

Departamento de Ciencias de la Computación (DCCO)

Carrera de Software

A&D Sw

Perfil del Proyecto

Presentado por: Grupo 3

Tutor académico: Ing. Jenny A Ruiz R

Ciudad: Sangolquí

Fecha: 11/12/2025

Tabla de contenido

Introducción	5
Planteamiento del trabajo	5
2.1 Formulación del problema	5
2.2 Justificación.....	6
Sistema de Objetivos	7
3.1. Objetivo General	7
3.2. Objetivos Específicos (03)	7
Alcance.....	7
Marco Teórico.....	8
1. IDEs y Herramientas de Desarrollo Utilizadas	8
1.1 Visual Studio Code (VS Code)	8
1.2 Docker	9
1.3 Git y GitHub/GitLab.....	9
1.4 PostgreSQL	9
1.5 OCR Tools (Tesseract / Vision APIs)	9
2. Patrones de Diseño Aplicados en el Proyecto	9
2.1 Patrones GoF (Gang of Four)	10
2.2 Procesamiento de Imágenes y OCR en Sistemas Financieros.....	10
3. Accesibilidad y Usabilidad	10
5.1 Metodología (Marco de trabajo 5W+2H)	11

Ideas a Defender	11
Justificación de la prioridad	12
Planificación Detallada del Proyecto.....	12
1. Fase de Análisis	12
2. Fase de Priorización	13
3. Fase de Diseño	14
IDEs y Herramientas de Desarrollo Seleccionadas.....	15
Visual Studio Code	15
Docker.....	15
Git + GitHub/GitLab	16
Patrones de Diseño Aplicados.....	16
Resultados Esperados.....	17
Viabilidad.....	19
8.1 Humana	20
8.1.1 Tutor Empresarial	20
8.1.2 Tutor Académico.....	21
8.1.3 Estudiantes	21
8.2 Tecnológica.....	21
8.2.1 Hardware	22
8.2.2 Software	23
Conclusiones y recomendaciones.....	24

9.1 Conclusiones.....	24
9.2 Recomendaciones.....	25
Planificación para el Cronograma	25
Referencias	26

1. Introducción

En la actualidad, la empresa AGROTAC enfrenta un problema crítico relacionado con la gestión manual y no estandarizada de comprobantes de pago. El proceso actual se realiza mediante el envío de fotografías por WhatsApp y posterior registro en hojas de Excel, lo que ha provocado duplicaciones, pérdida de información, errores humanos y falta de trazabilidad.

Esta situación afecta directamente a la administración de créditos, la revisión de pagos y la toma de decisiones gerenciales. Debido a esta necesidad evidente, surge el proyecto *PagoSeguroAGROTAC*, cuyo propósito es diseñar y desarrollar un sistema informático moderno, automatizado y seguro que resuelva las deficiencias actuales.

El proyecto se orienta a mejorar el control, la eficiencia y la transparencia de la gestión de pagos mediante una plataforma centralizada que automatice procesos críticos. Su diseño se fundamenta en principios de ingeniería de software, análisis de requisitos, validación de funcionalidades y aplicación de matrices de planificación.

2. Planteamiento del trabajo

2.1 Formulación del problema

La empresa no cuenta con un sistema confiable que gestione de forma automatizada los comprobantes de pago. La administración mediante WhatsApp y Excel provoca:

- Falta de control sobre duplicados.
- Errores en registros manuales.
- Pérdida de trazabilidad histórica.
- Demoras en la validación de pagos.
- Ausencia de notificaciones y alertas.

La solución planteada consiste en el desarrollo de *PagoSeguroAGROTAC*, un sistema informático capaz de recibir comprobantes directamente del cliente, validar la información, detectar duplicados, notificar resultados y generar reportes administrativos. El sistema gestionará créditos, procesará pagos, visualizará morosidad y permitirá la toma de decisiones basada en datos.

2.2 Justificación

La implementación del sistema tiene un impacto significativo en la eficiencia operativa. Al reemplazar procesos manuales por flujos digitales:

Se garantiza la integridad de los datos registrados.

- Se reduce drásticamente la duplicación o pérdida de comprobantes.
- Se libera al personal administrativo de tareas repetitivas.
- Se obtiene trazabilidad completa del ciclo de vida de cada pago.
- Se mejora el monitoreo de créditos y la detección de morosidad.
- Se fortalece la toma de decisiones mediante reportes confiables.

