Una señal de alto

El contenido generado por IA puede ser incorrecto.

TSU Desarrollo de Software Multiplataforma

**Materias:**

Programación Estructurada

Sistemas Operativos

Conmutación y Enrutamiento de Redes

**Actividad:**

Proyecto Final (Cinema World)

**Alumnos:**

Jimenez Cruz Obed de Jesus

Urquidez Arredondo Misael

Nava Pineda Eduardo

**Profesores:**

Dr. Marylu Lara Lagunes

Dr. Emer Ignacio Bernal Jimenez

Ing. Humberto Vejar Polanco

**Fecha de entrega:**

Martes 12 de Agosto del 2025

**Cuatrimestre:** 2      **Grupo:** 2-C

**Índice**

[Índice de Ilustraciones 2](#_Toc205919276)

[1.Introducción 3](#_Toc205919277)

[1.1 Propósito 4](#_Toc205919278)

[1.2 Ámbito del Sistema 4](#_Toc205919279)

[1.3 Definiciones, Acrónimos y Abreviaturas 5](#_Toc205919280)

[1.3 Referencias 6](#_Toc205919281)

[1.5 Visión General del Documento 7](#_Toc205919282)

[2. Descripción General 8](#_Toc205919283)

[2.1 Perspectiva del Producto 8](#_Toc205919284)

[2.2 Funciones del Producto 9](#_Toc205919285)

[2.3 Características de los Usuarios 10](#_Toc205919286)

[2.4 Restricciones 11](#_Toc205919287)

[2.5 Suposiciones y Dependencias 12](#_Toc205919288)

[2.6 Requisitos Futuros 13](#_Toc205919289)

[3. Requisitos Específicos 14](#_Toc205919290)

[3.1 Interfaces Externas 14](#_Toc205919291)

[3.2 Funciones 15](#_Toc205919292)

[3.3 Requisitos de Rendimiento 17](#_Toc205919293)

[3.4 Restricciones de Diseño 18](#_Toc205919294)

[3.5 Atributos del Sistema 19](#_Toc205919295)

[3.6 Otros Requisitos 20](#_Toc205919296)

[4. Apéndices 22](#_Toc205919297)

[5. Conclusión 32](#_Toc205919298)

[6. Referencias apa 33](#_Toc205919299)

# **Índice de Ilustraciones**

[Ilustración 1. DIagrama de flujo inicio de programa 22](#_Toc205883834)

[Ilustración 2. diagrama de flujo Inicio (C.W) 22](#_Toc205883835)

[Ilustración 3. Código de método de pago 23](#_Toc205883836)

[Ilustración 4. Código de pago 24](#_Toc205883837)

[Ilustración 5. Código de suscripción parte 1 25](#_Toc205883838)

[Ilustración 6. Código de suscripción parte 2 26](#_Toc205883839)

[Ilustración 7. Código de suscripción parte 3 27](#_Toc205883840)

[Ilustración 8. Código de Cinema World parte 1 28](#_Toc205883841)

[Ilustración 9.Código de Cinema World parte 2 29](#_Toc205883842)

[Ilustración 10.Código de Cinema World parte 3 30](#_Toc205883843)

[Ilustración 11. Código de Cinema World parte 4 31](#_Toc205883844)

# **1.Introducción**

Este documento presenta de manera integral el desarrollo, implementación y documentación de un proyecto de programación realizado en Java utilizando un entorno de trabajo remoto basado en Ubuntu. Se siguieron las prácticas recomendadas por el estándar IEEE 830 para la redacción de requisitos de software, integrando aspectos solicitados tanto en la materia de Programación como en la de Sistemas Operativos y, en menor medida, en Redes de Computadoras. El trabajo se ejecutó mediante la instalación y configuración del lenguaje Java desde la terminal de Ubuntu, aprovechando herramientas de administración remota como X2Go para acceder al entorno de trabajo. La gestión de versiones y colaboración se realizó a través de GitHub, asegurando un control ordenado y documentado del código fuente. La parte de redes se limitó a la verificación básica de conectividad mediante el uso de \*\*ping\*\*, cumpliendo con los requisitos solicitados. La documentación que aquí se presenta detalla cada una de las fases y elementos involucrados, desde el propósito del sistema hasta las restricciones y requisitos técnicos, con un enfoque formal y orientado a la reproducibilidad del proceso.

## **Propósito**

El propósito de este documento es definir y detallar exhaustivamente los requisitos funcionales y no funcionales del sistema desarrollado, facilitando la comprensión clara y precisa de las funcionalidades que la aplicación en Java debe cumplir. Se busca que este documento sirva como una guía completa tanto para desarrolladores que deseen implementar o mejorar el sistema, como para evaluadores y docentes que requieran verificar el cumplimiento de los objetivos académicos planteados. El proyecto está diseñado para simular un entorno real de trabajo, donde se aplican conocimientos de programación, administración de sistemas y redes, mediante la integración de herramientas modernas y estándares de calidad reconocidos internacionalmente. Además, esta documentación pretende unificar las mejores prácticas de desarrollo de software con el uso de sistemas operativos robustos, control de versiones mediante GitHub, y protocolos básicos de redes, como el ping para verificación de conectividad. De esta manera, se establece una base sólida y estructurada que permita futuras ampliaciones o integraciones con sistemas más complejos, garantizando que el software desarrollado pueda ser entendido, reproducido, mantenido y evolucionado por usuarios con conocimientos básicos o intermedios en programación y administración de sistemas.

