UCUENCA

Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Desarrollo de un Sistema Integral de Gestión de Bodega utilizando un enfoque incremental basado en RUP: Caso Estudio en Empresa Eléctrica Azogues C.A.

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas

Autor:

Beatriz Cecilia Flores Rosales

Director:

Miguel Ángel Zúñiga Prieto

ORCID: 0000-0001-9369-1813

Cuenca, Ecuador

2024-09-04

Resumen

El aumento en la complejidad operativa y la necesidad de optimización en las empresas han incrementado la demanda de sistemas de gestión automatizados. La Empresa Eléctrica Azogues C.A. enfrenta desafíos en la gestión de su bodega debido a un sistema obsoleto que afecta la precisión y eficiencia en la gestión de inventarios.

El proyecto abarca las disciplinas principales de RUP: modelado de negocio, requisitos, análisis y diseño, implementación y evaluación. Se desarrolló un modelo de dominio para comprender los procesos de negocio, y se identificaron y documentaron los requisitos funcionales y no funcionales. Durante el análisis y diseño, se definió una arquitectura modular y escalable. La implementación siguió las directrices del modelo de diseño, alineando el código con la arquitectura definida. Las pruebas se realizaron con usuarios finales en diferentes escenarios, y la evaluación del sistema mediante el cuestionario SUS indicó alta usabilidad y satisfacción de los usuarios.

Los beneficios incluyen mayor precisión en la gestión de inventarios, reducción de errores humanos y mejor visibilidad de la información. Se recomienda ampliar funcionalidades, integrar con otros sistemas corporativos y realizar evaluaciones periódicas para garantizar la satisfacción continua.

Los insumos generados como el análisis del modelo de negocio, flujos de trabajo y prototipos, pueden ser utilizados por otros equipos de desarrollo para desarrollar aplicaciones similares, contribuyendo así al progreso del sector. Este trabajo demuestra que es posible desarrollar una solución personalizada para la gestión de bodega que mejore los niveles de eficiencia operativa y la satisfacción del usuario en un contexto empresarial.

Palabras clave del autor: Desarrollo de software, gestión de bodega, automatización de procesos, Proceso Racional Unificado (RUP)

Abstract

The increase in operational complexity and the need for optimization in companies have increased the demand for automated management systems. Empresa Eléctrica Azogues C.A. faces challenges in the management of its warehouse due to an obsolete system that affects the accuracy and efficiency in inventory management.

The project covers the main disciplines of RUP: business modeling, requirements, analysis and design, implementation and evaluation. A domain model was developed to understand the business processes, and functional and non-functional requirements were identified and documented. During the analysis and design, a modular and scalable architecture was defined. The implementation followed the guidelines of the design model, aligning the code with the defined architecture. Testing was performed with end users in different scenarios, and the evaluation of the system using the SUS questionnaire indicated high usability and user satisfaction.

Benefits include greater accuracy in inventory management, reduction of human errors and better visibility of information. It is recommended to expand functionalities, integrate with other corporate systems and perform periodic evaluations to ensure continued satisfaction.

The inputs generated, such as business model analysis, workflows and prototypes, can be used by other development teams to develop similar applications, thus contributing to the progress of the sector. This work demonstrates that it is possible to develop a customized solution for warehouse management that improves operational efficiency levels and user satisfaction in a business context.

Author Keywords: Software development, warehouse management, process automation, Rational Unified Process (RUP)

Índice de contenido

Índice de figuras

Índice de tablas

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han hecho posible la realización de esta tesis.

En primer lugar, a mi familia por su amor y apoyo incondicional, especialmente a mi mamá Cecilia Rosales, que sin su constante motivación este logro no habría sido posible.

A mi director de tesis, el Ing. Miguel Ángel Zuñiga, por su dedicación, paciencia y orientación durante todo el proceso de desarrollo de este trabajo.

A la Empresa Eléctrica Azogues C.A., por brindarme la oportunidad de realizar este proyecto y por su colaboración durante todo el proceso.

A mis amigos y compañeros de estudio, por su apoyo, motivación y por compartir conmigo esta etapa de mi vida académica.

Finalmente, a todos los profesores de la carrera por impartir sus conocimientos y su dedicación en nuestra formación.

Beatriz Cecilia Flores Rosales

Dedicatoria

Quiero dedicar este trabajo a mi familia, especialmente a mi mamá Cecilia Rosales, quién siempre creyó en mí y me apoyo en todo momento, a mis abuelitos Beatriz Rojas y Macario Rosales (+) que fueron mi pilar y mi fuerza para seguir adelante, a mi hermana Daniela quien siempre encontró las palabras correctas para motivarme a continuar y a mi ñaño Carlos (+) quién siempre me hizo sentir su apoyo y amor incondicional.

Finalmente, quiero dedicar este trabajo a todos los amigos que creyeron en mí y fueron parte de esta etapa, de manera especial a Cristian Marín por ser incondicional conmigo y el amigo que quiero conservar toda la vida.

Beatriz Cecilia Flores Rosales

Capítulo 1: Introducción

1.1. Contexto y Justificación

Los sistemas de información son fundamentales para el soporte y la optimización de los procesos de negocio en cualquier organización. Estos sistemas permiten una gestión eficiente de la información, facilitando la toma de decisiones, mejorando la productividad y asegurando la coherencia y la precisión en la ejecución de las operaciones. En el contexto de la gestión de bodegas, los sistemas de información son esenciales para manejar inventarios, controlar entradas y salidas de materiales, y optimizar el uso del espacio de almacenamiento.

Los sistemas de gestión de bodega presentan una serie de desafíos para algunas empresas, entre estos se puede mencionar la falta de integración y compatibilidad con otros sistemas corporativos, problemas de usabilidad, rendimiento y mantenimiento, y limitaciones para adaptarse a las necesidades específicas de la empresa. Los sistemas ERP, como SAP, Oracle ERP y Microsoft Dynamics, ofrecen soluciones para la gestión de inventarios y bodegas, sin embargo, al ser herramientas estándar adaptarse a los requerimientos específicos de cada empresa puede ser complicado.

La Empresa Eléctrica Azogues C.A. tiene como objetivo principal distribuir y comercializar energía eléctrica en condiciones técnicas y económicas óptimas (“Empresa Eléctrica Azogues C.A.” s.f.). Esta empresa se conforma de diversas áreas críticas que respaldan el cumplimiento de sus objetivos, siendo una de ellas el área de bodega, la cual es fundamental para garantizar un flujo eficiente de materiales y suministros necesarios para las operaciones diarias.

Actualmente, la Empresa Eléctrica Azogues C.A. enfrenta diferentes desafíos en la gestión de su bodega debido a la utilización de su sistema desarrollado en Cobol el cual, si bien ha sido funcional durante años, hoy en día presenta limitaciones que afectan la eficiencia operativa y la precisión en la gestión de inventarios. Entre las limitaciones identificadas se incluyen problemas de usabilidad, rendimiento y mantenimiento, así como una falta de integración y flexibilidad en las operaciones de bodega.

Ante estos desafíos, se hace evidente la necesidad de desarrollar un sistema de gestión de bodega personalizado que se ajuste a las particularidades de la Empresa Eléctrica Azogues C.A. La modernización del sistema de gestión de bodegas es crucial para mejorar la eficiencia operativa y la precisión en la gestión de inventarios. Un sistema moderno permitirá un seguimiento más preciso y eficiente de los materiales, reduciendo errores y optimizando el uso del espacio de almacenamiento. Además, la automatización de procesos disminuirá la dependencia de

entradas manuales, reduciendo los errores humanos y mejorando la eficiencia general. Proveer una plataforma escalable facilitará la implementación de futuras mejoras y la adaptación a nuevas tecnologías o cambios en las operaciones.

Para abordar estos desafíos, este trabajo de titulación tiene como enfoque principal el desarrollo de un Sistema Integral de Gestión de Bodega para la Empresa Eléctrica Azogues C.A. Este sistema pretende contribuir a la mejora de la gestión de inventarios ofreciendo una solución innovadora y sobre todo adaptada a las características y necesidades específicas de la empresa.

La solución propuesta se basa en la implementación de un sistema de gestión de bodega moderno utilizando el Proceso Racional Unificado (RUP). Este enfoque metodológico destaca la importancia en el análisis y diseño antes de la implementación, garantizando que el software cumpla con los requisitos, mitigue riesgos, mejore la calidad y facilite el mantenimiento futuro. Tradicionalmente, muchos proyectos de software tienden a enfocarse principalmente en el producto final, es decir, en el software en funcionamiento. Sin embargo, en este estudio se destacará la importancia de los pasos metodológicos previos que conducen a la programación, enfatizando cómo un enfoque estructurado y disciplinado como RUP puede conducir al éxito del proyecto.

1.2. Objetivos

1.2.1. Objetivo General

Construir un sistema de software para la gestión de bodegas, que automatice los procesos del área de bodega de la empresa Eléctrica Azogues C.A.

1.2.2. Objetivos Específicos

- Analizar los requerimientos funcionales y no funcionales del sistema.
- Diseñar una arquitectura de software que se adapte a las necesidades específicas de la gestión de bodega, definiendo los componentes y sus interacciones, garantizando una estructura modular y escalable del sistema.
- Implementar el Sistema: Construir el sistema de acuerdo con la arquitectura definida.

1.3. Metodología

La metodología empleada para el desarrollo de este trabajo de titulación se basa en el Proceso Racional Unificado (RUP) (Kruchten, 1999), un marco de desarrollo de software iterativo e incremental. RUP proporciona un enfoque estructurado que garantiza la calidad y eficacia en todo el ciclo de vida de un proyecto, asegurando la entrega de software de alta calidad que cumpla con las necesidades de los usuarios finales.

El Proceso Racional Unificado se divide en cuatro fases principales: Inicio, Elaboración, Construcción y Transición. Cada una de estas fases incluye una serie de iteraciones que permiten refinar y mejorar el sistema progresivamente. A continuación, se describen las fases y las actividades involucradas en esta metodología, y en la Figura 1.1 se puede apreciar de forma gráfica:

- Inicio: En esta fase inicial se define el alcance del proyecto y se desarrolla una visión general del sistema. Las principales disciplinas son:
  - Modelado de Negocio: Comprender y modelar los procesos de negocio y definir los objetivos del sistema.
  - Requisitos: Identificar y documentar los requisitos funcionales y no funcionales del sistema.
- Elaboración: Esta fase se centra en analizar los requerimientos, definir la arquitectura del sistema y mitigar los riesgos identificados. Las disciplinas clave en esta fase incluyen:
  - Análisis y Diseño: Transformar los requisitos en una arquitectura de sistema que pueda ser implementada. Además, se realiza la especificación detallada del diseño.
  - Implementación: Planificar y comenzar con la implementación del sistema basándose en la arquitectura definida.
  - Pruebas: Definir las estrategias y casos de prueba iniciales para asegurar la calidad del sistema.
  - Gestión de Configuración y Cambio: Establecer el control de versiones y gestionar los cambios en los requisitos y diseño.
- Construcción: Tiene como principal objetivo implementar y probar el sistema de acuerdo con la arquitectura definida. Las disciplinas que predominan en esta fase son:
  - Implementación: Desarrollar el código del sistema según los diseños establecidos.

- Pruebas: Ejecutar pruebas detalladas para verificar que el sistema funciona conforme a los requisitos y solucionar los defectos encontrados.
- Gestión de Configuración y Cambio: Continuar gestionando los cambios y mantener la integridad del sistema.

- Transición: En esta última fase se lleva a cabo la entrega final del producto y una verificación del software desarrollado para asegurar que cumpla con todos los requerimientos establecidos. Las disciplinas involucradas son:
  - Pruebas: Realizar pruebas finales para asegurar que el sistema esté listo para su despliegue.
  - Implementación: Realizar ajustes finales y resolver cualquier defecto pendiente.
  - Despliegue: Planificar y ejecutar la transición del sistema al entorno de producción.
  - Gestión de Proyecto: Evaluar el rendimiento del proyecto y documentar las lecciones aprendidas.

