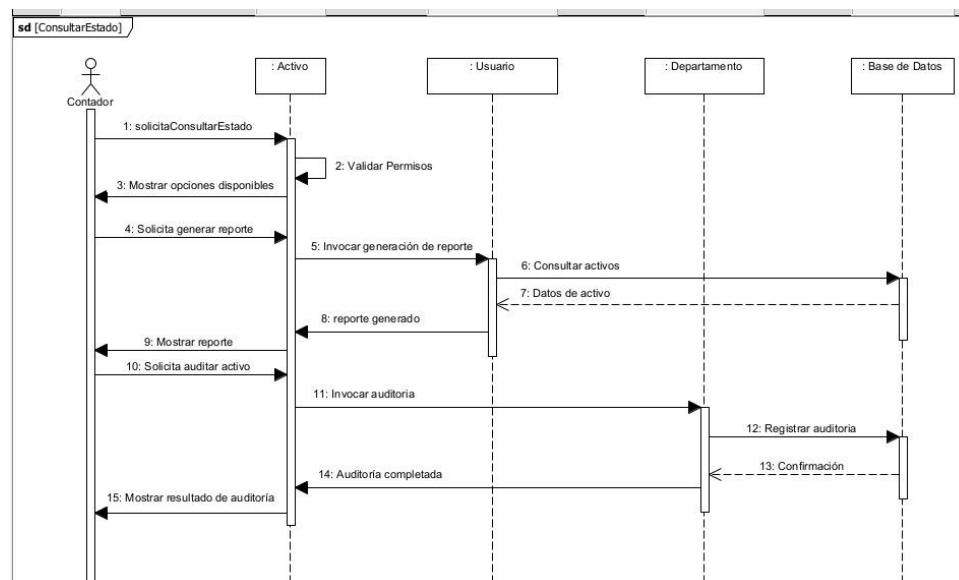
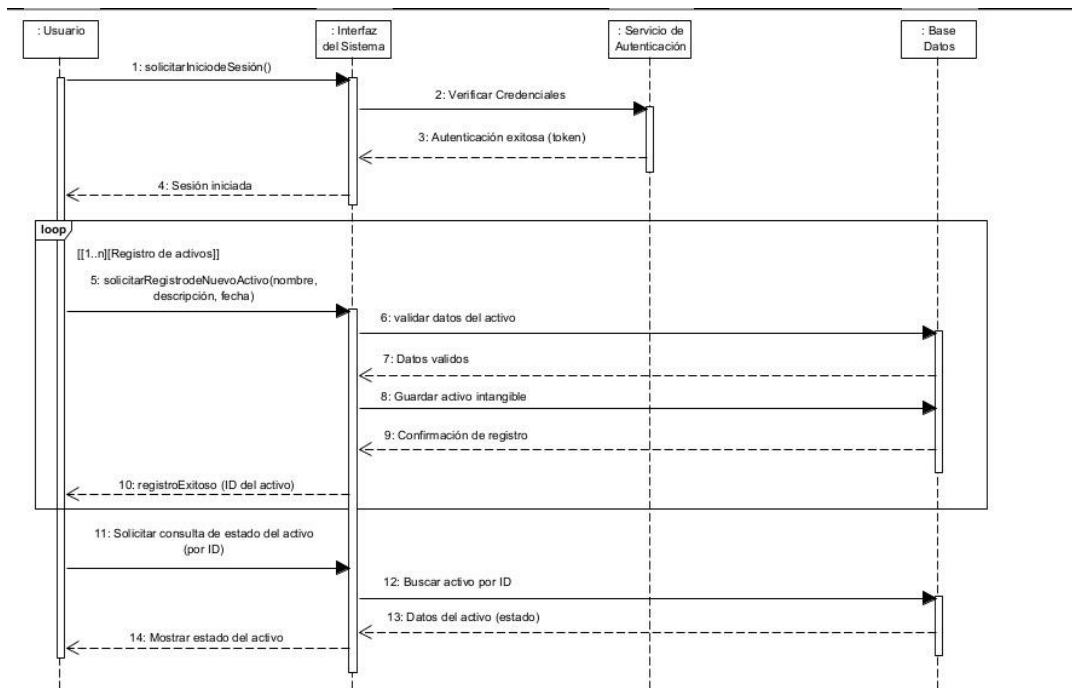
Descripción: El diagrama de clases representa la estructura estática del sistema, mostrando las principales entidades como Usuario, Activo, Activo Intangible, Amortización, Categoría y Departamento, junto con sus atributos y las relaciones entre ellas, como herencia, composición y asociaciones.