## **Ámbito del Sistema**

El sistema descrito en este proyecto es una aplicación de consola desarrollada en Java, que funciona en un ambiente Linux, específicamente en una distribución Ubuntu configurada para permitir acceso remoto mediante X2Go. Esta aplicación tiene como función principal facilitar la captura y validación de datos relacionados con métodos de pago, tales como tarjetas de crédito o débito, PayPal y transferencias bancarias, mostrando posteriormente un resumen seguro y protegido de la información ingresada sin almacenar datos confidenciales. El ámbito del sistema se restringe a una aplicación local sin integración con bases de datos o servicios externos, enfocada en la práctica de conceptos fundamentales de programación, administración de sistemas y gestión de redes básicas. Se plantea así como una plataforma educativa que combina el aprendizaje teórico con la aplicación práctica, en un entorno controlado que simula condiciones reales de trabajo profesional. El sistema está diseñado para ser independiente y portable, con una arquitectura modular que permite una fácil extensión hacia futuras funcionalidades, como interfaces gráficas, bases de datos o comunicación en red. Este alcance delimitado permite que el usuario se concentre en aspectos esenciales del desarrollo de software y administración remota, sin complicaciones derivadas de integraciones complejas o dependencias externas. En resumen, el sistema funciona como una herramienta práctica y didáctica que cubre múltiples áreas de conocimiento, ofreciendo un entorno accesible y eficiente para el aprendizaje integral y la aplicación de estándares profesionales en el desarrollo de software.

## **1.3 Definiciones, Acrónimos y Abreviaturas**

Este apartado agrupa y explica los términos técnicos, acrónimos y abreviaturas usados a lo largo del documento, para facilitar la comprensión de lectores con distintos niveles de conocimiento:

* Java: Lenguaje de programación orientado a objetos, portable y robusto, utilizado para desarrollar la aplicación principal del proyecto.
* Ubuntu: Distribución de Linux basada en Debian, usada como sistema operativo del servidor para el desarrollo y pruebas del software.
* X2Go: Software que permite el acceso remoto a entornos gráficos en sistemas Linux de forma segura y eficiente, facilitando el trabajo colaborativo y la administración remota
* GitHub: Plataforma web para la gestión de repositorios Git, que facilita el control de versiones, colaboración en proyectos y seguimiento histórico de cambios.
* Ping: Herramienta de red para verificar la conectividad entre equipos en una red IP, enviando paquetes ICMP y midiendo tiempos de respuesta.
* IEEE 830: Estándar internacional que establece las recomendaciones para la especificación de requisitos de software, que guía la estructura y contenido de este documento.
* CVV: Currículo
* C.W: Cinema world
* JDK (Java Development Kit): Conjunto de herramientas necesarias para desarrollar, compilar y ejecutar aplicaciones Java.
* CLI (Command Line Interface): Interfaz de línea de comandos, por la cual el usuario interactúa con el sistema operativo o aplicaciones.
* VPN (Virtual Private Network): Tecnología que permite establecer conexiones seguras y privadas a través de redes públicas.

Estas definiciones aseguran que cualquier lector pueda entender sin ambigüedades los términos técnicos y siglas presentes en la documentación, facilitando su lectura y aplicación.

## **Referencias**

La realización de este proyecto y la elaboración de esta documentación se apoyaron en diversas fuentes confiables y documentos técnicos especializados, que garantizan la fundamentación adecuada y el cumplimiento de estándares reconocidos:

* Oracle Java Documentation: La documentación oficial de Oracle sobre el lenguaje Java, que abarca desde conceptos básicos hasta aspectos avanzados del lenguaje y su ecosistema.
* Ubuntu Official Documentation: Manuales y guías oficiales para la administración, instalación y configuración del sistema operativo Ubuntu, utilizados para establecer el entorno de desarrollo.
* GitHub Docs: Documentación oficial sobre el uso de GitHub para control de versiones, gestión de repositorios y colaboración en proyectos de software.
* IEEE Standard 830-1998: Estándar internacional que regula la especificación de requisitos de software, proporcionando las directrices para la correcta elaboración de documentos técnicos como el presente.
* Tutoriales y guías en línea sobre la instalación y configuración de X2Go, que facilitaron la implementación del acceso remoto al servidor.
* Manuales y referencias sobre comandos básicos de Linux para administración de sistemas y redes, esenciales para las operaciones realizadas en el entorno Ubuntu.