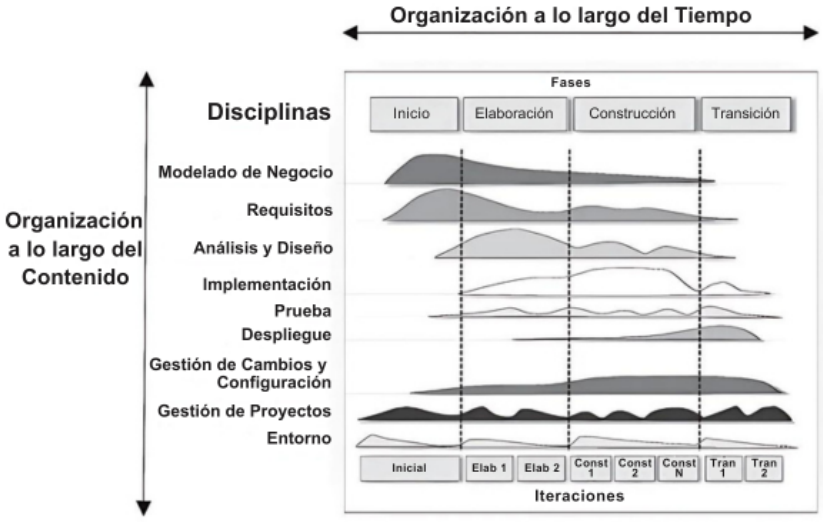


Figura 1.1: Fases del Proceso Racional Unificado. Fuente: Kruchten, 1999

Esta estructura metodológica asegura que cada fase del desarrollo esté respaldada por actividades y disciplinas específicas que contribuyen a la construcción de un sistema robusto y alineado con las necesidades del negocio.

1.4. Estructura del Documento

Este documento se estructura en seis capítulos, diseñados para ofrecer una visión detallada del desarrollo del Sistema Integral de Gestión de Bodega para la Empresa Eléctrica Azogues

C.A, utilizando un enfoque incremental basado en el Proceso Unificado de Desarrollo (RUP). A continuación, se detalla la organización del documento:

Capítulo 1. Introducción. Este capítulo establece el marco inicial del proyecto, presentando el contexto, la justificación del desarrollo del sistema de gestión de bodega y la metodología usada. Se detallan los objetivos del proyecto y se proporciona una visión general de la estructura del documento, orientando al lector sobre el contenido y la organización del trabajo.

Capítulo 2. Marco Teórico: Se presentan los conceptos necesarios para un buen entendimiento de los espacios tecnológicos y metodologías utilizadas durante el desarrollo de este trabajo; con un enfoque especial en el Proceso Racional Unificado (RUP).

Capítulo 3. Desarrollo del Sistema de Gestión de Bodega utilizando un enfoque incremental basado en RUP: Este capítulo sigue el marco de desarrollo RUP, se incluyen las disciplinas del modelado de negocio, requisitos, análisis y diseño e implementación.

Capítulo 4. Pruebas y Evaluación del Sistema: Este capítulo presenta una descripción del proceso de pruebas y evaluación del sistema para garantizar que cumpla con los requisitos especificados y proporcione una experiencia de usuario satisfactoria.

Capítulo 5. Conclusiones: El capítulo final sintetiza los resultados obtenidos, presenta las conclusiones del proyecto y ofrece recomendaciones para investigaciones o desarrollos posteriores.

Capítulo 2: Marco Teórico

En este capítulo, se procederá a establecer las explicaciones de los conceptos fundamentales dentro del campo relacionado con el desarrollo del sistema de gestión de bodega. Se analizarán en detalle tanto los principios elementales como aquellos más específicos del entorno tecnológico en el cual se desenvuelve el proyecto. Esto incluye una revisión del Proceso Racional Unificado de Desarrollo (RUP), que es la metodología adoptada para este proyecto, así como una descripción de las herramientas y tecnologías empleadas. Además, se hará mención de los detalles de la arquitectura de software seleccionada.

2.1. Proceso Racional Unificado (RUP)

El Proceso Racional Unificado (RUP) es un enfoque estructurado para el desarrollo de software que asegura calidad y eficacia a lo largo del ciclo de vida de un proyecto. RUP ofrece un marco de trabajo flexible y adaptable a las necesidades específicas de cada proyecto. Este proceso esta organizado en varias fases, cada una con sus actividades y roles específicos (Anwar, 2014).

Dentro del contexto de RUP es importante definir algunos términos relacionados. En primer lugar, se encuentra el concepto de fase, que hace referencia a las etapas secuenciales por las que atraviesa el desarrollo de software. Cada fase tiene objetivos específicos y se compone de diferentes actividades. Por otro lado, se encuentra el término iteración, que indica la repetición de un conjunto de actividades dentro de una fase, las iteraciones permiten una entrega incremental y una retroalimentación temprana (Abuchar Porras, 2023).

2.1.1. Estructura de RUP

La Figura 2.1 muestra la arquitectura general del Proceso Racional Unificado, se tiene dos estructuras o dos dimensiones:

1. El eje horizontal representa el tiempo y muestra los aspectos dinámicos del proceso en términos de fases e iteraciones.
2. El eje vertical representa las disciplinas de procesos estáticos del proceso, este se presenta en términos de actividades, artefactos, roles, flujos de trabajo y disciplinas.

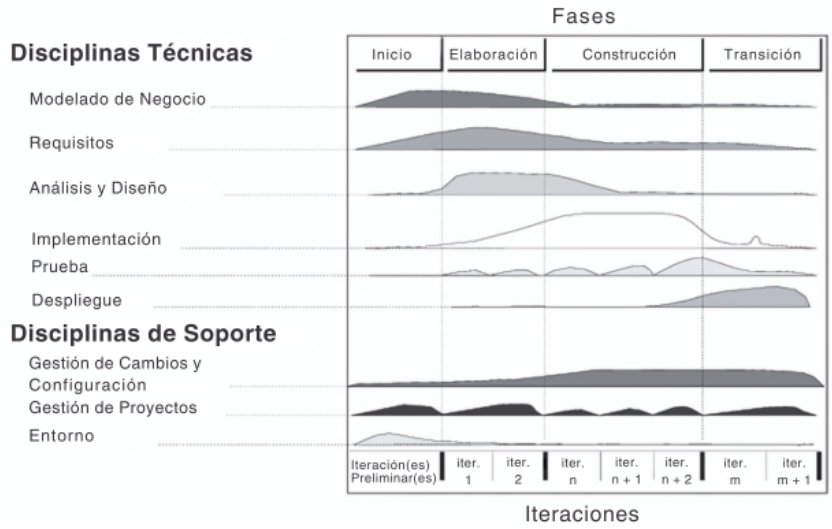


Figura 2.1: Estructura del Proceso Racional Unificado. Fuente: Kruchten, 1999

2.1.2. Estructura Estática

Hace referencia a la descripción del proceso en términos de sus componentes estáticos, como roles, actividades, artefactos, flujos de trabajo y disciplinas. Estos elementos proporcionan una base sólida para comprender cómo se organiza y se ejecuta el proceso de desarrollo de software en un entorno controlado y disciplinado.

Roles

Representan las responsabilidades y funciones asignadas a los miembros del equipo de desarrollo de software, cada uno tiene tareas específicas que contribuyen al éxito del proyecto. A continuación, se presentan algunos de los principales roles de RUP y en qué disciplina actúan:

- Analista de Negocios (Business Analyst): Se enfoca en comprender los requisitos del negocio, identificar oportunidades de mejora y traducirlos en requisitos de software. Este rol actúa tanto en la disciplina de modelado de negocios como en la de requisitos.
- Arquitecto de Software (Software Architect): Es el encargado de definir la arquitectura del sistema. Está involucrado en la disciplina de análisis y diseño y la disciplina de implementación.
- Diseñador de Software (Software Designer): Está involucrado en la disciplina de análisis y diseño, y es responsable de convertir los requisitos del sistema en un diseño técnico detallado.

- Desarrollador de software (Software Developer): Está involucrado en la disciplina de implementación, y está enfocado en la implementación de los componentes del sistema, escribiendo código, compilando y probando los elementos desarrollados.
- Evaluador (Tester): En la disciplina de pruebas, es el encargado de planificar, diseñar, ejecutar y evaluar las pruebas de software para garantizar la calidad del producto final.

Actividades

Son las acciones concretas que se realizan dentro del proceso para lograr objetivos específicos. Las actividades están organizadas en secuencias lógicas para guiar la ejecución del trabajo, cada actividad tiene un propósito que está generalmente expresado en términos de crear o actualizar artefactos y cada una está asignada a un rol específico (Kruchten, 1999).

Artefactos

Son los productos de trabajo generados durante el proceso, como documentos, modelos, diagramas, código fuente, planes, entre otros. Los artefactos capturan información importante y sirven como evidencia del progreso y la calidad del trabajo realizado. A continuación, se describen los principales artefactos de RUP y en la Figura 2.2 se puede apreciar gráficamente.

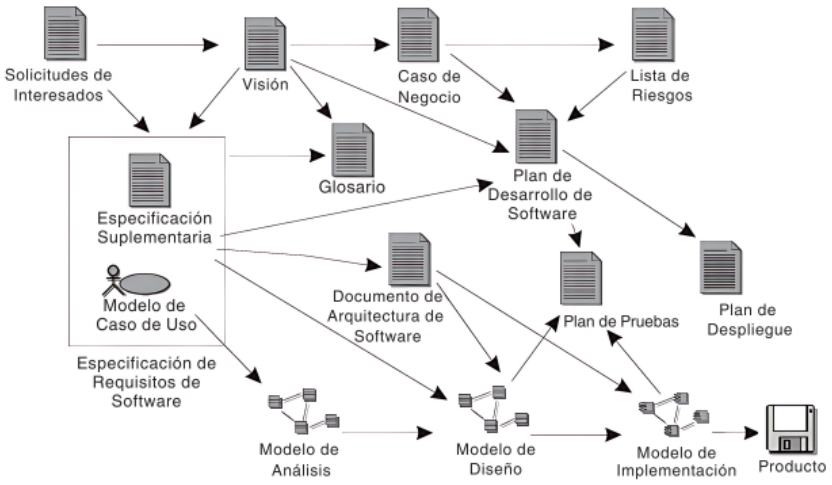


Figura 2.2: Principales artefactos del Proceso Racional Unificado. Fuente: Kruchten, 1999

- Solicitudes de las partes interesadas (Stakeholder Requests): Recopilación de las opiniones y expectativas de las partes interesadas para proporcionar información sobre los requisitos y comprender las necesidades reales. La información obtenida permite determinar el comportamiento del sistema.

- Visión (Vision): Define la misión y alcance del negocio.
- Caso de uso de negocio (Business Case): Describe la estructura y procesos del negocio.
- Lista de Riesgos (Risk List): Riesgos que podrían aparecer durante el desarrollo del proyecto.
- Modelo de Casos de Uso (Use-Case Model): Modelo que describe un escenario de interacción entre un actor y el sistema para lograr un objetivo de negocio específico.
- Especificaciones suplementarias (Supplementary Specification): Sirven como complemento al modelo de casos de uso, permite capturar la interacción precisa entre los actores y el sistema, ofreciendo una representación detallada del flujo de eventos y las condiciones que rigen la funcionalidad del sistema (Jacobson et al., 1999).
- Glosario (Glossary): Define una terminología común a ser utilizada de forma consistente durante todo el proyecto, partiendo del modelado de negocios.
- Plan de Desarrollo de Software (Software Development Plan): Establece la estrategia y el enfoque para el desarrollo del sistema.
- Documento de Arquitectura de Software (Software Architecture Document): Describe la estructura y organización del sistema incluyendo sus componentes y subsistemas.
- Plan de Pruebas (Test Plan): Define la estrategia de pruebas, casos de prueba y criterios de aceptación.
- Plan de Despliegue (Deployment Plan): Especifica los pasos y tácticas para implementar el sistema en el entorno de producción.
- Documento de Requisitos de Software (Software Requirements Specification): Especifica los requisitos funcionales y no funcionales del sistema.
- Modelo de Análisis (Analysis Model): Modelo que consiste en un modelo de objetos que describe cómo se realizan los casos de uso en términos de roles y entidades de negocios que interactúan entre sí.
- Modelo de Diseño (Design Model): Muestra la estructura detallada del sistema incluyendo clases, relaciones y componentes.
- Modelo de Implementación (Implementation Model): Describe cómo se implementarán los componentes y subsistemas del sistema.