El proyecto se justifica plenamente tanto en términos técnicos como empresariales, ya que agrega valor directo a la administración financiera y mejora la experiencia de clientes y personal interno.

3. Sistema de Objetivos

3.1. Objetivo General

Automatizar la gestión de comprobantes y créditos mediante el desarrollo del sistema PagoSeguroAGROTAC, garantizando trazabilidad, minimización de errores y optimización de reportes administrativos.

3.2. Objetivos Específicos (03)

1. Implementar un sistema seguro que valide comprobantes, detecta duplicados y registre transacciones sin errores.
2. Diseñar módulos de reportes administrativos y financieros que apoyen la toma de decisiones gerenciales.
3. Desarrollar un flujo automatizado de notificaciones (vencimientos, morosidad, pagos confirmados) para mejorar la comunicación con los clientes.

4. Alcance

El sistema permitirá ejecutar las siguientes funcionalidades:

- Registro e inicio de sesión por roles (Gerente, Asistente, Cliente).
- Registro y validación automatizada de comprobantes de pago.

- Gestión de créditos: otorgamiento, consulta y seguimiento.
- Pagos en línea, pagos por transferencia y carga de comprobantes.
- Generación de reportes por cliente, por fechas y de morosidad.
- Importación de datos desde Excel o CSV.
- Emisión de notificaciones automáticas por vencimientos.
- Generación de certificados en PDF.
- Revisión integral del estado de cuenta del cliente.

5. Marco Teórico

El desarrollo del sistema PagoSeguroAGROTAC requiere una base conceptual sólida que abarque herramientas de programación modernas, entornos de desarrollo integrados (IDEs), metodologías de diseño de software, patrones arquitectónicos, procesamientos avanzados de documentos y mecanismos de integración. A continuación se detallan los fundamentos teóricos necesarios para comprender la construcción del sistema.

1. IDEs y Herramientas de Desarrollo Utilizadas

El proyecto utiliza herramientas de propósito profesional que permiten construir, depurar, versionar y desplegar aplicaciones de software de manera eficiente.

1.1 Visual Studio Code (VS Code)

Visual Studio Code es un IDE ligero, multiplataforma y extensible, ampliamente utilizado en proyectos web y backend. Permite integrar herramientas como Docker, Git, linters, debuggers y extensiones para JavaScript, TypeScript y Node.js, facilitando la construcción del sistema PagoSeguroAGROTAC.

VS Code es considerado uno de los entornos más versátiles debido a su arquitectura basada en extensiones y su integración natural con Git y contenedores [1].

1.2 Docker

Docker permite empaquetar la aplicación dentro de contenedores reproducibles que incluyen todas las dependencias del sistema. Gracias a esto, el entorno de desarrollo y producción se mantiene consistente, evitando errores derivados de configuraciones distintas.

Docker es una herramienta estándar en la industria moderna para el despliegue de microservicios y APIs [2].

1.3 Git y GitHub/GitLab

Para control de versiones se emplea Git, lo cual facilita el trabajo colaborativo, el manejo de ramas, la revisión de código y el despliegue automatizado mediante CI/CD. El control de versiones es fundamental para asegurar trazabilidad y auditoría del proyecto [3].

1.4 PostgreSQL

PostgreSQL se utiliza como sistema de gestión de bases de datos por su robustez, conformidad con ACID, extensibilidad y soporte para transacciones complejas. Es altamente recomendado para sistemas de pagos o de validación donde la consistencia es prioritaria [4].

1.5 OCR Tools (Tesseract / Vision APIs)

Para el reconocimiento automático de comprobantes, el sistema puede integrar motores OCR como Tesseract o servicios avanzados basados en IA. Estos motores permiten extraer texto desde imágenes e interpretar campos como fecha, valor y número de referencia [5][6].

2. Patrones de Diseño Aplicados en el Proyecto

Los patrones de diseño son soluciones probadas para problemas recurrentes en el desarrollo de software. El sistema utiliza patrones GoF y patrones arquitectónicos.