Estas fuentes constituyen la base técnica y metodológica que respalda la calidad y confiabilidad del desarrollo y la documentación del proyecto.

## **1.5 Visión General del Documento**

Este documento se estructura en cuatro secciones principales que permiten al lector navegar de forma lógica y eficiente por el contenido, facilitando tanto la comprensión general como el acceso a detalles técnicos específicos:

* La Introducción proporciona el contexto, objetivos y alcance del proyecto, ofreciendo una visión general clara sobre la finalidad del trabajo y los estándares empleados.
* La Descripción General detalla el entorno en el que se desarrolla el producto, sus funcionalidades básicas, el perfil de los usuarios previstos, así como las restricciones y dependencias que condicionan su uso y evolución.
* La sección de Requisitos Específicos profundiza en las funcionalidades concretas que el sistema debe cumplir, abarcando interfaces, procesos, requisitos de rendimiento, atributos de calidad y limitaciones de diseño.
* Finalmente, los Apéndices contienen información complementaria y soporte técnico relevante, como fragmentos de código, instrucciones de instalación y configuración, que facilitan la replicabilidad y mantenimiento del proyecto.

La organización busca equilibrar la profundidad técnica con la accesibilidad, siendo útil tanto para desarrolladores y administradores, como para evaluadores académicos y profesionales del área.

# **2. Descripción General**

## **2.1 Perspectiva del Producto**

El producto desarrollado es una aplicación independiente que se ejecuta en un entorno Linux (Ubuntu) mediante línea de comandos, orientada a la captura y validación de datos para métodos de pago, simulando procesos comunes en aplicaciones comerciales y financieras. Esta independencia tecnológica y operativa le confiere portabilidad y flexibilidad para ser adaptado a otros sistemas o integrarse a soluciones mayores.BEl proyecto está concebido como una herramienta educativa que integra tres áreas fundamentales: programación en Java, administración de sistemas operativos y conceptos básicos de redes. Esta integración facilita una comprensión multidisciplinaria, permitiendo al usuario no solo programar, sino también configurar entornos remotos seguros y verificar la conectividad básica entre equipos. Desde el punto de vista arquitectónico, el producto está diseñado con modularidad y claridad, implementando cada funcionalidad como métodos separados y evitando dependencias externas o tecnologías complejas. Esto facilita su mantenimiento, ampliación y reutilización en futuros proyectos o contextos más avanzados. El producto no cuenta con interfaz gráfica ni almacenamiento persistente, limitándose a la ejecución local en consola y la interacción directa con el usuario. Sin embargo, esta simplicidad es intencional, priorizando la comprensión y dominio de conceptos fundamentales que sirven de base para desarrollos posteriores más complejos.

## **2.2 Funciones del Producto**

El sistema realiza un conjunto definido de funciones que cumplen con el propósito de simular la captura y procesamiento básico de métodos de pago de forma segura y sencilla. Estas funciones se describen a continuación de manera detallada y secuencial:

* Presentación del menú de métodos de pago: El sistema despliega un menú claro y sencillo con opciones enumeradas para que el usuario seleccione el método de pago deseado, validando que la opción elegida sea válida y, en caso contrario, solicitando reingreso sin interrumpir la ejecución.
* Captura de datos del método de pago: Dependiendo de la opción seleccionada, el sistema solicita información específica como nombre del titular, últimos dígitos de la tarjeta, mes y año de expiración y código CVV. Se incluyen validaciones para asegurar que los datos tengan un formato correcto y estén dentro de rangos aceptables, evitando la captura de datos erróneos o malformados.
* Validación y procesamiento interno\*\*: La aplicación realiza comprobaciones básicas para verificar que la información ingresada sea coherente (por ejemplo, que el mes esté entre 1 y 12, que el año no sea anterior al actual), y gestiona internamente estos datos evitando la exposición de información sensible o almacenamiento no autorizado.
* Visualización de resumen seguro: El sistema muestra un resumen con los datos relevantes para el usuario, ocultando información sensible como los dígitos completos de la tarjeta o el código CVV, garantizando así la privacidad y seguridad del usuario durante la interacción.
* Gestión modular de funciones: Cada funcionalidad está implementada como un método o procedimiento independiente, facilitando la legibilidad del código, su mantenimiento y futuras ampliaciones sin afectar el conjunto global del sistema.
* Control de errores y robustez: Se implementan mecanismos que permiten al sistema manejar entradas inválidas o errores de usuario sin que se produzcan fallas o cierres abruptos, mostrando mensajes claros y solicitando correcciones oportunas.

Estas funciones configuran un flujo operativo sencillo pero completo, que asegura una experiencia de usuario coherente y un código bien estructurado acorde con buenas prácticas de desarrollo.