Flujos de Trabajo

Representan las secuencias de actividades relacionadas que guían la ejecución del proceso, describen cómo se llevan a cabo las actividades en el contexto de un proyecto de software (Kruchten, 1999).

Disciplinas

Son áreas de enfoque dentro del proceso que agrupan actividades relacionadas, cada disciplina aborda aspectos específicos del desarrollo de software. RUP tiene nueve disciplinas básicas, de las cuales seis son técnicas y tres de soporte (Kruchten, 1999).

Las disciplinas técnicas son las siguientes: modelado de negocios, requisitos, análisis y diseño, implementación, prueba y despliegue. Por otra parte, las disciplinas de soporte son las siguientes: gestión de proyectos, gestión de cambios y configuración, y entorno.

La disciplina de modelado de negocio tiene como propósito principal, comprender la estructura de la empresa u organización objetivo y de esta forma garantizar que todas las partes interesadas tengan una comprensión común y a partir de este definir de manera efectiva los requisitos del sistema y satisfacer las necesidades del negocio.

La disciplina de requisitos, toma como punto de partida el modelado de negocio para la comprensión del problema a resolver y el alcance de dicho problema. El objetivo de esta disciplina es establecer un acuerdo entre las partes interesadas y el equipo de desarrollo sobre lo que debe hacer el sistema. Los requisitos pueden ser funcionales o no funcionales, los primeros describen las funcionalidades que el sistema debe ser capaz de realizar y por otra parte los requisitos no funcionales hacen referencia a las características del sistema relacionadas con atributos de calidad como el de rendimiento, seguridad, usabilidad, escalabilidad, entre otros.

La disciplina de análisis y diseño, por su parte tiene como propósito principal convertir los requisitos del sistema en una especificación de cómo implementar el sistema. Se recalca la importancia de comprender los requisitos y transformarlos en un diseño de sistema seleccionando la mejor estrategia de implementación. El análisis tiene como propósito analizar los requisitos de la captura de requisitos, refinarlos y solucionar los aspectos que se quedaron sin resolver, para conseguir una compresión más precisa de los requisitos y una descripción mantenible de los mismos y que permitan estructurar el sistema en su totalidad, se puede decir que el análisis sirve como una primera aproximación al diseño (Jacobson et al., 1999).

Durante el análisis, se comienza a materializar los requisitos en elementos constructivos del sistema, tales como clases y componentes, que serán la base para el diseño. Esta disciplina es esencial para asegurar que las funciones del sistema se alineen con las necesidades de

negocio y proporcionen una solución coherente y sostenible (Jacobson et al., 1999).
Por otro lado, el diseño le da la forma al sistema para que soporte todos los requisitos identificados previamente. En el diseño se modela el sistema y se encuentra la estructura ideal, para que soporte todos los requisitos identificados. Este proceso se basa en una comprensión detallada de los requisitos y la estructura del sistema definida durante las actividades de análisis (Jacobson et al., 1999).
La disciplina de implementación tiene como propósitos; definir la organización del código usando subsistemas organizados por capas, implementar clases y objetos en términos de componentes, realizar pruebas de los componentes desarrollados e integrar los resultados en un sistema ejecutable. La disciplina de pruebas se centra en valorar la calidad del producto y por último la disciplina de despliegue tiene como propósito principal planificar y ejecutar la implementación exitosa del sistema de software en el entorno de producción.

2.1.3. Estructura Dinámica

La estructura dinámica describe el ciclo de vida de RUP, es decir, detalla cómo se desarrolla el proceso a lo largo del tiempo. RUP se compone de las siguientes fases:

Inicio

La fase de inicio es la primera etapa del proceso RUP, cuyo objetivo principal es establecer la visión y el alcance del proyecto. En esta fase se definen los objetivos, alcance y restricciones del proyecto. Entre las actividades que se realizan en esta fase está la identificación de los stakeholders, la delimitación del sistema, la definición de los casos de uso y la elaboración de un plan preliminar para el desarrollo. También, se realiza una evaluación de riesgos para identificar posibles obstáculos y se establece un marco de trabajo inicial para guiar el proceso de desarrollo de software. Al finalizar esta fase, se espera contar con una visión clara del proyecto y un plan detallado para la siguiente fase (Dragos, 2021).

Elaboración

La fase de Elaboración en RUP se centra en realizar un análisis más detallado y completo de los requisitos del sistema. Durante esta fase, se realiza una planificación detallada, se desarrollan casos de uso ampliados y se crea un modelo de análisis para comprender mejor las necesidades del proyecto. También se prepara un prototipo del sistema y se evalúa su viabilidad técnica y funcional. Además, se llevan a cabo actividades de gestión de riesgos para identificar y abordar cualquier problema potencial (Dragos, 2021).

Construcción

Durante la fase de Construcción en RUP, se desarrolla el software utilizando iteraciones y entregas incrementales. Los requisitos ya están definidos y se trabaja en la implementación de las funcionalidades. Se lleva a cabo la codificación, pruebas unitarias y de integración, así como la documentación necesaria. Además, se refina la arquitectura y se realizan pruebas de rendimiento. Es fundamental la colaboración constante entre los desarrolladores y los diferentes roles involucrados en el proceso. Al final de esta fase, se debe tener una versión estable del software listo para ser probado en la fase de Transición (Dragos, 2021).

Transición

En la fase de Transición se realiza la entrega final del producto al cliente. Durante esta fase se da lugar la verificación del software desarrollado para asegurar que cumpla con todos los requerimientos establecidos y no presente errores. Además, se realizan pruebas de integración y de aceptación con el propósito de validar su correcto funcionamiento, así como también, se realiza la capacitación y documentación necesaria para que el cliente pueda utilizar el sistema adecuadamente. Por último, se establece el soporte necesario para resolver cualquier problema o defecto que pueda surgir después de la entrega. En pocas palabras, la fase de Transición se enfoca en finalizar el proyecto de manera exitosa, garantizando que el producto final cumpla con los estándares de calidad y satisfaga las necesidades del cliente (Dragos, 2021).

2.2. Herramientas y Tecnologías

En esta subsección, se describen las principales herramientas y tecnologías empleadas, incluyendo Angular para el desarrollo frontend, Spring Boot para el desarrollo backend y PostgreSQL como sistema de gestión de bases de datos.

2.2.1. Angular

Angular es un framework de desarrollo creado por Google que permite crear aplicaciones eficientes y sofisticadas (“Angular”, s.f.). Esta plataforma incluye: un marco basado en componentes, una colección de librerías integradas que cubren una variedad de funcionalidades y un conjunto de herramientas de desarrollo para ayudar a desarrollar, crear, evaluar y actualizar código (“Getting started with Angular | MDN”, 2024).

2.2.2. Spring Boot

Spring Framework es un framework de Java que permite un modelo integral de programación y configuración para aplicaciones empresariales basadas en Java (“Spring Framework”, s.f.). Uno de sus módulos más importantes es Spring Boot (“Spring Boot”, s.f.), el cual es considerado como una forma de simplificar el desarrollo de aplicaciones Spring. Este proporciona preconfiguraciones para el desarrollo rápido de aplicaciones, lo que permite desvincularse de las complejidades en las configuraciones. Las aplicaciones que usan Spring Boot pueden ejecutarse y funcionar de manera independiente sin necesidad de un servidor web externo (Enn, 2023).

2.2.3. PostgreSQL

PostgreSQL (Group, 2024) es un sistema de gestión de bases de datos de código abierto conocido por su robustez y conformidad con el estándar SQL. Es reconocido por su fiabilidad, flexibilidad y soporte de estándares técnicos abiertos. PostgreSQL soporta tipos de datos relacionales y no relacionales, lo que lo convierte en una de las bases de datos disponibles más compatibles, estables y maduras (“¿Qué es PostgreSQL?”, 2024).

2.3. Arquitectura de Software

Cuando se habla de arquitectura se hace referencia al conjunto de componentes interrelacionados que le dan forma a un sistema. La arquitectura de software define los principios y directrices que guían el diseño y la evolución de un sistema. Existen arquitecturas de sistemas informáticos de propósito general y otras específicas para área en especial. Se pueden mencionar algunas de las arquitecturas existentes: arquitecturas genéricas de sistemas multimedia; genéricas de escritorio; cliente-servidor; business-to-business, Internet, Intranet y Extranet, etc (Englander, 2021).

2.3.1. Arquitectura por Capas

La arquitectura por capas organiza el software en capas distintas, cada una de las cuales tiene responsabilidades específicas. Cada capa proporciona servicios a la capa superior y utiliza los servicios de la capa inferior. Este modelo promueve la separación de responsabilidades, la modularidad y la mantenibilidad del sistema. Las capas comunes en una arquitectura por capas incluyen:

- Capa de presentación: Es responsable de interactuar con el usuario.
- Capa de negocio: Esta capa contiene la lógica de negocio.
- Capa de acceso a datos: Es encargada de acceder a los datos alojados en la base de datos.

2.3.2. Modelo Cliente-Servidor

La arquitectura de software cliente-servidor describe una relación de colaboración entre dos tipos de componentes: clientes y servidores. En este modelo los clientes solicitan servicios o recursos, mientras que los servidores los proveen. Por ejemplo, un cliente puede ser un navegador web, mientras que un servidor puede ser un servidor ubicado en cualquier parte del mundo esperando que un navegador le solicite una página web. Actualmente, este modelo es esencial para muchas aplicaciones y sistemas distribuidos, incluyendo aplicaciones web, sistemas de bases de datos, entre otros (Garibay, 2023). A continuación, se describen los roles involucrados en este modelo:

- Cliente: Es una aplicación o dispositivo que realiza solicitudes de servicios, recursos o datos al servidor. Los clientes suelen ser las interfaces de usuario que los usuarios finales utilizan para interactuar con el sistema (Garibay, 2023).
- Servidor: Es una aplicación o dispositivo que responde a las solicitudes de los clientes proporcionando servicios, recursos o datos. Estos recursos y servicios pueden incluir información, imágenes, videos, archivos, bases de datos, entre otros. Los servidores gestionan, procesan y almacenan los datos y pueden responder múltiples solicitudes de clientes al mismo tiempo (Garibay, 2023).

Entre las principales características de este modelo está: la separación de responsabilidades al definir las funciones del cliente y del servidor; la interacción a través de una red utilizando protocolos de comunicación como HTTP, TCP/IP, etc. Por último, la independencia de la plataforma, pues los clientes y servidores pueden ejecutarse en diferentes plataformas y sistemas operativos.

2.3.3. API REST

Una API (Application Programming Interface) es un conjunto de reglas y convenciones que permiten la comunicación e intercambio de datos y servicios entre dos o más sistemas mediante

mensajes y llamadas a funciones. La API REST es un tipo de API que permite que diferentes aplicaciones se comuniquen entre sí a través de internet utilizando el protocolo HTTP para realizar operaciones CRUD (Create, Read, Update, Delete) (Márquez Reyes, 2023).