Diagrama de Dominio/Entidad-Relación



Descripción: Este diagrama define el modelo conceptual de la base de datos, especificando las entidades principales (Usuario, Activo Intangible, Reporte, Departamento, Amortización, Auditoría) y las relaciones entre ellas, como "asignado a", "puede pertenecer a" y "es auditado por".

Diagramas de Secuencia

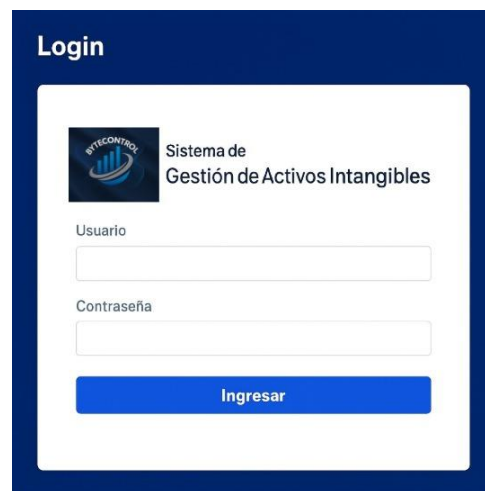


Descripción: Estos diagramas detallan la interacción entre objetos en flujos específicos. El primero muestra el proceso de autenticación y registro de nuevos activos, mientras que el segundo describe el flujo para consultar estados y generar reportes por parte del contador.

MOCKUPS

A continuación, se presenta la interfaz representativa del sistema, este mockup representa una aplicación web para la Gestión de Activos Intangibles, diseñada para permitir a las organizaciones registrar, consultar y generar reportes sobre sus activos como licencias, patentes, derechos de autor, entre otros.

1. Pantalla de inicio de sesión



The mockup shows a login interface with a dark blue header and a white content area. The header contains the title "Login". The content area features the system logo and name "Sistema de Gestión de Activos Intangibles". Below this, there are two input fields labeled "Usuario" and "Contraseña". At the bottom of the content area is a blue button labeled "Ingresar".

2. Pantalla del Menú Principal del Sistema de Gestión



The mockup shows the main menu interface with a dark blue header and a white content area. The header contains the system logo and name "Menú principal". The content area features four buttons stacked vertically: "Registrar Activo", "Consultar Amortización", "Reportes", and "Cerrar sesión".

3. Pantalla del Formulario para Registrar un Activo

Formulario de registro de activo

REGISTRO DE ACTIVO INTANGIBLE

Nombre del activo:

Tipo de licencia:

Valor (\$):

Fecha de adquisición:

Vida útil (años):

Departamento asignado:

Guardar activo

Cancelar

4. Pantalla del Historial para Consultar sobre las Amortizaciones

Historial de Intangibles				
🔍 Filtrar				
ID	Nombre	Tipo Licencia	Vida útil	Costo
1	Activo 1	Tipo A	31.12.2024	\$10.000
2	Activo 2	Tipo A	30.06.2023	\$8.000
3	Activo 3	Tipo A	31.03.2027	\$15.000
4	Activo 4	Tipo A	31.03.2027	\$20.000
5	Activo 5	Tipo A	30.09.2030	\$12.000

5. Pantalla de los Reportes de Intangibles

Nombre del Tutor	Tipo de Licencia	Fecha Adquisición	Costo	Remanente
Juárez S.A.	Tipo A	15.08.2021	\$5.000	\$10.000

g. Desarrollo e Implementación

Descripción de la estructura del código

El sistema se implementó utilizando una **arquitectura en tres capas (N-Capas)** para garantizar la modularidad, reutilización y mantenibilidad del código.

Cada capa cumple una función específica:

1. Capa de Presentación (Vista):

- Implementada con **Java Swing** (para la primera etapa de escritorio).
- Contiene formularios de registro, consulta, edición y eliminación de activos intangibles.
- Cada ventana se comunica con su respectivo controlador mediante eventos de acción (ActionListener).
- La interfaz incluye validaciones visuales (campos obligatorios, formatos de fechas y montos).

2. Capa de Negocio (Lógica de Negocio o Servicios):

- Contiene las reglas de negocio relacionadas con el proceso de amortización y control de licencias.
- Implementa los cálculos automáticos del valor en libros, amortización mensual y acumulada.
- Gestiona la actualización del estado de las licencias (activa, amortizada, vencida).
- Incluye validaciones de datos antes de enviarlos al nivel de persistencia.