## **2.3 Características de los Usuarios**

Los usuarios previstos para este sistema son principalmente estudiantes y desarrolladores en formación interesados en adquirir habilidades prácticas en programación en Java, administración de sistemas Linux y fundamentos básicos de redes. Estos usuarios poseen conocimientos elementales a intermedios en programación, manejo de terminal y conceptos básicos de redes, y buscan consolidar estos conocimientos mediante la aplicación práctica en un entorno controlado y realista. Asimismo, el sistema está orientado para ser utilizado por docentes y evaluadores que requieren una herramienta clara y estructurada para enseñar, demostrar y evaluar competencias técnicas relacionadas con la ingeniería de software, administración de sistemas y comunicación en redes. Los usuarios valoran la simplicidad y claridad en la interacción, así como la modularidad y documentación del código, que les permite entender fácilmente la estructura del programa y modificarlo para experimentar con nuevas funcionalidades. Este perfil de usuario también implica que el sistema debe ser tolerante a errores y brindar mensajes claros para facilitar el aprendizaje, evitando sobrecargar con complejidades innecesarias y manteniendo un enfoque educativo práctico y efectivo.

## **2.4 Restricciones**

El desarrollo y operación del sistema están sujetos a una serie de restricciones técnicas y funcionales que delimitan su diseño y uso, buscando mantener la simplicidad y efectividad del proyecto:

* Interacción exclusivamente en consola: No se implementó interfaz gráfica, por lo que todas las operaciones deben realizarse mediante terminal o consola, limitando la experiencia visual pero facilitando la portabilidad y compatibilidad.
* Ausencia de almacenamiento persistente: No se utiliza ninguna base de datos ni archivos para guardar información, por lo que todos los datos son temporales y se manejan exclusivamente en memoria durante la ejecución del programa.
* Dependencia del entorno Java y Linux: El sistema requiere que el JDK esté correctamente instalado y configurado en Ubuntu, condicionando su funcionamiento a la disponibilidad y configuración adecuada de este entorno.
* Seguridad limitada: Al tratarse de un proyecto académico, no se aplican mecanismos avanzados de cifrado ni autenticación, por lo que la seguridad de los datos depende principalmente de la correcta interacción del usuario y del entorno cerrado donde se ejecuta.
* Red limitada a verificación básica: Las funcionalidades de red se reducen a la comprobación de conectividad mediante el comando ping y acceso remoto seguro vía X2Go, sin implementación de protocolos o servicios de comunicación complejos.
* Recursos limitados del servidor: El entorno de desarrollo dispone de recursos hardware limitados, lo que puede impactar el rendimiento bajo cargas elevadas o escenarios más complejos.

Estas restricciones aseguran que el proyecto se mantenga en un ámbito controlado, propicio para el aprendizaje y la práctica sin complicaciones innecesarias.

## **2.5 Suposiciones y Dependencias**

Para garantizar el correcto desarrollo y operación del sistema, se asumen ciertas condiciones y dependencias que el entorno y los usuarios deben cumplir:

* Conectividad estable para acceso remoto: Se asume que los usuarios cuentan con una conexión de red confiable para utilizar X2Go y acceder remotamente al entorno gráfico del servidor Ubuntu.
* Instalación correcta de Java: Se requiere que la versión del JDK esté correctamente instalada y configurada en el sistema operativo, permitiendo la compilación y ejecución sin errores.
* Dominio básico de comandos de terminal: Se supone que los usuarios tienen conocimientos mínimos para operar la terminal Linux, incluyendo compilación de código Java, navegación de directorios y ejecución de comandos básicos.
* Acceso y permisos en GitHub: El repositorio del proyecto debe ser accesible y contar con permisos adecuados para clonar, actualizar y gestionar versiones del código fuente.
* Conocimientos fundamentales de programación: Se espera que los usuarios comprendan conceptos básicos de programación estructurada, uso de variables, condicionales y métodos en Java, facilitando la comprensión y modificación del código.

Estas suposiciones permiten enfocar el proyecto en aspectos técnicos sin abordar problemas derivados de configuraciones incorrectas o falta de habilidades básicas.

## **2.6 Requisitos Futuros**

El proyecto está concebido con una visión de evolución y ampliación que contempla múltiples áreas de mejora para incrementar su funcionalidad, usabilidad y robustez:

* Implementación de interfaces gráficas: Añadir interfaces gráficas utilizando tecnologías como Java Swing o JavaFX para mejorar la experiencia del usuario y facilitar la interacción.
* Integración de bases de datos: Incorporar sistemas de almacenamiento persistente mediante bases de datos SQL o NoSQL para gestionar usuarios, métodos de pago y registros históricos.
* Comunicación en red avanzada: Desarrollar funcionalidades cliente-servidor que permitan la interacción en red mediante sockets o servicios web REST, ampliando el alcance del sistema.
* Seguridad reforzada: Implementar mecanismos de cifrado, autenticación y validación avanzada para proteger la información sensible y garantizar la integridad de las transacciones.
* Portabilidad multiplataforma mejorada: Automatizar despliegues y configuraciones para que el sistema funcione de forma nativa en diferentes sistemas operativos con mínima intervención.
* Automatización de pruebas: Desarrollar pruebas unitarias, de integración y funcionales automatizadas para asegurar la calidad y estabilidad del software en el tiempo.
* Documentación continua: Mantener la documentación actualizada y enriquecida, incluyendo manuales de usuario, guías de instalación y planes de mantenimiento.