2.4. Seguridad y Autenticación

En el desarrollo de aplicaciones empresariales la seguridad es un aspecto importante que debe ser abordado, pues proteger los datos sensibles y asegurar que solo los usuarios autorizados accedan a determinadas funcionalidades es fundamental para mantener la integridad y la confiabilidad del sistema.
Entre las tecnologías y mecanismos más comunes para garantizar la seguridad y autenticación se encuentran:

- Spring Security: Es un marco de control de acceso y autenticación que provee mecanismos para la autenticación y autorización en aplicaciones basadas en Spring (“Spring Security”, s.f.).
- JSON Web Token (JWT): Es un estándar abierto para la creación de tokens de acceso para una autenticación segura entre cliente y servidor. JWT permite verificar la identidad del usuario y autorizar el acceso a recursos protegidos de manera eficiente (“Token web JSON (JWT) | IBM”, 2024).
- OAuth2: Es un protocolo de autorización que permite a las aplicaciones que lo usan obtener acceso limitado a los datos de un usuario sin exponer sus credenciales (“¿Qué es OAuth 2.0 y para qué sirve?”, s.f.).
- LDAP (Lightweight Directory Access Protocol): Un protocolo abierto utilizado para acceder y mantener servicios de directorio distribuidos a través de una red (“LDAP.com”, s.f.).
- SSL/TLS (Secure Sockets Layer/Transport Layer Security): Son certificados que protegen las conexiones a internet mediante el cifrado de los datos (“¿Qué son SSL, TLS y HTTPS? | DigiCert”, s.f.).

2.5. Patrones de Diseño

Los patrones de diseño se pueden definir como soluciones reutilizables para problemas comunes en el diseño de software (“Patrones de diseño / Design patterns”, s.f.). Estos patrones

simplifican el desarrollo de código al hacerlo más fácil de entender y administrar. Es importante tener en cuenta el uso de patrones de diseño durante el desarrollo de aplicaciones, ya que estos contribuyen a la creación de sistemas flexibles y mantenibles.

2.5.1. Patrón Repository

El patrón Repository permite la separación de la lógica de acceso a datos y la lógica de negocio. Este patrón provee una interfaz para la interacción con la base de datos, lo cual permite un código limpio y mantenible (Garrido Tejero, 2021).

2.5.2. Patrón Data Transfer Object (DTO)

El Patrón Data Transfer Object (DTO) es utilizado para agrupar y transferir datos entre diferentes capas de una aplicación. Este patrón mejora el rendimiento, ya que reduce la cantidad de llamadas necesarias entre la capa de presentación y la capa de negocio, minimiza la exposición de los datos transferidos y facilita la validación y transformación de los datos (Daniele y Romero, 2006).

2.6. Sistemas de Gestión de Bodega

Los sistemas de gestión de bodega (WMS, por sus siglas en inglés) son sistemas de información que facilitan la gestión de las operaciones diarias en una bodega o almacén (“¿Qué es un WMS (sistema de gestión de almacenes)?”, s.f.). Estos sistemas tienen varias responsabilidades clave, que incluyen:

- Control de Inventarios: Seguimiento preciso de la cantidad y ubicación de los productos dentro del almacén.
- Recepción de mercancías: Incluye la verificación y el almacenamiento de los productos receptados.
- Automatización de procesos: Reducción de tareas manuales.
- Integración con otros sistemas: Conexión con sistemas ERP y otras aplicaciones corporativas para asegurar la coherencia de la información en toda la empresa.

En la actualidad, la gestión eficiente de inventarios y activos se ha convertido en un desafío crítico para empresas e instituciones, tanto privadas como públicas. La complejidad inherente

a estos procesos ha llevado a reconocer la necesidad de contar con sistemas informáticos especializados que simplifiquen y automaticen estas operaciones. La tecnología se ha convertido en un elemento clave para la competitividad y el éxito empresarial en un entorno cada vez más digitalizado y globalizado (Fernández Samprieto, 2017).

Trabajos como los de (Cabrera Pérez, 2020), (Carrera Proaño y Quito Chuquin, 2020) y (Vivar Maldonado, 2015) sobre sistemas de gestión de inventarios/bodegas, subrayan la importancia de implementar tecnologías avanzadas para superar las limitaciones de los métodos tradicionales. Entre las soluciones de software comerciales para la gestión de inventario/bodega se incluyen sistemas como SAP ERP ("SAP ERP", s.f.), Oracle NetSuite ("Business Software, Business Management Software | NetSuite", s.f.), Odoo ("ERP y CRM de fuente abierta | Odoo", s.f.) y Microsoft Dynamics 365 ("Dynamics 365 Business Central", s.f.). Si bien SAP ERP y Oracle NetSuite son robustos y cuentan con amplias funcionalidades, a menudo presentan desafíos en términos de costos y adaptabilidad. Odoo, como solución de código abierto, ofrece flexibilidad, pero requiere esfuerzos significativos en personalización y mantenimiento. Microsoft Dynamics 365, a pesar de su integración con productos Microsoft, puede ser costoso y complejo de implementar y operar.

A pesar de las ventajas de las diferentes plataformas comerciales mencionadas, estos sistemas presentan desafíos significativos en términos de personalización y costo. Aunque son robustos, pueden no ser la mejor opción para todas las empresas. Investigaciones como (Tulaskar et al., 2022) destacan la importancia de diseñar soluciones personalizadas que optimicen los flujos de trabajo y mejoren la eficiencia operativa. Asimismo, estudios comparativos como (Mishra y Alzoubi, 2023) enfatizan la necesidad de utilizar metodologías de desarrollo de software que proporcionen un marco estructurado y organizado para guiar el proceso de desarrollo, asegurando calidad, eficiencia y éxito del proyecto.

Los trabajos de (Ruiz Salvatierra, 2019) y (Cárdenas Lara et al., 2019) también resaltan la importancia de diseñar un sistema que se adapte a las necesidades y características únicas de una organización para mejorar la eficiencia y efectividad en la gestión de inventarios. En este contexto, el desarrollo de un sistema a medida se presenta como una opción viable y ventajosa.

De manera similar, el trabajo (Castillo De La Torre y Guinet Huerta, 2023) muestra cómo la integración de soluciones móviles con un ERP existente puede reducir el tiempo de ejecución de tareas de control de existencias y preparación de pedidos, utilizando metodologías como PMBOK y RUP para estructurar el desarrollo del software.

Además, otros estudios refuerzan la importancia de aplicar una metodología de desarrollo de

software estructurada. Por ejemplo, en el estudio (Jaramillo, 2016), se aplicó la metodología RUP junto con el patrón de diseño MVC, destacando la organización, la gestión eficiente del tiempo y recursos, la calidad del producto final, y la comunicación efectiva entre los miembros del equipo. Igualmente, en el desarrollo de un sistema de inventarios, la metodología RUP fue crucial para asegurar un marco estructurado desde la concepción hasta la implementación, facilitando la gestión eficiente del proyecto (Castillo Sarzosa, 2017).

Finalmente, el estudio (Contreras Paredes, 2017) también subraya la importancia de usar una metodología de desarrollo como RUP. Este enfoque estructurado permitió una gestión disciplinada del proyecto, la identificación y mitigación temprana de riesgos, y la documentación adecuada de cada fase del proyecto.

En conclusión, si bien las herramientas comerciales pueden proporcionar soluciones completas a menudo pueden presentar limitaciones en cuanto a la personalización y a los costos. Por esta razón, realizar un análisis para determinar la mejor opción es esencial, ya que en muchos casos desarrollar una herramienta desde cero que se ajuste a las necesidades de la empresa puede ser la mejor alternativa. Además, este enfoque permite utilizar metodologías de desarrollo de software estructuradas que garantizan la calidad, eficiencia y éxito del proyecto.

Capítulo 3: Desarrollo del Sistema Integral de Gestión de Bodega utilizando un enfoque incremental basado en RUP

Este capítulo describe detalladamente el proceso de desarrollo del Sistema Integral de Gestión de Bodega, aplicando el Proceso Racional Unificado (RUP). El objetivo principal es desarrollar un sistema personalizado para la Empresa Eléctrica Azogues C.A., que en adelante será referenciada como empresa. RUP se caracteriza por ser iterativo e incremental, permitiendo que el progreso del proyecto se visualice a través de distintos artefactos generados en cada una de las disciplinas de RUP. Estos artefactos reflejan las iteraciones y refinamientos sucesivos que culminan en la creación de productos de software de alta calidad.

En este capítulo se documentan los artefactos resultantes de los flujos de trabajo realizados para este proyecto. Cada sección se enfoca en una disciplina específica de RUP y se presentan los artefactos generados a partir de las actividades realizadas.

El desarrollo del sistema comienza con la disciplina de Modelado de negocio en la que se realiza el modelo de dominio para comprender los procesos de negocio y las necesidades de los stakeholders. En la disciplina de Requisitos, se capturan y documentan los requisitos mediante el modelo de casos de uso, dicho modelo documenta las interacciones con los stakeholders y define las funcionalidades del sistema.

Posteriormente, en la disciplina de Análisis y Diseño, se abstraen las especificaciones para definir la estructura interna del sistema. Este modelo de análisis se refina en el modelo de diseño, que prepara el camino para una implementación eficaz.

La disciplina de Implementación se rige por las directrices establecidas en el modelo de diseño, esto asegura que el código desarrollado esté alineado con la arquitectura definida en la disciplina de análisis y diseño.

Finalmente, en la disciplina de Pruebas se verifica y valida la funcionalidad del sistema para garantizar que cumpla con los requisitos establecidos.

Cada uno de estos modelos mencionados son artefactos de las disciplinas principales de RUP: Modelado de negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas y Despliegue. Este enfoque metodológico asegura que cada componente del sistema es examinado y desarrollado de manera integral, reflejando el dinamismo y la retroalimentación continua que son fundamentales para la metodología RUP.

3.1. Modelado de Negocio

El desarrollo del sistema comienza con el modelado del negocio para la comprensión de los procesos de negocio y las necesidades de los stakeholders, durante esta disciplina se toma el rol de analista de negocios y como artefacto se establece la creación de un modelo de dominio, para representar la estructura y las entidades clave del dominio desde una perspectiva de alto nivel.

Para desarrollar el modelo de dominio se realizó un análisis de los diagramas de procesos proporcionados por la empresa. Estos diagramas detallan el flujo operacional y las interacciones dentro del proceso de gestión de bodega lo que permitió identificar los objetos de negocio relevantes y sus relaciones.

Además, se llevaron a cabo entrevistas con el encargado de la bodega y otros empleados involucrados en los procesos de gestión de inventario y solicitudes de bodega. Estas entrevistas ayudaron a esclarecer cómo se llevan a cabo las operaciones diarias, los términos y el lenguaje específico utilizado, asegurando que el modelo de dominio refleje fielmente la realidad operativa y las necesidades del usuario.

En el contexto de la Empresa Eléctrica Azogues C.A., la gestión de bodega es un proceso esencial, ya que, asegura la disponibilidad y el adecuado manejo de los recursos para el funcionamiento eficiente de la empresa. El modelo de dominio completo está incluido en la sección de Anexos (ver extracto del modelo en la Figura 3.1). Este modelo captura y organiza los elementos clave de este proceso, reflejando la realidad operativa y facilitando la comprensión de cómo se manejan las operaciones de bodega en un entorno real, la empresa.

El modelo de dominio propuesto se ha representado mediante un diagrama de clases, dicho diagrama ilustra cómo se relacionan las distintas entidades (información) del dominio entre sí. Los procesos de ingresos y egresos realizados son denominados operaciones. Dentro de bodega se diferencian tres tipos de artículos: bienes, materiales y suministros de oficina.