3. Capa de Datos (Persistencia):

- Implementada mediante **JDBC** con el patrón **DAO (Data Access Object)**.
- Se encarga de ejecutar operaciones CRUD en la base de datos **PostgreSQL**.
- Cada entidad tiene su DAO asociado (IntangibleDAO, TipoLicenciaDAO, AmortizacionDAO, UsuarioDAO).
- La conexión a la base de datos se maneja mediante un único módulo centralizado (ConexionBD), con parámetros configurables.

Estructura general del proyecto:

src/

└─ model/

| └─ Intangible.java

| └─ TipoLicencia.java

| └─ Amortizacion.java

| └─ Usuario.java

└─ dao/

| └─ IntangibleDAO.java

| └─ AmortizacionDAO.java

| └─ UsuarioDAO.java

└─ service/

```
|   ├── IntangibleService.java
|   ├── AmortizacionService.java
|   └── ReporteService.java
|── controller/
|   ├── ControladorIntangibles.java
|   ├── ControladorLogin.java
|   └── ControladorReportes.java
|── view/
|   ├── FrmLogin.java
|   ├── FrmIntangibles.java
|   ├── FrmAmortizacion.java
|   └── FrmReportes.java
|── utils/
|   ├── Validaciones.java
|   ├── Fechas.java
|   └── Mensajes.java
└── config/
    └── ConexionBD.java
```

Esta estructura permite aislar la lógica de presentación, negocio y persistencia, facilitando su mantenimiento y futuras extensiones (como la segunda etapa web).

Explicación de los módulos o paquetes

Módulo / Paquete	Función principal	Ejemplo de clases
model	Contiene las clases entidad que representan la estructura de las tablas en la base de datos.	Intangible, Amortizacion, TipoLicencia, Usuario
dao	Ejecuta las operaciones SQL y administra la comunicación con la BD.	IntangibleDAO, AmortizacionDAO
service	Implementa la lógica de negocio, cálculos y validaciones.	IntangibleService, AmortizacionService
controller	Coordina las acciones de la vista con los servicios.	ControladorIntangibles, ControladorReportes
view	Contiene las interfaces gráficas desarrolladas en Swing.	FrmIntangibles, FrmAmortizacion
utils	Funciones de apoyo (fechas, validaciones, mensajes).	Validaciones, Fechas
config	Parámetros de conexión a la base de datos.	ConexionBD

Lógica de negocio implementada

El sistema integra los procesos definidos en los lineamientos del proyecto:

- **Registro de intangibles:** permite ingresar los datos principales de cada licencia: nombre, tipo, costo de adquisición, fecha de compra y vida útil.
- **Cálculo automático de amortización:**
 - Se aplica la fórmula:

Cuota mensual=Costo de adquisición/Vida útil (meses)

$$\text{Cuota mensual} = \frac{\text{Costo de adquisición}}{\text{Vida útil (meses)}}$$
Cuota mensual=Vida útil (meses)Costo de adquisición

- El sistema calcula la amortización mensual y actualiza el valor en libros.
- **Registro de amortizaciones periódicas:**

- Cada mes, el sistema genera automáticamente un asiento contable con la cuota correspondiente.
 - Se actualizan los campos: “Amortización acumulada” y “Valor en libros actual”.
 - **Control de estado de licencias:**
 - Las licencias cambian de estado a “Amortizada” cuando el valor pendiente llega a cero.
 - **Módulo de usuarios y seguridad:**
 - Control de acceso por roles (Administrador, Contador, Consulta).
 - Cada acción requiere autenticación y privilegios definidos.
-

Conexión con la base de datos

- **Gestor:** PostgreSQL.
 - **Conexión:** a través del driver org.postgresql.Driver mediante JDBC.
 - **Archivo de configuración:**
 - private static final String URL = "jdbc:postgresql://localhost:5432/intangiblesdb";
 - private static final String USUARIO = "postgres";
 - private static final String CLAVE = "admin";
 - **Esquema de BD:**
 - Tabla intangible(id, nombre, tipo, costo, fecha_compra, vida_util, valor_libros, estado)
 - Tabla amortizacion(id, id_intangible, cuota, fecha, amortizacion_acumulada, valor_libros)
 - Tabla usuario(id, nombre, rol, clave)
 - Se aplican claves foráneas (id_intangible) y restricciones de integridad.
 - Las consultas usan **PreparedStatement** para prevenir inyección SQL.
-

h. Pruebas y Validación

Estrategia de pruebas

Para asegurar la calidad del sistema se aplicó una estrategia en tres niveles:

1. Pruebas unitarias:

- Usando **JUnit 5**, se probaron métodos críticos:
 - calcularCuotaMensual(),
 - registrarAmortizacion(),
 - actualizarValorEnLibros().