Estas mejoras plantean un camino claro para la evolución del sistema hacia aplicaciones más complejas y profesionales, facilitando su adopción en contextos reales y comerciales.

# **3. Requisitos Específicos**

## **3.1 Interfaces Externas**

El sistema interactúa con diversos componentes externos, cuya correcta configuración es fundamental para garantizar un funcionamiento adecuado y seguro:

* Interfaz de usuario (UI): La interacción se realiza a través de la consola de comandos del sistema operativo Ubuntu, accesible localmente o mediante sesiones remotas X2Go con entorno gráfico XFCE, que proporcionan una experiencia ligera y eficiente.
* Herramientas de desarrollo y gestión: El entorno requiere la instalación y configuración del JDK para la compilación y ejecución del código Java, así como el uso de Git y GitHub para el control de versiones, permitiendo colaboración y seguimiento del proyecto.
* Servicios de red y seguridad: El servidor debe permitir conexiones SSH mediante el puerto 22, con firewalls configurados para garantizar un acceso seguro, además del uso de herramientas como ping para verificación de conectividad.
* Clientes remotos: Los usuarios pueden acceder al servidor desde diferentes sistemas operativos (Windows, Linux, macOS) utilizando clientes compatibles con X2Go, asegurando flexibilidad y accesibilidad.
* Utilidades de red básicas: Se emplean comandos estándar del sistema Linux para tareas como comprobación de red, gestión de usuarios y configuración del entorno.

Estas interfaces garantizan un entorno integrado que soporta las necesidades funcionales y operativas del sistema.

## **3.2 Funciones**

El sistema debe cumplir con las siguientes funciones operativas y técnicas, detalladas para asegurar claridad en su desarrollo e implementación:

1. Mostrar menú interactivo para selección del método de pago:

* Desplegar opciones numeradas para que el usuario seleccione el método deseado.
* Validar que la opción seleccionada corresponda a un valor válido, solicitando reingreso en caso contrario.
* Permitir la opción de salida del programa de forma controlada.

2. Capturar datos específicos según método de pago:

* Solicitar nombre completo del titular de la cuenta o tarjeta.
* Capturar últimos cuatro dígitos del número de tarjeta, asegurando que sean numéricos y de longitud correcta.
* Pedir mes y año de expiración, validando rangos apropiados (mes 1-12, año no menor al actual).
* Solicitar código CVV, validando formato y longitud adecuada.

3. Validar entradas y controlar errores:

* Comprobar que cada dato cumple con el formato y rango esperado.
* En caso de error, mostrar mensajes claros y solicitar corrección sin terminar abruptamente la ejecución.

4. Mostrar resumen seguro de la información:

* Presentar el nombre del titular y tipo de método de pago seleccionado.
* Mostrar número de tarjeta parcialmente oculto, revelando solo los últimos cuatro dígitos.
* Mostrar fecha de expiración en formato legible.
* Ocultar el código CVV para proteger la privacidad del usuario.

5. Permitir repetir operaciones o salir:

* Ofrecer al usuario la posibilidad de realizar una nueva captura o finalizar el programa.

6. Estructura modular y código limpio:

* Implementar cada función en métodos separados, con nombres descriptivos y comentarios explicativos, facilitando mantenimiento y escalabilidad.
* Estas funciones configuran el comportamiento completo del sistema, asegurando funcionalidad, usabilidad y robustez.

7. cw.java

* Clase principal del programa. Gestiona el flujo inicial de ejecución, mostrando el menú para registrar usuarios, iniciar sesión o salir.
* Determina si el usuario es administrador o cliente normal y redirige a Admin.java o Reproductor.java según el caso.
* También conecta con Suscripcion.java para manejo de pagos y membresías.

8. Suscripcion.java

* Encargada de procesar pagos y administrar las suscripciones de los usuarios.
* Controla fechas de inicio y fin de membresía, tipos de suscripción, métodos de pago y puntos asociados.
* Valida si un usuario tiene acceso activo al contenido.

9. Reproductor.java

* Maneja la reproducción de contenido. Presenta opciones de listas, reproduce elementos y administra puntos o recompensas para el usuario.
* Regresa al menú principal después de finalizar la reproducción.