2.1 Patrones GoF (Gang of Four)

- Strategy: se usa para encapsular diferentes formas de validar comprobantes (validación estricta, OCR avanzado, validación manual).
- Factory Method / Abstract Factory: permite crear procesadores de comprobantes según su tipo (físico, digital, PDF, imagen) [7].
- Observer: útil en la notificación automática cuando un comprobante es procesado o rechazado.
- Singleton: en casos controlados, por ejemplo para configuraciones globales de la aplicación [7].

2.2 Procesamiento de Imágenes y OCR en Sistemas Financieros

El OCR (Optical Character Recognition) es crucial para automatizar la lectura de comprobantes.

Según Yu et al., los modelos actuales combinan CNNs, Transformers y alineación multimodal para mejorar el reconocimiento de documentos financieros [5].

El dataset CORU [6] demuestra que la combinación OCR + NLP aumenta significativamente la precisión en la extracción de datos de recibos.

3. Accesibilidad y Usabilidad

El sistema se guía por las pautas de accesibilidad WCAG 2.1, que establecen estándares para asegurar que el software pueda ser usado por personas con limitaciones cognitivas, motrices o visuales [8].

5.1 Metodología (Marco de trabajo 5W+2H)

¿QUÉ?	¿CÓMO?	¿QUIÉN?	¿CUÁNDΟ?	¿POR QUÉ?
Desarrollo del sistema PagoSeguroAGROTAC: módulo de registro, validación de comprobantes, notificaciones y reportes.	Metodología ágil: backlog en Excel, priorización por MoSCoW, descomposición en historias, Sprint planning, revisión y retrospectiva. Integración continua y pruebas automatizadas.	Equipo de proyecto: 3 Gerente de proyecto/tutor empresarial, tutor académico, estudiantes desarrolladores.	Planificado por sprints: Sprint 0 (configuración) 1 semana, Sprint 1 (1 semana cada uno).	Para transformar un proceso manual en uno automatizado, confiable y auditable que mejore la gestión de pagos.

Tabla.1 Marco de trabajo 5W+2H

6. Ideas a Defender

6. Ideas a Defender

Durante el proceso de análisis del sistema, se identificó que la correcta operación de la plataforma depende de la interacción coordinada de múltiples módulos funcionales, los cuales abarcan desde la autenticación y gestión de usuarios hasta la administración de créditos, pagos, contenidos, reportes y auditoría. Para asegurar una implementación ordenada y alineada con los objetivos del sistema, se aplicó un proceso de priorización de requisitos basado en criterios técnicos y de negocio.

Mediante una matriz de priorización fundamentada en la criticidad del requisito, su dependencia con otros módulos, el impacto directo en el negocio y la frecuencia de uso por parte de los actores del sistema, se determinó que el **núcleo funcional del sistema** está compuesto por los módulos de acceso, gestión de usuarios y control transaccional.

Como resultado de este análisis, se definió como **Requisito Prioritario Absoluto** el siguiente:

Requisito Prioritario Absoluto: RF-1 – Gestionar Acceso al Sistema,

ya que constituye el punto de entrada obligatorio para todos los actores del sistema y habilita el uso del resto de funcionalidades.

Por consecuencia directa, también se priorizaron sus subrequisitos:

- RF-1.1 Registrar Cliente
- RF-1.2 Registrar Asistente
- RF-1.3 Iniciar Sesión
- RF-1.4 Recuperar Contraseña
- RF-1.5 Cerrar Sesión
- RF-1.6 Acceder al Dashboard

Adicionalmente, y considerando la planificación del backlog y el alcance definido para el proyecto, se priorizaron los siguientes requisitos funcionales complementarios, los cuales dependen del correcto funcionamiento del módulo de acceso:

- **RF-002 – Gestión de Perfil de Usuario**
- **RF-005 – Gestión de Usuarios**
- **RF-006 – Gestión de Créditos**
- **RF-007 – Gestión de Pagos**
- **RF-008 – Gestión de Contenidos del Sistema**
- **RF-009 – Generación de Reportes**
- **RF-010 – Gestión de Notificaciones**
- **RF-012 – Auditoría del Sistema**
- **RF-013 – Configuración General del Sistema**

Estos requisitos fueron considerados dentro del backlog del proyecto al representar funcionalidades necesarias para garantizar la operatividad, control, trazabilidad y administración del sistema en un entorno real.