El proceso de ingreso se inicia cuando los artículos son suministrados por diferentes proveedores, que están registrados en el sistema. Cada artículo ingresado es registrado en el sistema. Este registro facilita la gestión y localización eficiente de los artículos dentro de la bodega, donde cada pieza es almacenada de acuerdo con su tipo y necesidades específicas.

Las solicitudes de egreso y solicitudes de compra son documentos impresos en formularios predefinidos. Las solicitudes de egreso, enviadas desde los departamentos requirentes de la empresa, reflejan la necesidad para diversas operaciones, proyectos o actividades diarias. Estas solicitudes son aprobadas antes de proceder con el egreso. Por otra parte, las solicitudes

de compra son exclusivamente realizadas por el departamento de bodega, basándose en el stock y los requerimientos de los distintos departamentos. Tanto las notas de entrega y notas de devolución hacen referencia a documentos físicos que son usados para el ingreso o egreso de bienes respectivamente.

Para mantener el control del inventario se realizan tomas físicas periódicamente. Estas actividades son realizadas para asegurar que el inventario físico coincide con los registros del sistema y para ajustar cualquier discrepancia que pueda surgir debido a errores de registro o problemas de entrega.

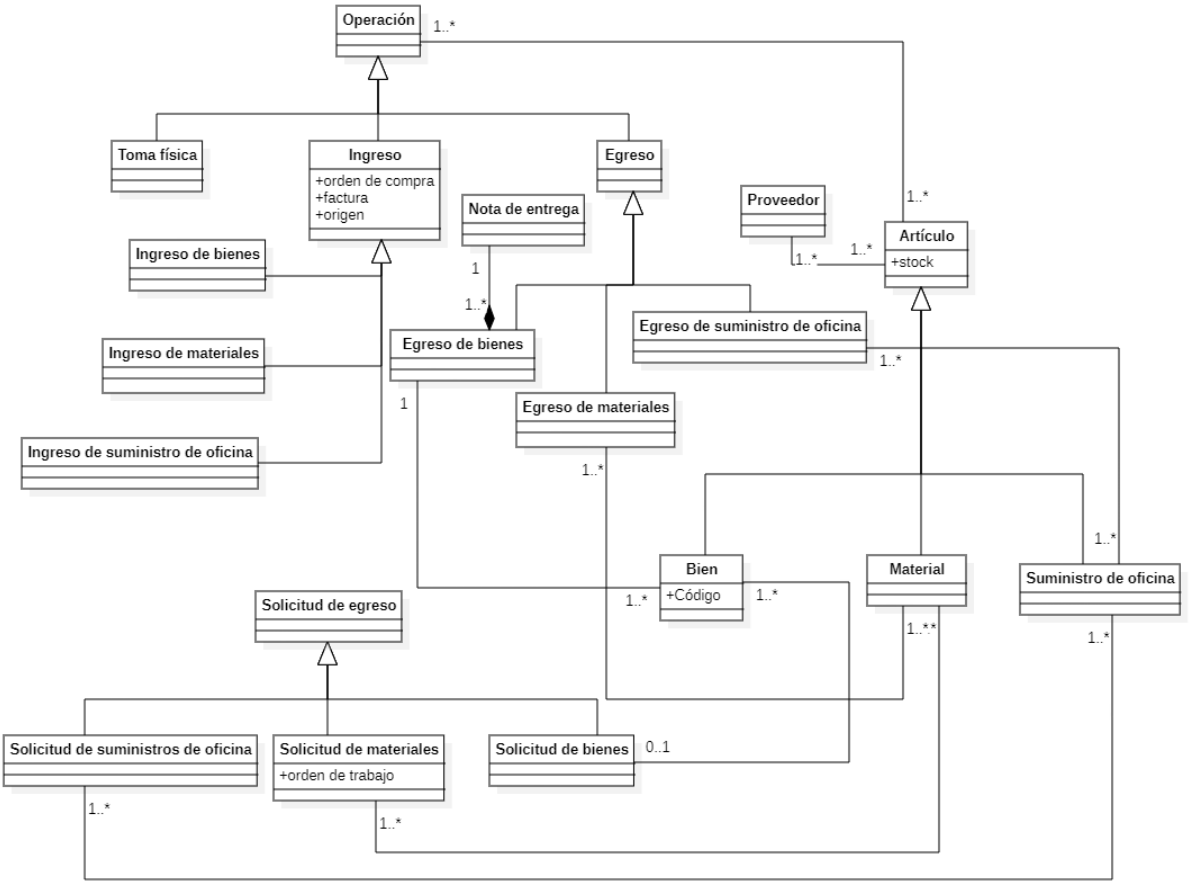


Figura 3.1: Extracto del Modelo de Dominio

A continuación, se explican en qué consiste las clases principales que conforman el modelo de dominio:

- Operación: Esta clase actúa como una categoría general para cualquier tipo de transacción dentro de la bodega. Es fundamental para el seguimiento de todas las actividades que afectan el inventario. La clase ingreso detalla los bienes que entran a la bodega mientras que la clase egreso registra los bienes que salen de la bodega. Adicionalmen-

te, la clase toma física registra el proceso de verificación física y conteo de los artículos en bodega.

- Proveedor: Representa las entidades que suministran artículos a la empresa, gestionan información relevante como RUC, nombre, dirección y contacto.
- Artículo: Contiene información detallada sobre cada artículo almacenado en la bodega, como nombre, stock, descripciones específicas. Adicionalmente, las clases bienes, materiales y suministros de oficina, representan los tipos de artículos.
- Bodega: Almacena información sobre las ubicaciones físicas donde se guardan los artículos.
- Departamento: Define las diferentes áreas de la empresa.
- Solicitud de egreso: Representa los diferentes tipos de solicitudes (bienes, materiales y suministros de oficina) que los funcionarios de la empresa pueden hacer dependiendo de sus necesidades específicas. Las solicitudes están asociadas con los departamentos que requieren egresos de la bodega.

3.2. Requisitos

Tras establecer el modelo de dominio, se puede partir con la definición y especificación de los requisitos del sistema, durante esta disciplina se toma el rol de analista de negocios. Esta disciplina de requisitos es crucial, ya que los requisitos funcionales y no funcionales determinan las expectativas del sistema, estableciendo un marco común de entendimiento entre las partes interesadas.

El proceso de identificación de requisitos se beneficia directamente de la comprensión obtenida a través de la definición del modelo de dominio, permitiendo una alineación con las necesidades reales del contexto del sistema y los objetivos empresariales.

Esta sección se centra en tres aspectos fundamentales del desarrollo del sistema: identificación de actores y casos de uso, especificación de casos de uso y creación de prototipos para la interfaz de usuario.

Se inicia con la identificación de los actores principales y los casos de uso para esto, se capturan las interacciones entre los usuarios y el sistema, así como también las necesidades específicas de la empresa.

A continuación, se detalla la especificación de los casos de uso, profundizando en los requisitos funcionales y estableciendo los límites del sistema. Esto establece las bases para comprender

el alcance y las expectativas del sistema, y facilita la estimación precisa de recursos necesarios para su implementación.

Finalmente, se concluye con el diseño de prototipos de interfaz de usuario que permiten validar y refinar los requisitos del sistema. Los prototipos permiten comunicar las funcionalidades planificadas a los stakeholders y asegurar que la visión del sistema concuerda con las expectativas del usuario final.

Todos estos artefactos mencionados no solo clarifican el alcance del sistema, sino que también constituyen una base esencial para la planificación de las fases subsiguientes de desarrollo bajo el enfoque RUP.

3.2.1. Identificación de Actores y Casos de Uso

En primer lugar, se identifica a los actores que interactúan con el sistema. En la Figura 3.2 se muestra el diagrama de jerarquía de actores con los diferentes actores que interactúan con el sistema. En este caso se tiene a los usuarios, organizados en varios roles dependiendo del departamento al que pertenecen. Cada uno de estos roles tienen un nivel de acceso distinto en el sistema y se los representa como actores.

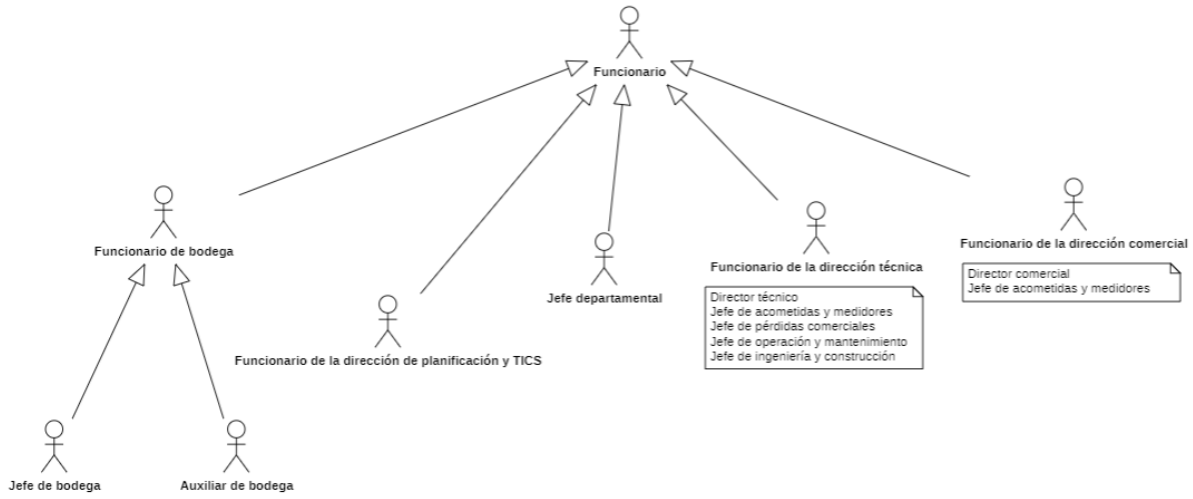


Figura 3.2: Diagrama de jerarquía de actores

A continuación, se presenta la Tabla 3.1 donde se muestra una descripción detallada de los actores identificados.

Actor	Descripción
Funcionario	Cualquier usuario que interactúa con el sistema, con la capacidad de realizar solicitudes a bodega.
Funcionario de bodega, jefe de bodega, auxiliar de bodega	Tipos de usuario encargado de la gestión de la bodega.
Jefe departamental, funcionario de la dirección técnica, comercial y planificación y TICS	Tipos de usuario con la posibilidad de aprobar solicitudes a bodega.

Tabla 3.1: Descripción de los actores del sistema

A partir de los actores identificados, se establecen los casos de uso que describen las diferentes actividades o funciones de negocio que serán soportadas por el sistema. Cada caso de uso representa un fragmento de funcionalidad del sistema y en concreto el modelo de casos de uso muestra la interacción entre los usuarios y el sistema.

El diagrama de casos de uso está organizado por paquetes, esta decisión responde a la cantidad de funcionalidades identificadas. Esta separación por paquetes permite una comprensión más clara y un mejor manejo de los casos de uso. Este diagrama ofrece una visión total de las funcionalidades del sistema agrupadas para reflejar los diferentes procesos que conforman la gestión de bodega. Se han identificado un total de 35 casos de uso.

En el paquete de la Figura 3.3, el *Funcionario de Bodega* juega un rol central, realizando actividades esenciales como la toma física para verificar la exactitud del inventario, la consulta y actualización de datos de artículos, y la gestión de solicitudes de compra. También se encarga de buscar y eliminar artículos según sea necesario, asegurando que la información del inventario esté siempre actualizada y sea relevante.

Además, los funcionarios tienen la capacidad de autorizar accesos a otros usuarios, garantizando que solo el personal autorizado pueda operar dentro del sistema. Las operaciones generales dentro de la bodega incluyen la generación de reportes de operaciones y de stock, que son críticos para el seguimiento y la planificación estratégica.