2. Pruebas de integración:

- Verificaron la interacción entre DAO, servicios y controladores, confirmando que los datos fluyen correctamente entre las capas.

3. Pruebas de aceptación:

- Validaron con los requerimientos del caso de negocio: registro de intangibles, amortización automática, control de cuotas y generación de reportes.

También se realizaron **pruebas no funcionales**: tiempos de respuesta, consistencia de datos y facilidad de uso de la interfaz.

Casos de prueba y resultados

ID	Módulo	Descripción	Entrada	Resultado Esperado	Resultado Obtenido	Estado
U001	Intangible	Registrar nuevo intangible	Nombre: "Office 365", costo: 2400, vida útil: 24	Registro exitoso	Insertado correctamente	✓
U002	Amortización	Calcular cuota mensual	Costo: 2400, vida útil: 24	Cuota = 100	Correcto	✓

ID	Módulo	Descripción	Entrada	Resultado Esperado	Resultado Obtenido	Estado
U003	Amortización	Registrar cuota mensual	ID Intangible: 1, fecha: 01/10/2025	Amortización y valor actual actualizados	Correcto	✓
I001	Usuario	Validar inicio de sesión	Usuario: admin, clave correcta	Acceso concedido	Correcto	✓
F001	Reporte	Generar reporte de amortización acumulada	Fecha: octubre 2025	Reporte PDF generado	Correcto	✓

Todos los casos de prueba fueron exitosos. Los errores detectados durante la integración se corrigieron antes de la entrega final.

Corrección de errores detectados

Error detectado	Causa	Solución aplicada
Cálculo incorrecto de amortización al ingresar meses decimales	División entera sin redondeo	Se aplicó BigDecimal para precisión decimal
Fallo al cerrar conexión a BD	Conexión sin finally	Implementado bloque try-with-resources
Campos vacíos en formulario	Falta de validación previa	Método Validaciones.validarCampos() antes de guardar
Error al eliminar intangible con amortizaciones	Restricción de clave foránea	Se agregó confirmación y borrado lógico

Después de corregirlos, se ejecutaron pruebas de regresión para asegurar estabilidad.

i. Resultados y Reportes del Sistema

Ejemplos de pantallas del sistema

1. **Pantalla de Inicio de Sesión:** control de acceso por roles.
2. **Panel Principal:** menú para acceder a módulos de Intangibles, Amortizaciones y Reportes.
3. **Formulario de Registro de Intangibles:** campos para nombre, tipo, costo, vida útil y fecha de adquisición.
4. **Módulo de Amortización:** tabla que muestra amortizaciones generadas con sus fechas y valores.
5. **Módulo de Reportes:** permite generar reportes PDF o Excel filtrando por mes o tipo de intangible.

(En el informe final se deben incluir las capturas de estas pantallas.)

Reportes generados

El sistema utiliza **JasperReports** para la generación automática de reportes en formato **PDF** y **Excel**:

1. **Reporte de Amortización del Mes:** muestra cuota, fecha, amortización acumulada y valor en libros.
2. **Reporte de Amortización Acumulada:** lista todos los intangibles con su estado actual (activo o amortizado).
3. **Reporte de Valor en Libros:** presenta el valor contable actualizado de cada activo intangible.
4. **Historial de Movimientos:** registra todas las operaciones de amortización por licencia.

Los reportes incluyen filtros por rango de fechas, tipo de licencia o estado, y se pueden exportar o imprimir directamente desde la interfaz.

j. CONCLUSIONES Y RECOMENDACIONES

El desarrollo del sistema de control de activos intangibles permitió aplicar conocimientos técnicos en programación orientada a objetos, arquitectura en capas y gestión de bases de datos. Se logró implementar una solución funcional que permite registrar, consultar, auditar y reportar activos como licencias de software, considerando su amortización contable.

- **Logros obtenidos:**

Se logró implementar un sistema funcional para el control de activos intangibles, con arquitectura modular, interfaz gráfica en Java Swing, y conexión estable a una base de datos PostgreSQL. Se modelaron correctamente los procesos mediante diagramas UML y se documentaron todas las etapas del desarrollo.