6.1 Justificación de la Prioridad

La priorización de los requisitos seleccionados se sustenta en las siguientes razones técnicas y funcionales:

- Todo usuario debe autenticarse obligatoriamente para acceder a funcionalidades como créditos, pagos, contenidos y reportes.
- La gestión de perfil y usuarios (RF-002 y RF-005) depende directamente de la identificación correcta del actor dentro del sistema.
- Los módulos de créditos y pagos (RF-006 y RF-007) involucran información sensible, lo que exige controles de acceso, trazabilidad y seguridad.
- La generación de reportes y auditoría (RF-009 y RF-012) requiere sesiones válidas y roles bien definidos para garantizar la integridad de la información.
- El sistema debe permitir el seguimiento de acciones realizadas por cada actor (Cliente, Asistente y Gerente), lo cual es posible únicamente mediante un control de acceso robusto.
- Los requisitos no funcionales asociados a seguridad, integridad y monitoreo (RNF-04, RNF-06 y RNF-09) dependen directamente del correcto manejo de sesiones y roles.

Esta priorización permitió asegurar que el sistema cuente con una base sólida antes de implementar funcionalidades más avanzadas, reduciendo riesgos técnicos y facilitando el control del desarrollo incremental.

6.2 Planificación Detallada del Proyecto

La planificación del proyecto se estructuró en fases, adoptando un enfoque incremental que permitió organizar el desarrollo de los requisitos priorizados de forma progresiva y controlada.

6.2.1 Fase de Análisis

Durante esta fase se llevaron a cabo las siguientes actividades:

- Estudio del proceso manual actual utilizado por la organización, basado principalmente en hojas de cálculo (Excel) y validaciones manuales.
- Identificación de los actores del sistema (Cliente, Asistente y Gerente) y definición de sus responsabilidades.
- Análisis de los procesos relacionados con gestión de usuarios, créditos, pagos, contenidos y reportes.
- Documentación de reglas de negocio asociadas a créditos, pagos, control de accesos y auditoría.
- Definición y validación de los requisitos funcionales (RF) y no funcionales (RNF) del sistema.

6.2.2 Fase de Priorización

En esta fase se aplicó una matriz de priorización basada en los criterios de valor para el negocio, riesgo asociado a la no implementación, dependencia con otros requisitos y esfuerzo estimado de desarrollo.

- El RF-1 se identificó como un “cuello de botella”, por lo que se definió su implementación como prioritaria.
- Los requisitos RF-002, RF-005 y RF-013 se priorizaron en una segunda capa al depender directamente del módulo de acceso.
- Los requisitos RF-006, RF-007, RF-008 y RF-010 se planificaron como funcionalidades operativas clave.
- Los módulos de reportes y auditoría (RF-009 y RF-012) se programaron como soporte de control y supervisión.

Este enfoque de priorización cuantitativa está alineado con prácticas recomendadas en la Ingeniería de Requisitos y respaldado por autores como Wiegers y Beatty [10] y Sommerville [11].

6.2.3 Fase de Diseño

La fase de diseño incluyó las siguientes actividades:

- Elaboración de casos de uso generales y específicos por actor.
- Diseño de diagramas UML para representar flujos, escenarios y relaciones entre módulos.
- Definición de una arquitectura por capas (presentación, lógica de negocio y datos).
- Selección de patrones de diseño adecuados al contexto del sistema.
- Diseño de la base de datos, contemplando entidades como usuarios, roles, créditos, pagos, contenidos, auditoría y configuraciones del sistema.

6.3 IDEs y Herramientas de Desarrollo Seleccionadas

La selección de herramientas responde a criterios de eficiencia, compatibilidad y estandarización.

6.3.1 Visual Studio Code

Elegido por:

- Extensiones profesionales para JavaScript, TypeScript, Node.js.
- Integración nativa con Git y Docker.
- Depurador integrado.
- Terminal interna para ejecutar el backend y scripts.
- Alta comunidad, documentación y soporte.
- Se convirtió en el IDE central para desarrollo frontend y backend.

6.3.2 Docker

Docker se seleccionó para:

- Ejecutar el backend, base de datos y servicios de manera aislada.
- Simular entornos de producción.
- Garantizar portabilidad del sistema en diferentes máquinas.