La gestión de operaciones abarca desde el ingreso hasta el egreso de bienes, facilitando la tramitación y la consulta de operaciones, así como la anulación de las mismas cuando sea necesario. Por otro lado, la administración de proveedores permite el registro, actualización, consulta y eliminación de proveedores.

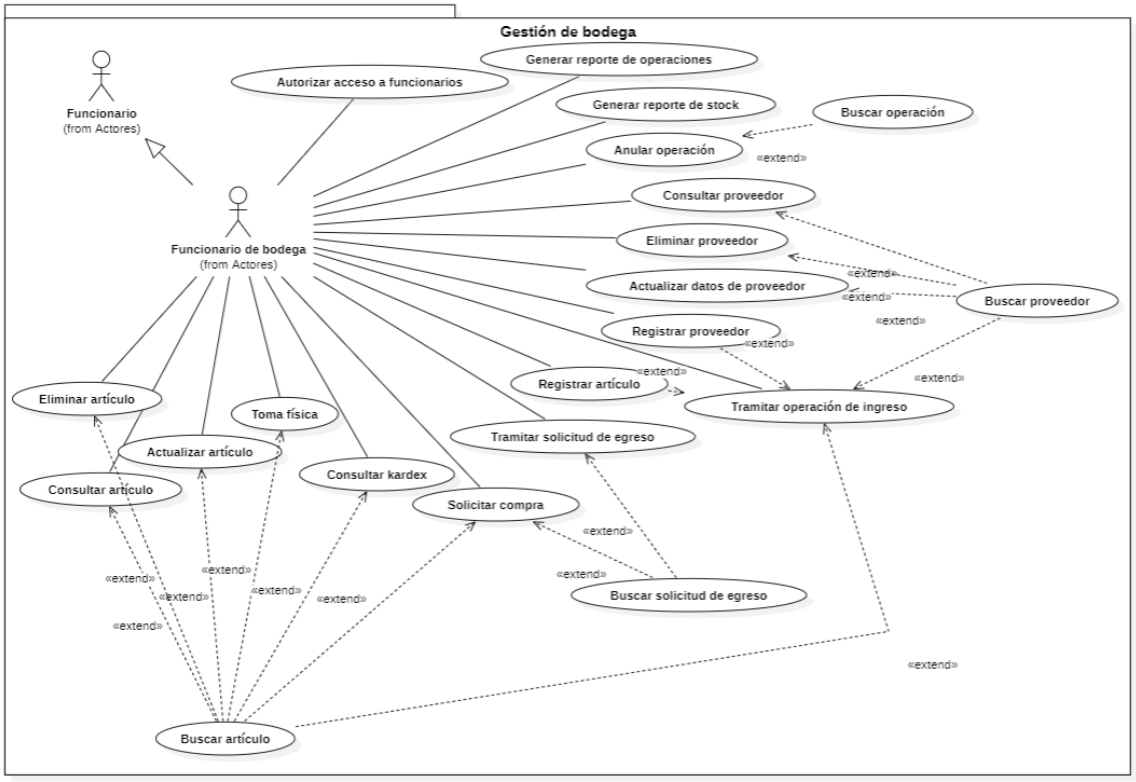


Figura 3.3: Diagrama de Casos de uso - Paquete: Gestión de bodega

Por otra parte, el paquete de solicitud de artículos de la Figura 3.4, ilustra el proceso de solicitar artículos, muestra que todos los funcionarios pueden gestionar sus necesidades de recursos, consultar el estado de sus solicitudes, y cancelar o buscar solicitudes específicas, pero haciendo diferencia entre los funcionarios de las direcciones técnica y comercial que solo ellos pueden solicitar materiales.

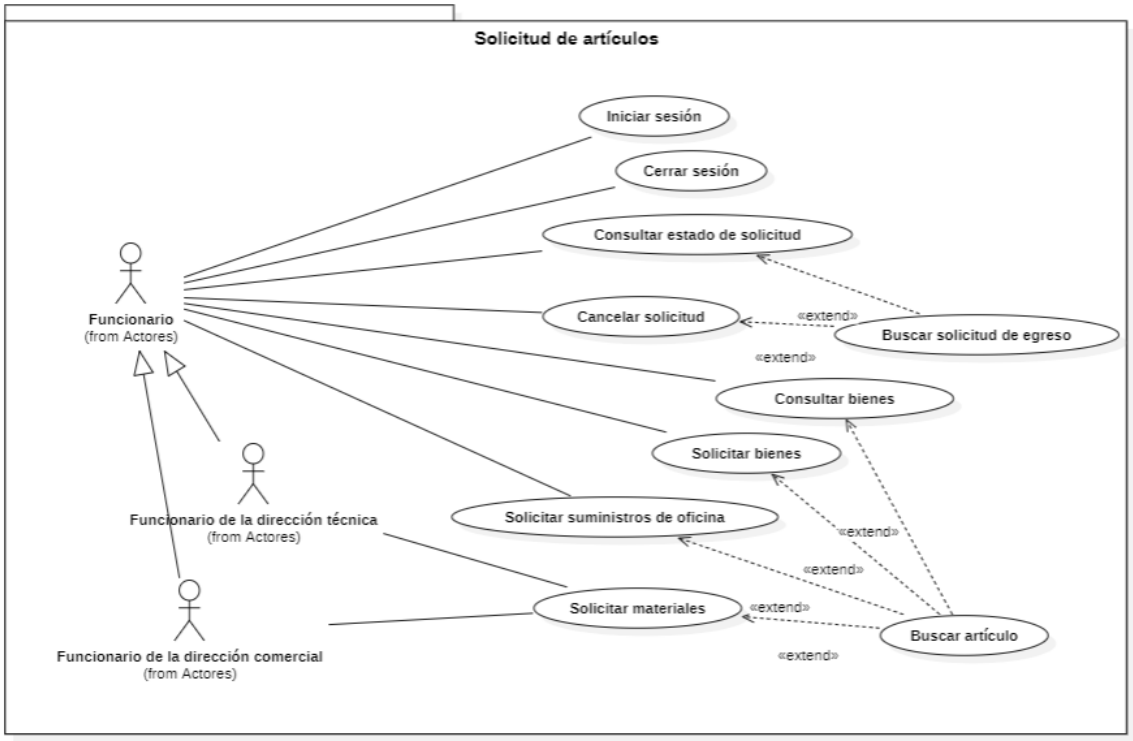


Figura 3.4: Diagrama de Casos de uso - Paquete: Solicitud de artículos

El último paquete del diagrama de la Figura 3.5 está centrado en la aprobación de solicitudes, se puede apreciar el flujo de trabajo para la autorización de recursos necesarios para las operaciones diarias. Adicionalmente se describen las responsabilidades asignadas a los jefes departamentales y a los funcionarios de las direcciones técnica y comercial para revisar y aprobar solicitudes de bienes, materiales y suministros de oficina según corresponda, para de esta forma garantizar que las demandas de cada departamento se manejen de manera eficiente y en concordancia con las políticas de la empresa.

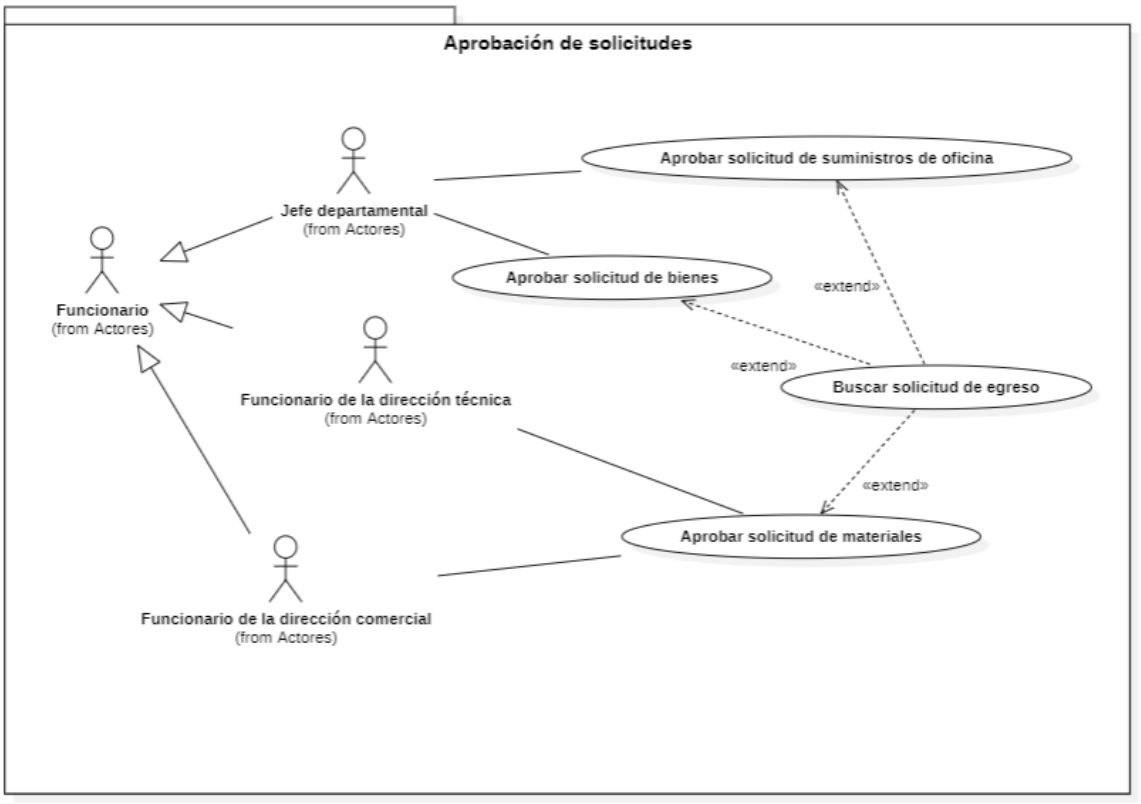


Figura 3.5: Diagrama de Casos de uso – Paquete: Aprobación de solicitudes

Adicionalmente, el diagrama de contexto de casos de uso de la Figura 3.6 al ser un diagrama de estados, muestra como los diferentes casos de uso, representados como transiciones, se relacionan a través de los diferentes estados durante la ejecución del sistema.