10. Admin.java

* Proporciona el menú de administrador para gestionar usuarios. Permite ver todos los registros de usuarios y eliminar cuentas, exceptuando el administrador principal. Lee y escribe datos en el archivo 'usuarios.txt'

## **3.3 Requisitos de Rendimiento**

Para garantizar una experiencia de usuario óptima y un funcionamiento eficiente, el sistema debe cumplir con ciertos parámetros de rendimiento:

* La respuesta a las entradas del usuario debe ser inmediata, sin retardos perceptibles, asegurando una interacción fluida y sin interrupciones.
* La compilación y ejecución del programa Java desde terminal deben completarse en menos de un segundo en entornos con recursos mínimos, optimizando el ciclo de desarrollo y pruebas.
* El acceso remoto mediante X2Go debe mantener latencias bajas, garantizando que las acciones del usuario se reflejen rápidamente en el entorno remoto para evitar frustración o errores por retrasos.
* El consumo de recursos del sistema, incluyendo CPU y memoria RAM, debe ser mínimo y eficiente, evitando que la aplicación afecte el rendimiento general del servidor o cause bloqueos.

Estos requisitos aseguran que el sistema sea práctico para su uso en entornos educativos y de desarrollo, manteniendo una buena experiencia y estabilidad operativa.

## **3.4 Restricciones de Diseño**

El diseño del sistema debe respetar ciertas restricciones que garantizan coherencia, mantenibilidad y cumplimiento de los objetivos del proyecto:

* La aplicación debe ser modular, con funciones implementadas en métodos independientes, usando nomenclatura clara y consistente, y código comentado para facilitar la comprensión.
* Se debe emplear exclusivamente Java estándar, evitando librerías externas o dependencias que compliquen la instalación o ejecución.
* No se debe implementar interfaz gráfica, limitando la interacción a la consola para mantener la simplicidad y enfoque académico.
* Las validaciones deben asegurar la integridad y coherencia de los datos, evitando errores que provoquen fallos o comportamientos inesperados.
* El código debe ser portable y funcionar en cualquier distribución Linux que cuente con JDK instalado y configurado adecuadamente.
* Se debe evitar el almacenamiento persistente, garantizando que la información sensible no quede registrada en disco o sistemas externos.

Estas restricciones promueven la calidad y robustez del desarrollo, facilitando la adaptación y uso en diferentes contextos.

## **3.5 Atributos del Sistema**

El sistema desarrollado presenta atributos clave que definen su calidad y valor para los usuarios y administradores:

* Portabilidad: Puede ejecutarse en cualquier sistema Linux con Java configurado, permitiendo su uso en diversos entornos sin modificaciones significativas.
* Seguridad: Protege la privacidad del usuario al no almacenar ni mostrar datos completos sensibles, siguiendo buenas prácticas para evitar filtraciones.
* Usabilidad: Ofrece una interacción sencilla y clara mediante mensajes comprensibles y un flujo lógico, facilitando su uso por usuarios con conocimientos básicos.
* Mantenibilidad: Su código está organizado, comentado y estructurado en módulos, facilitando futuras modificaciones y ampliaciones sin afectar la estabilidad.
* Fiabilidad: Maneja entradas erróneas de forma controlada, evitando cierres inesperados o errores críticos durante la ejecución.
* Escalabilidad: Su arquitectura modular permite incorporar nuevas funcionalidades y mejoras de forma ordenada y eficiente.

Estos atributos aseguran que el sistema sea una herramienta confiable, accesible y durable para fines educativos y de desarrollo.

## **3.6 Otros Requisitos**

Además de los requisitos funcionales y técnicos, el proyecto debe cumplir con aspectos adicionales importantes para su correcta implementación y uso:

* La documentación debe seguir estándares internacionales (IEEE 830), ofreciendo claridad, orden y precisión para facilitar la comprensión y evaluación.
* El repositorio GitHub debe contener el código fuente completo, junto con un historial de versiones, archivos de configuración y documentación asociada, permitiendo la colaboración y seguimiento.
* Se debe garantizar que el sistema permita acceso remoto seguro y eficiente mediante herramientas como X2Go, asegurando que los usuarios puedan operar y administrar el entorno sin complicaciones.
* Los usuarios deben disponer de guías o tutoriales básicos para la instalación y configuración del entorno, facilitando la puesta en marcha del sistema.
* Se recomienda mantener prácticas de seguridad en el servidor Ubuntu, como configuración de firewalls y actualizaciones, para proteger el entorno donde se ejecuta la aplicación.

Estos aspectos complementan la funcionalidad técnica, asegurando una experiencia integral y profesional.

# **4. Apéndices**

Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.

Ilustración 1. DIagrama de flujo inicio de programa

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración 2. diagrama de flujo Inicio (C.W)