- Facilitar el despliegue mediante contenedores.
- Se diseñó una arquitectura donde cada servicio puede correr en su propio contenedor.

6.3.3 Git + GitHub/GitLab

Git se utilizó para:

- Controlar versiones de código.
- Trabajar colaborativamente con ramas por desarrollador.
- Revisar cambios (pull requests).
- Mantener historial y auditoría de desarrollo.
- Realizar despliegues automatizados.

6.1 Patrones de Diseño Aplicados

Para garantizar una arquitectura mantenible se aplicaron patrones reconocidos:

6.4.1 Strategy

- Permite cambiar dinámicamente el método de validación del comprobante:
- Validación por reglas de negocio
- Validación por estructura
- Validación por comparación de datos
- Reduce el acoplamiento y hace el sistema extensible.

6.4.2 Factory Method / Abstract Factory

- Fundamental para la creación de objetos cuando existen múltiples variantes de comprobantes.
- Permite que el sistema pueda extenderse fácilmente para soportar nuevos formatos.

6.4.3 Singleton

Aplicado para:

- Configuración global del proyecto.
- Parámetros de inicialización de la base de datos.
- Acceso único a ciertos servicios internos.
- Se evitó el uso excesivo para no limitar escalabilidad.

6.4.4 MVC y Arquitectura en Capas

Usada de forma general:

- Capa de presentación: React, páginas por rol.
- Capa de negocio: Validación, reglas, roles, sesiones.
- Capa de datos: PostgreSQL, repositorios, consultas.

Esto asegura la separación de responsabilidades y reduce errores.

7. Resultados Esperados

El desarrollo del sistema PagoSeguroAGROTAC permitirá obtener resultados concretos tanto en el ámbito técnico como en el proceso formativo dentro de la asignatura de Análisis y Diseño de Software. Los resultados esperados del proyecto son los siguientes:

7.1. Modelos de análisis correctamente elaborados

Se espera generar un conjunto completo y coherente de artefactos de análisis, incluyendo:

- especificación de requisitos funcionales y no funcionales
- casos de uso detallados
- modelo de dominio
- diccionario de datos
- backlog priorizado.

Estos artefactos deben reflejar fielmente las necesidades del negocio y servir como base formal para la etapa de diseño.

7.2. Integración del trabajo colaborativo con herramientas profesionales

Se espera que el equipo aplique metodologías y herramientas de trabajo colaborativo utilizadas en la industria:

- control de versiones con Git
- elaboración de ramas para diseño
- revisión de cambios (pull/merge requests)
- trazabilidad de decisiones.

Este resultado demuestra que el equipo comprendió los flujos de trabajo modernos en proyectos reales.

8. Viabilidad

La viabilidad del proyecto PagoSeguroAGROTAC se evalúa considerando los recursos económicos, humanos y tecnológicos necesarios para garantizar el desarrollo adecuado del sistema dentro del entorno académico. Esta sección demuestra que el proyecto puede ejecutarse con los recursos disponibles y que su implementación es factible tanto a nivel técnico como financiero. La siguiente tabla detalla los costos asociados al desarrollo del proyecto, considerando únicamente los recursos indispensables para las actividades de análisis, diseño y construcción inicial del sistema.

Cantidad	Descripción	Valor Unitario (USD)	Valor Total (USD)
	Equipo en casa		
1	Laptop ASUS RAYZEN 7 5500U / 8gb RAM / 256gb SSD	600	600
	Software		
1	Sistema operativo Windows 10	145	145
1	Visual Studio Code	0	0
1	Docker	0	0
1	FileZilla	0	0
1	Jira	0	0
	TOTAL		745

Tabla 2 Presupuesto del proyecto

El costo total del proyecto asciende a 745 USD, principalmente atribuible al equipo de cómputo. Las herramientas de software utilizadas son gratuitas o de código abierto, lo cual reduce significativamente los costos totales del proyecto.

La arquitectura de PagoSeguroAGROTAC utiliza tecnologías modernas como Docker, APIs REST, PostgreSQL y procesamiento de documentos. A pesar de ello, estas herramientas funcionan eficientemente en equipos de gama media.

Por lo tanto, el hardware disponible supera los requisitos mínimos y es totalmente adecuado para el desarrollo del sistema.