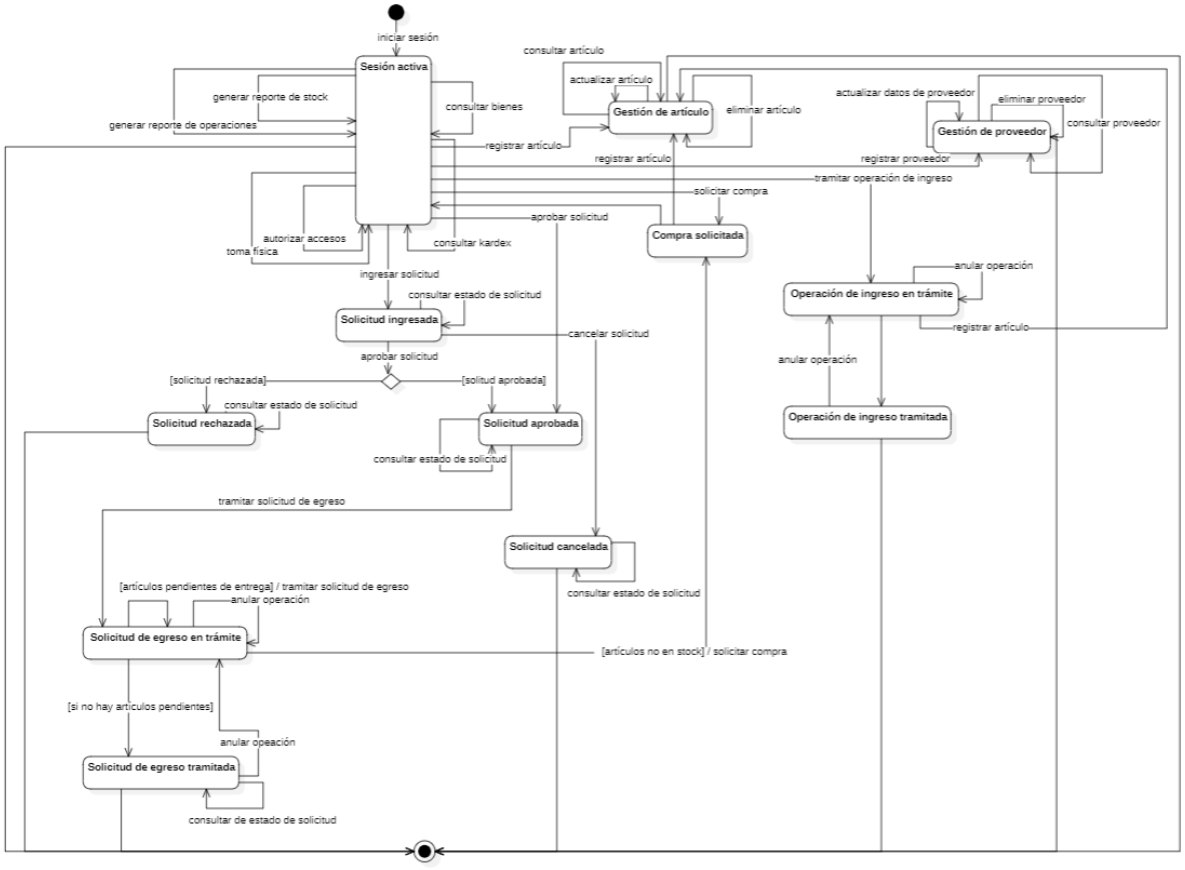


Figura 3.6: Diagrama de contexto de casos de uso

Estos diagramas no solo sirven como una herramienta de comunicación entre los desarrolladores y otros stakeholders, sino que también son críticos para tener una visión completa de los requisitos que debe satisfacer el sistema y la planificación de las iteraciones del sistema.

3.2.2. Especificación de Casos de Uso

Cada caso de uso representa una funcionalidad del sistema, considerando los 35 casos de uso en total, para evitar superposiciones innecesarias se optó por consolidar los casos de uso *Solicitar bienes*, *Solicitar materiales* y *Solicitar suministros de oficina*, en un único caso de uso denominado *Ingresar solicitud*. Similarmente, los casos *Aprobar solicitud de bienes*, *Aprobar solicitud de materiales* y *Aprobar solicitud de suministros de oficina*, se unificó bajo el nombre de *Aprobar solicitud*. Esta decisión reduce la redundancia y contempla la gestión eficiente de las solicitudes a bodega, ya que todas podrán ser procesadas como variantes de un tipo de solicitud con procedimientos ligeramente diferenciados.

Dada la extensión de los casos de uso identificados se incluyen en esta documentación solo

aquellas especificaciones que son cruciales para la comprensión del lector sobre la funcionalidad del sistema. Para ello, para especificar los casos de uso se han utilizado diagramas de estados. Esta técnica de modelado visual no solo facilita la descripción de interacciones entre los actores y el sistema, sino muestra de manera clara los flujos de interacciones alternativos y de excepción en los casos de uso.

Los diagramas de estado construidos en este trabajo constan de un estado inicial, un estado final, y la secuencia de interacciones que definen el flujo básico que está de color verde, y la secuencia de interacciones para flujos alternativos en color rojo. A continuación, se presentan las especificaciones de los casos de uso de mayor prioridad en el contexto del sistema a construir.

Dentro de la gestión de bodega se cuentan con dos operaciones bien definidas: ingreso y egreso. En las Figuras 3.7, 3.8, 3.9 y 3.10, se presentan los diagramas de estados que describen estos procesos, comenzando con el caso de uso *Tramitar operación de ingreso*, el cual describe detalladamente el procedimiento a realizar cuando se da un ingreso a bodega. Para visualizar la operación de egreso, se presentan los casos de uso *Ingresar solicitud*, *Aprobar solicitud* y finalmente *Tramitar operación de egreso*. La Tabla 3.2 presenta los casos de uso a ser expuestos.

Identificador	Caso de uso
CU1	Tramitar operación de ingreso
CU2	Ingresar solicitud
CU3	Aprobar solicitud
CU4	Tramitar operación de egreso

Tabla 3.2: Casos de uso

La Figura 3.7 muestra el diagrama de estados para el caso de uso CU1: *Tramitar operación de ingreso*, detallando el flujo de actividades desde una sesión activa hasta la confirmación y registro de una operación de ingreso.

El proceso comienza con el funcionario solicitando tramitar una operación de ingreso, el sistema requiere información como el origen del ingreso, fecha, número de factura, orden de compra, código de proveedor y observaciones. El funcionario puede buscar y registrar proveedores y artículos, proporcionando los códigos y detalles necesarios, tras seleccionar los artículos y especificar las cantidades, el funcionario puede confirmar o cancelar el ingreso. Finalmente, el sistema genera un comprobante de ingreso, completando la operación y cambiando el estado a Operación de ingreso registrada.

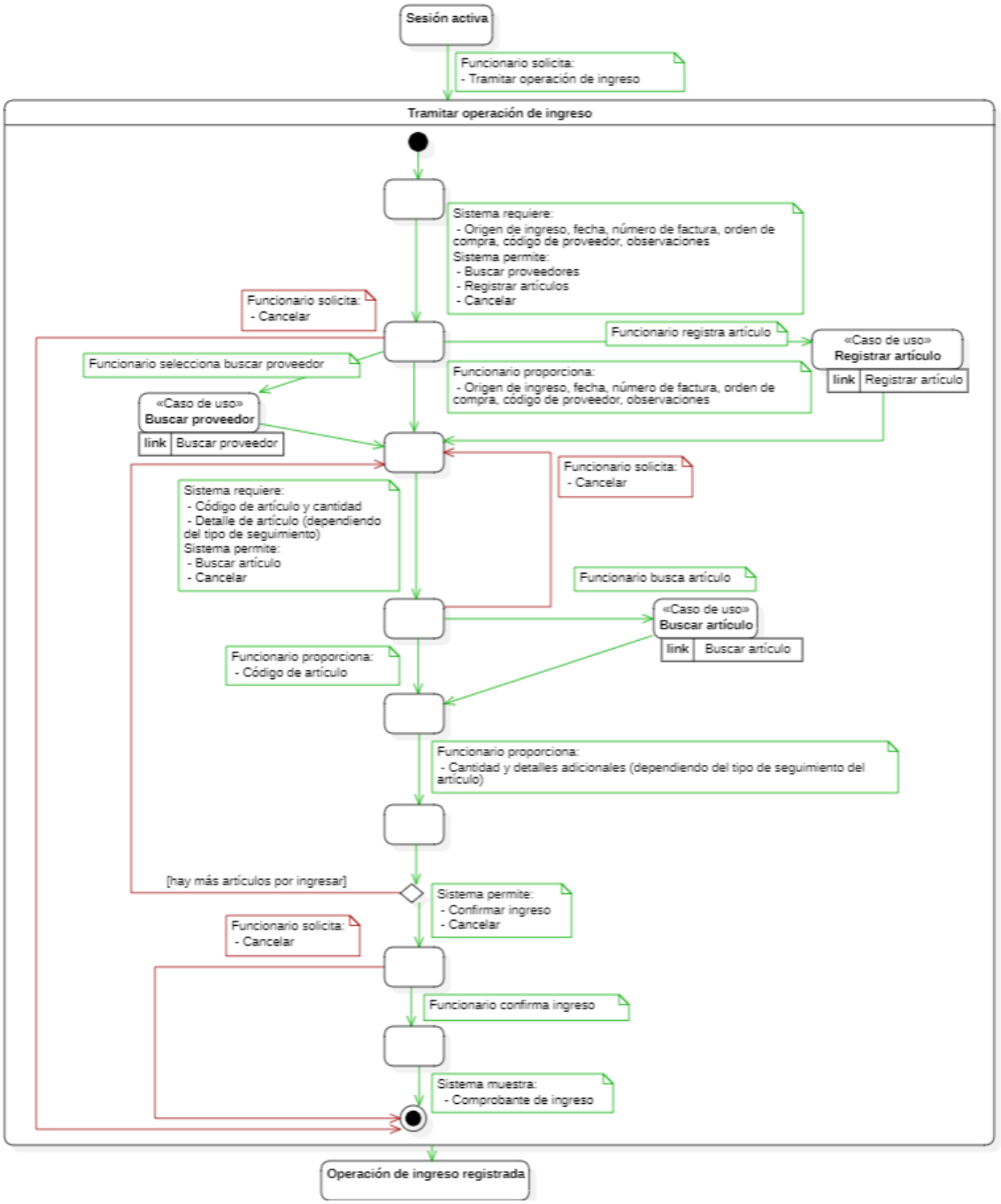


Figura 3.7: Diagrama de estados del caso de uso CU1: Tramitar operación de ingreso

Por otra parte, para mostrar la operación de egreso (ver Figura 3.8), la cual presenta el diagrama de estados para el caso de uso CU2: *Ingresar solicitud*. Dicho diagrama muestra cómo se parte de una Sesión activa, el funcionario inicia el proceso seleccionando y completando los detalles requeridos para una nueva solicitud, incluyendo el tipo de solicitud, la fecha y observaciones relevantes. Posteriormente, tiene la opción de buscar y seleccionar artículos específicos, proporcionando las cantidades deseadas. Una vez todos los ítems son confirmados, el sistema genera un comprobante de la solicitud, marcando la transición al estado final

de Solicitud ingresada.