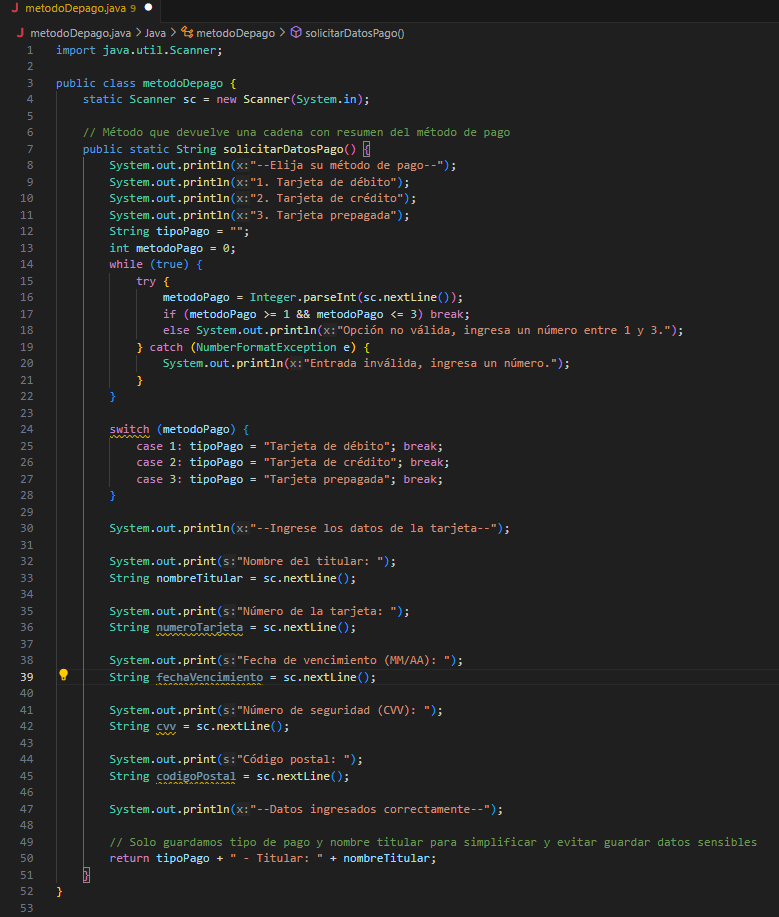


Ilustración 3. Código de método de pago



Ilustración 4. Código de pago

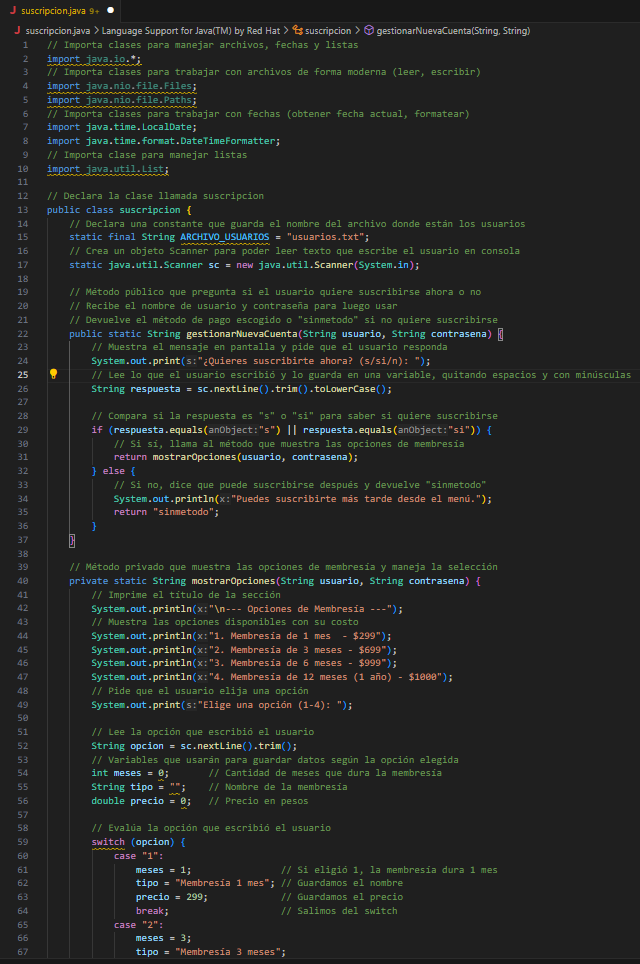


Ilustración 5. Código de suscripción parte 1



Ilustración 6. Código de suscripción parte 2

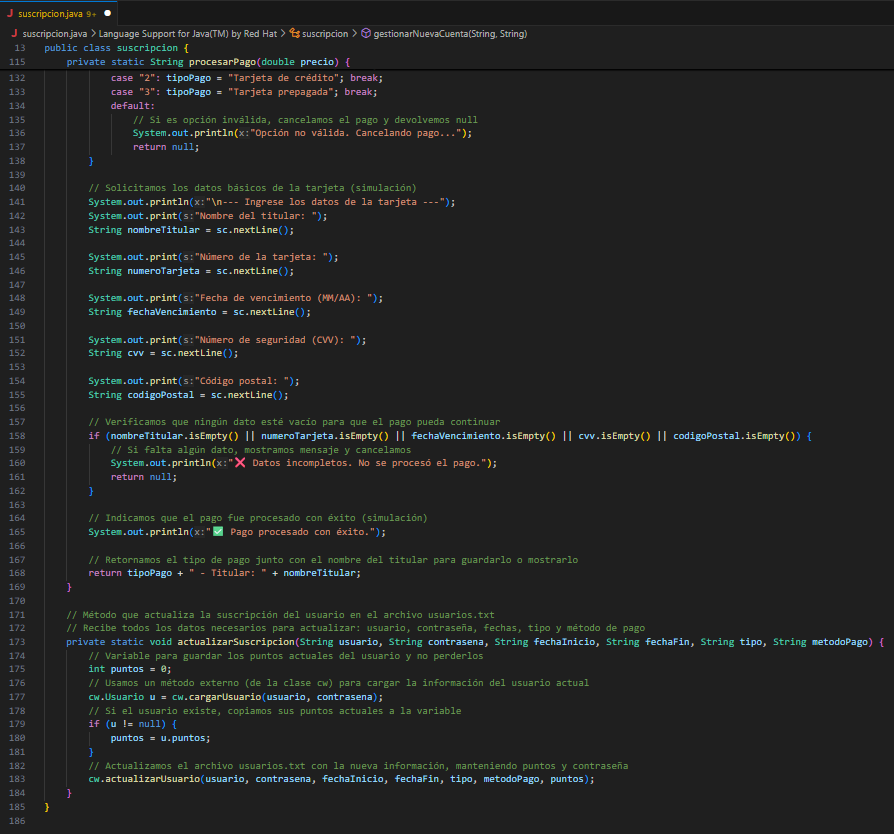


Ilustración 7. Código de suscripción parte 3



Ilustración 8. Código de Cinema World parte 1



Ilustración 9.Código de Cinema World parte 2



Ilustración 10.Código de Cinema World parte 3



**Ilustración 11. Código de Cinema World parte 4**

# **5. Conclusión**

En conclusión, este documento ha presentado una descripción completa y detallada del proyecto de programación desarrollado en Java sobre un entorno Ubuntu accesible remotamente, siguiendo los lineamientos y estándares establecidos por el IEEE 830 para especificación de requisitos de software. La aplicación desarrollada representa una integración efectiva de conceptos fundamentales de programación, administración de sistemas operativos y redes, enfocándose en la captura y validación segura de datos relacionados con métodos de pago, dentro de un contexto educativo y profesional. Se ha detallado el propósito del documento como una guía tanto para desarrolladores como para evaluadores, estableciendo un marco sólido para la comprensión, implementación y mantenimiento del software, con una arquitectura modular que facilita futuras ampliaciones. El ámbito del sistema queda definido como una aplicación de consola, portable y sencilla, que no depende de bases de datos ni interfaces gráficas, enfocándose en el manejo correcto de datos y la interacción segura con el usuario. Además, se identificaron y describieron en detalle los requisitos funcionales y no funcionales, las restricciones de diseño, las características de los usuarios previstos, y las condiciones de entorno necesarias para su funcionamiento, junto con un análisis de las posibles mejoras y ampliaciones que pueden implementarse para convertir el proyecto