La arquitectura de PagoSeguroAGROTAC utiliza tecnologías modernas como Docker, APIs REST, PostgreSQL y procesamiento de documentos. A pesar de ello, estas herramientas funcionan eficientemente en equipos de gama media.

La laptop Ryzen 7 disponible ofrece capacidad suficiente para ejecutar simultáneamente:

- Contenedores Docker con 2–3 microservicios activos
- Procesos de OCR y validación de documentos.
- Servidor backend local ([Node.js](#)).
- Visual Studio Code y herramientas auxiliares.

8.1 Humana

8.1.1 Tutor Empresarial

Gerente Granito Agrotac

• **Responsabilidades**

- Validar que los procesos del sistema cumplan con la realidad operativa de la empresa.
- Proveer los flujos de negocio relacionados con créditos, pagos y revisión de comprobantes.
- Supervisar la correcta implementación de reglas financieras y políticas internas.
- Retroalimentar entregables del equipo técnico.

8.1.2 Tutor Académico

Ing. Jenny Ruiz

- **Responsabilidades**

- Supervisar la aplicación correcta de metodologías de análisis y diseño.
- Revisar artefactos: casos de uso, backlog, matriz 5W+2H, historias de usuario.
- Verificar que el sistema cumpla con los objetivos formativos de la asignatura.
- Evaluar sprints del equipo.

8.1.3 Estudiantes

- **Responsabilidades**

- Desarrollar la arquitectura propuesta (API REST, módulo OCR, deduplicación, reportes).
- Implementar patrones de diseño (Repository, Strategy, Observer).
- Documentar backlog, casos de uso, diagramas UML y evidencia técnica.
- Realizar despliegues usando Docker y control de versiones con Git.

El proyecto requiere habilidades combinadas de ingeniería de software, programación y análisis de requisitos, por lo que el equipo humano disponible permite implementarlo.

8.2 Tecnológica

8.2.1 Hardware

	Requisitos mínimos	Disponibilidad
Memoria RAM	4 GB de RAM	Alta
Almacenamiento	10 GB de espacio de almacenamiento	Alta

Tabla 3. Requisitos de Hardware

La arquitectura del sistema utiliza herramientas modernas (Docker, OCR, procesamiento de imágenes, APIs REST), pero estas funcionan eficientemente en equipos de gama media. La laptop indicada permite ejecutar simultáneamente:

- Docker con 2–3 microservicios
- Procesos OCR (Tesseract o derivados)
- IDE + servidor local

8.2.2 Software

	Requisitos mínimos	Disponibilidad
Sistema Operativo	Se recomienda Windows 10 u 11, macOS 10.10 o Ubuntu 16	Alta
IDE	Es recomendable Visual Studio Code debido a su conexión con FTP, sin embargo, cualquier IDE con esta funcionalidad funciona.	Alta
Contenedores	Docker Desktop	Alta

Tabla 4. Requisitos de Software

Las herramientas utilizadas son gratuitas o de libre acceso. La solución usa tecnologías estándar en la industria:

- Docker para despliegue y ambiente homogéneo.
- VS Code para desarrollo ágil con extensiones para Git, Docker y formato.
- PostgreSQL para persistencia robusta.

La laptop Ryzen 7 disponible ofrece capacidad suficiente para ejecutar simultáneamente:

- Contenedores Docker con 2–3 microservicios activos.
- Procesos de OCR y validación de documentos.
- Servidor backend local (Node.js).
- Visual Studio Code y herramientas auxiliares.