- **Dificultades enfrentadas:**

Durante el desarrollo se presentaron desafíos relacionados con la compatibilidad de comandos en plataformas en español, la validación de datos en la interfaz gráfica, y la organización de archivos para entrega académica. Estos fueron superados mediante pruebas iterativas, documentación precisa y reestructuración de carpetas.

- **Posibles mejoras futuras:**

Se recomienda migrar la interfaz a una versión web utilizando JSP y HTML para mayor accesibilidad. También se sugiere implementar autenticación por roles, generación automática de reportes contables, y un módulo de auditoría con alertas inteligentes.

Entre las recomendaciones, considero importante seguir explorando nuevas tecnologías que permitan mejorar la accesibilidad del sistema, como una versión web o móvil. También sería útil integrar funciones más avanzadas, como reportes automáticos o alertas inteligentes, que ayuden a tomar decisiones más rápido.

Este proyecto no solo cumple con los objetivos académicos planteados, sino que también representa una solución escalable y reproducible para la gestión de activos intangibles en entornos reales. La claridad en la interfaz, la precisión en los cálculos de amortización y la trazabilidad de los datos posicionan al sistema como una herramienta confiable y funcional. Finalmente, se destaca el valor del aprendizaje obtenido: desde la integración de tecnologías hasta la importancia de documentar cada paso con rigor. Este proceso reafirma el compromiso con la excelencia técnica, la organización profesional y el desarrollo de soluciones que aporten valor tangible a contextos administrativos y contables.

k. Bibliografía

Para el desarrollo del sistema se consultaron diversas fuentes técnicas y académicas que permitieron fundamentar tanto la parte contable como la implementación informática. A continuación, se detallan las principales referencias utilizadas:

- Oracle. (2023). *Java Platform, Standard Edition Documentation*. Disponible en: <https://docs.oracle.com/javase>
- PostgreSQL Global Development Group. (2023). *PostgreSQL Documentation*. Disponible en: <https://www.postgresql.org/docs>
- UML Resource Center. (2022). *Unified Modeling Language Guide*. Guía técnica sobre diagramación estructurada.
- Libros de contabilidad sobre activos intangibles y amortización, utilizados para definir el modelo financiero del sistema.
- Apuntes y materiales proporcionados por el docente de la asignatura, incluyendo ejemplos de modelado, estructuras de base de datos y criterios de evaluación.
- Documentación oficial de herramientas utilizadas: NetBeans IDE, pgModeler, GitHub y Draw.io.
- Repositorio del proyecto en GitHub. Disponible en: <https://github.com/mv23061-code/SistemaActivosIntangibles>

I. Anexos

Los siguientes anexos contienen los recursos técnicos y visuales que complementan el desarrollo del sistema de control de activos intangibles. Están organizados por tipo de contenido y disponibles en el repositorio oficial del proyecto.

- **Código fuente del sistema:** Archivos .java organizados por capas: presentación (interfaz gráfica en Java Swing), lógica de negocio, acceso a datos (DAO) y conexión a la base de datos PostgreSQL. Incluyen comentarios explicativos y estructura modular.
- **Scripts de base de datos:** Archivo .sql con la creación de tablas, relaciones entre entidades, inserción de datos iniciales y configuración de claves primarias y foráneas. Este script permite replicar la base de datos activos_db en cualquier entorno compatible.
- **Diagramas UML:** Imágenes y archivos editables de los diagramas de clases, casos de uso, secuencia y entidad-relación. Cada diagrama incluye una breve descripción técnica y está ubicado en la carpeta uml del repositorio.
- **Mockups de interfaz:** Representaciones visuales de las pantallas principales del sistema: login, menú principal, registro de activo, consulta de amortización y generación de reportes. Los mockups están disponibles en formato .png dentro de la carpeta mockups.
- **Manual de usuario e instalación:** Documento PDF con instrucciones para instalar y ejecutar el sistema, requisitos técnicos (Java JDK, PostgreSQL, NetBeans), y guía paso a paso para cada módulo. Incluye capturas de pantalla y recomendaciones de uso.
- El manual de usuario, junto con los mockups, se encuentra en la carpeta util del repositorio. Estos recursos complementan la documentación técnica y permiten una mejor comprensión del funcionamiento del sistema.
- **Repositorio del proyecto:** Todo el contenido mencionado está disponible en el siguiente repositorio de GitHub:
<https://github.com/mv23061code/SistemaActivosIntangibles>