en una solución más robusta y completa. Finalmente, se establecieron aspectos técnicos complementarios, como la configuración de herramientas externas (X2Go, GitHub), protocolos de red básicos, y buenas prácticas de seguridad y mantenimiento del entorno, asegurando que el sistema pueda ser utilizado y evaluado de manera confiable y efectiva. Este trabajo, por tanto, constituye una base sólida para la formación y desarrollo profesional en las áreas de programación, administración de sistemas y redes, brindando un recurso integral y bien documentado que puede ser adaptado y expandido para múltiples propósitos y escenarios futuros.

# **6. Referencias apa**

1. Canonical Ltd. (s.f.). Documentación oficial de Ubuntu. Recuperado de <https://help.ubuntu.com/>
2. Comunidad Ubuntu España. (s.f.). Documentación en español de Ubuntu. Recuperado de <https://wiki.ubuntu.com/SpanishDocumentation>
3. Oracle. (s.f.). Java: Tecnología para el desarrollo de aplicaciones. Recuperado de <https://www.java.com/es/>
4. JavaHispano. (s.f.). Documentación y tutoriales de Java. Recuperado de <https://www.javahispano.org/documentacion/>
5. Red Hat, Inc. (s.f.). Documentación de Red Hat Enterprise Linux 9. Recuperado de <https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/9/>
6. Mozilla Developer Network. (s.f.). Guía de seguridad en redes. Recuperado de <https://developer.mozilla.org/es/docs/Web/Security>
7. Cisco Systems, Inc. (s.f.). Fundamentos de redes. Recuperado de <https://www.cisco.com/c/es_mx/solutions/small-business/resource-center/networking-basics.html>
8. Microsoft Docs. (s.f.). Conceptos básicos de redes. Recuperado de <https://learn.microsoft.com/es-es/windows-server/networking/>
9. Instituto Nacional de Ciberseguridad (INCIBE). (s.f.). Guías y recursos sobre seguridad en redes. Recuperado de <https://www.incibe.es/protege-tu-empresa/guias-y-recursos>
10. Aulaclic. (s.f.). Curso básico de redes informáticas. Recuperado de https://www.aulaclic.es/redes/index.htm