9. Conclusiones y recomendaciones

9.1 Conclusiones

- El proyecto permitió identificar claramente la problemática central de la empresa AGROTAC, ya que el análisis evidenció fallas significativas en la gestión manual de comprobantes, como duplicaciones, pérdida de información y falta de trazabilidad. Este diagnóstico sirvió como base para estructurar un análisis formal del sistema y demostrar la necesidad de una solución automatizada sustentada en principios de ingeniería de software.
- La fase de análisis permitió elaborar artefactos completos, tales como los requisitos funcionales y no funcionales, los casos de uso, el backlog y la matriz 5W+2H. Estos artefactos muestran que el equipo aplicó correctamente las técnicas enseñadas en la asignatura, especialmente al priorizar el requisito RF-1 como punto crítico del sistema, demostrando una comprensión del impacto que tiene la autenticación en el resto de funcionalidades.
- El diseño del sistema se desarrolló de manera estructurada mediante diagramas UML, modelo de dominio y arquitectura en capas, logrando organizar adecuadamente las responsabilidades del sistema. Asimismo, la selección de patrones como Strategy y Factory favoreció la extensibilidad, la modularidad y la reducción del acoplamiento, lo que evidencia el dominio de conceptos fundamentales de diseño de software.
- El uso de herramientas profesionales como VS Code, Docker y Git contribuyó significativamente a la calidad del proceso de diseño, ya que permitieron mantener control de versiones, registrar la evolución del análisis y asegurar la coherencia entre los diferentes artefactos del proyecto. Esto demuestra que el equipo adquirió competencias prácticas alineadas con las metodologías modernas de desarrollo.
- La evaluación de la viabilidad económica, humana y tecnológica confirmó que el proyecto puede ejecutarse sin restricciones importantes, ya que el hardware disponible cumple con los requisitos técnicos y las herramientas seleccionadas son gratuitas y de libre acceso. Esto valida que las decisiones tomadas durante el análisis y diseño fueron realistas, justificadas y alineadas con el contexto académico.

9.2 Recomendaciones

- Es aconsejable profundizar en el uso de patrones de diseño y patrones arquitectónicos para fortalecer aún más la estructura del sistema. Patrones como Repository o Adapter pueden mejorar la separación de responsabilidades y facilitar el mantenimiento en fases posteriores.
- Es importante mantener y reforzar la trazabilidad entre los requisitos, el diseño y los artefactos generados. Herramientas como GitHub Projects o matrices de trazabilidad ayudarán a mantener sincronización entre análisis, diseño y planificación.
- Se recomienda documentar criterios de aceptación para cada requisito, lo cual facilitará establecer parámetros de verificación y asegurará que cada función diseñada cumpla con las expectativas del usuario y del negocio.

Planificación para el Cronograma:

#	TAREA	INICIO	FIN
1	Introducción	02/12/2025	03/12/2025
2	Modificación Base de Datos	03/12/2025	05/12/2025
3	Priorización de Requisitos	05/12/2025	06/12/2025
4	Diseño de Casos de Uso	06/12/2025	08/12/2025
5	Elaboración del Backlog	08/12/2025	09/12/2025
6	Desarrollo de Matrices	09/12/2025	10/12/2025

7	Redacción del Marco Teórico	10/12/2025	11/12/2025
8	Resultados, conclusiones y ajustes	11/12/2025	11/12/2025

Tabla 5. Cronograma del proyecto.

Referencias

- [1] E. Gamma, R. Helm, R. Johnson y J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994. Disponible: <https://www.oreilly.com/library/view/design-patterns-elements/0201633612/>
- [2] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003. Disponible: <https://martinfowler.com/books/eaa.html>
- [3] G. Hohpe y B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003. Disponible: <https://www.enterpriseintegrationpatterns.com/>
- [4] OWASP Foundation, “OWASP Cheat Sheet Series,” 2025. Disponible: <https://cheatsheetseries.owasp.org/>
- [5] A. Metwally, D. Agrawal y A. El Abbadi, “Duplicate detection in data streams using sliding windows,” *SIGMOD '05: Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 85–96, 2005. Disponible: <https://dl.acm.org/doi/10.1145/1066157.1066205>
- [6] J. M. Yu et al., “Receipt Recognition Technology Driven by Multimodal Alignment,” *Electronics*, vol. 14, no. 5, 2025. Disponible: <https://www.mdpi.com/2079-9292/14/5/832>
- [7] Percona LLC, “PostgreSQL Backup Best Practices,” 2024. Disponible: <https://www.percona.com/blog/>
- [8] World Wide Web Consortium (W3C), “Web Content Accessibility Guidelines (WCAG) 2.1,” 2018. Disponible: <https://www.w3.org/TR/WCAG21/>

[9] A. Abdallah et al., "CORU: Comprehensive Post-OCR Parsing and Receipt Understanding Dataset," *arXiv preprint*, 2024.
Disponible: <https://arxiv.org/abs/2405.11010>

[10] K. E. Wiegert and J. Beatty, *Software Requirements*, 3rd ed. Microsoft Press, 2013.

[11] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.