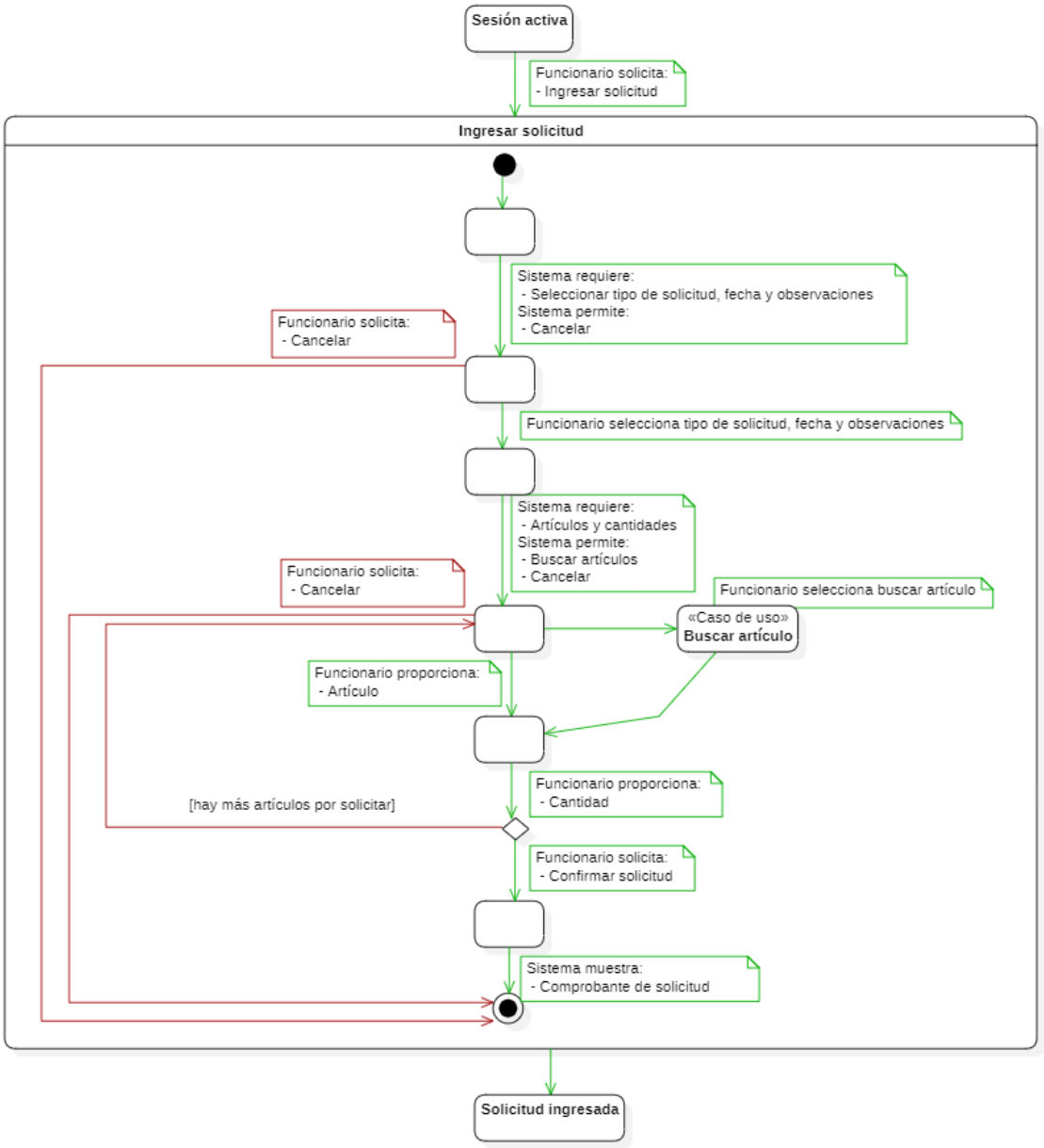


Figura 3.8: Diagrama de estados del caso de uso CU2: Ingresar solicitud

La Figura 3.9 muestra la especificación del caso de uso CU3: *Aprobar solicitud*, que ilustra el flujo detallado de acciones que un funcionario realiza para gestionar solicitudes de egreso dentro del sistema. Desde el estado inicial de Solicitud ingresada, el funcionario puede iniciar la acción de aprobar una solicitud, para lo cual el sistema requiere el número de solicitud y la fecha correspondiente. El funcionario tiene la opción de buscar solicitudes específicas de egreso o cancelar la acción en cualquier momento. Una vez encontrada la solicitud, el sistema muestra detalles completos de la misma, permitiendo al funcionario aprobar o rechazar la solicitud, y añadir comentarios opcionales si se desea. La decisión tomada conduce a uno de los estados

finales: Solicitud aprobada o Solicitud rechazada.

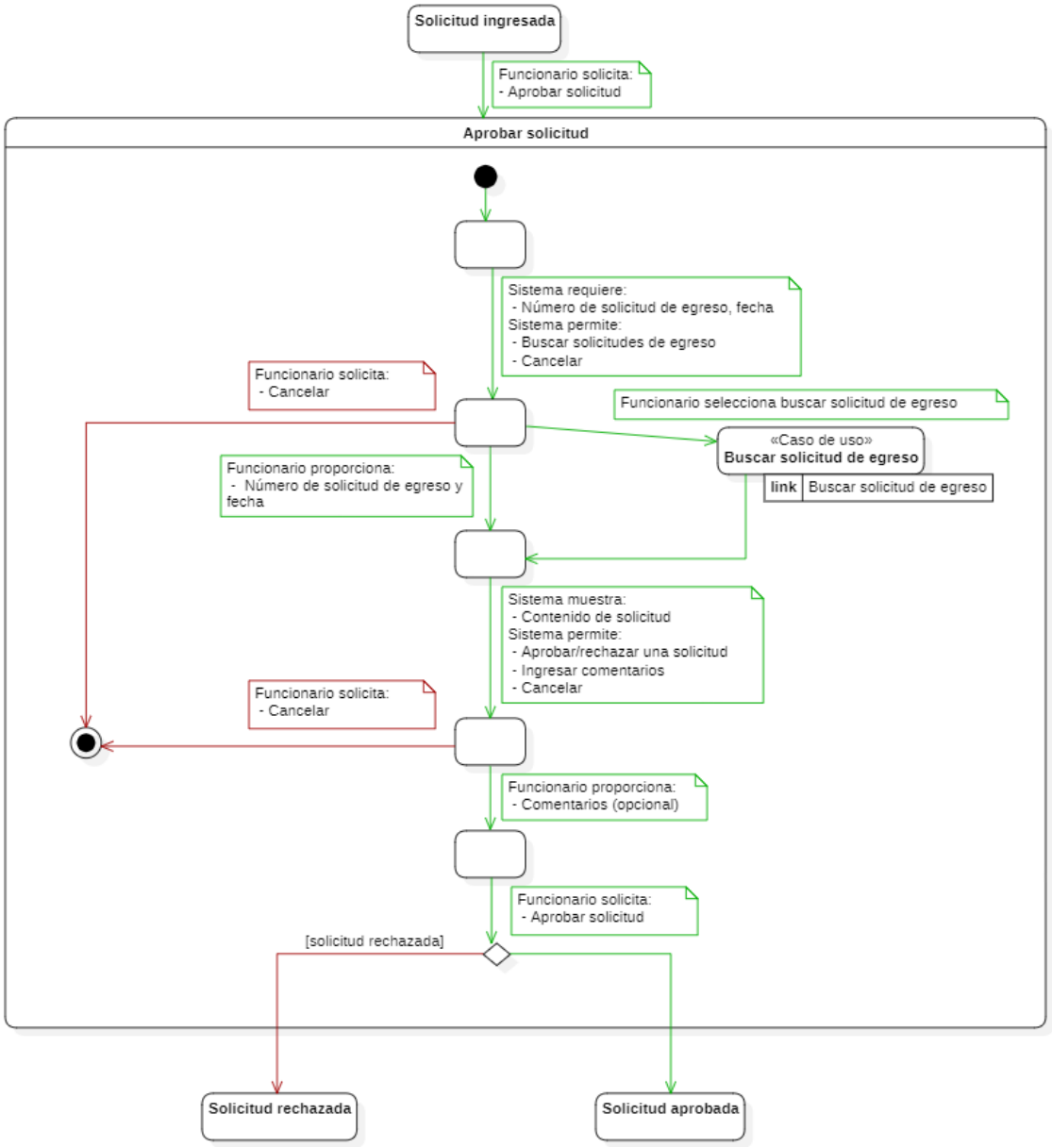


Figura 3.9: Diagrama de estados del caso de uso CU3: Aprobar solicitud

Finalmente, para completar el proceso de la operación de egreso está la Figura 3.10 que muestra el diagrama de estados para el caso de uso CU4: *Tramitar solicitud de egreso*, detallando el proceso desde que una solicitud es aprobada hasta que se completa su trámite. El flujo comienza con el funcionario seleccionando una solicitud de egreso aprobada. El sistema muestra las solicitudes pendientes de tramitar, incluyendo detalles como el número, tipo de solicitud, solicitante, y fecha de aprobación. El funcionario proporciona el número de solicitud y la fecha correspondiente, y el sistema despliega la lista de artículos solicitados junto con sus cantidades disponibles en stock. El funcionario introduce la cantidad de artículos a entregar y puede

confirmar el egreso si los artículos están en stock o solicitar una compra si no hay suficiente inventario. Finalmente, el sistema genera un comprobante de egreso, completando el proceso y llevando el estado a Solicitud de egreso tramitada o Compra solicitada en caso de falta de stock.

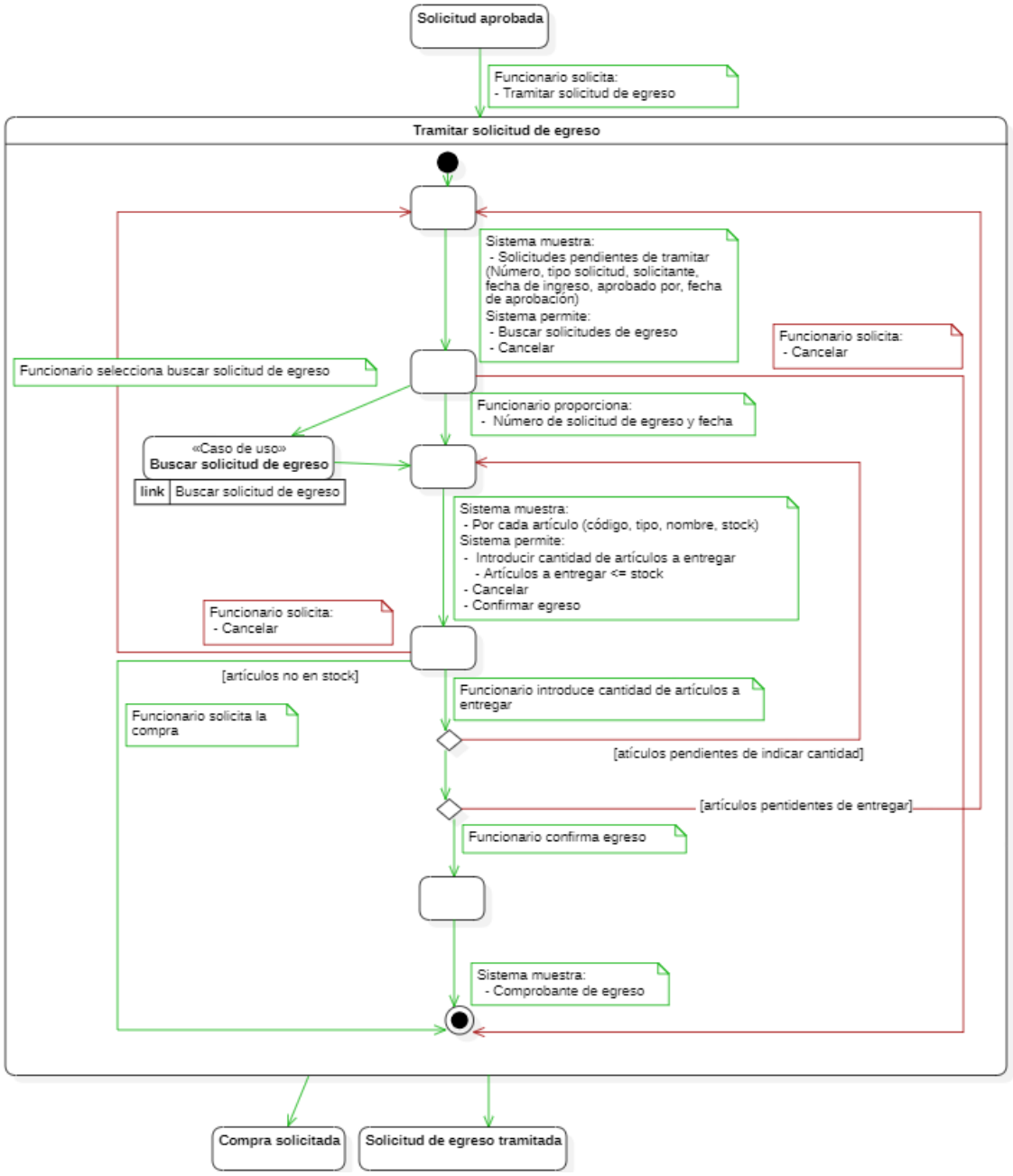


Figura 3.10: Diagrama de estados del caso de uso CU4: Tramitar operación de egreso

3.2.3. Prototipos de Interfaz de Usuario

En esta sección se diseñaron las interfaces de usuario considerando los casos de uso definidos como prioritarios. A continuación, se presentan algunos de los prototipos de interfaces de

usuario realizados, los cuales reflejan las interacciones entre los actores y el sistema, dichos prototipos fueron realizados con la herramienta draw.io (“draw.io”, s.f.).

La Figura 3.11 proporciona una vista general de las principales interfaces del sistema y sus relaciones.

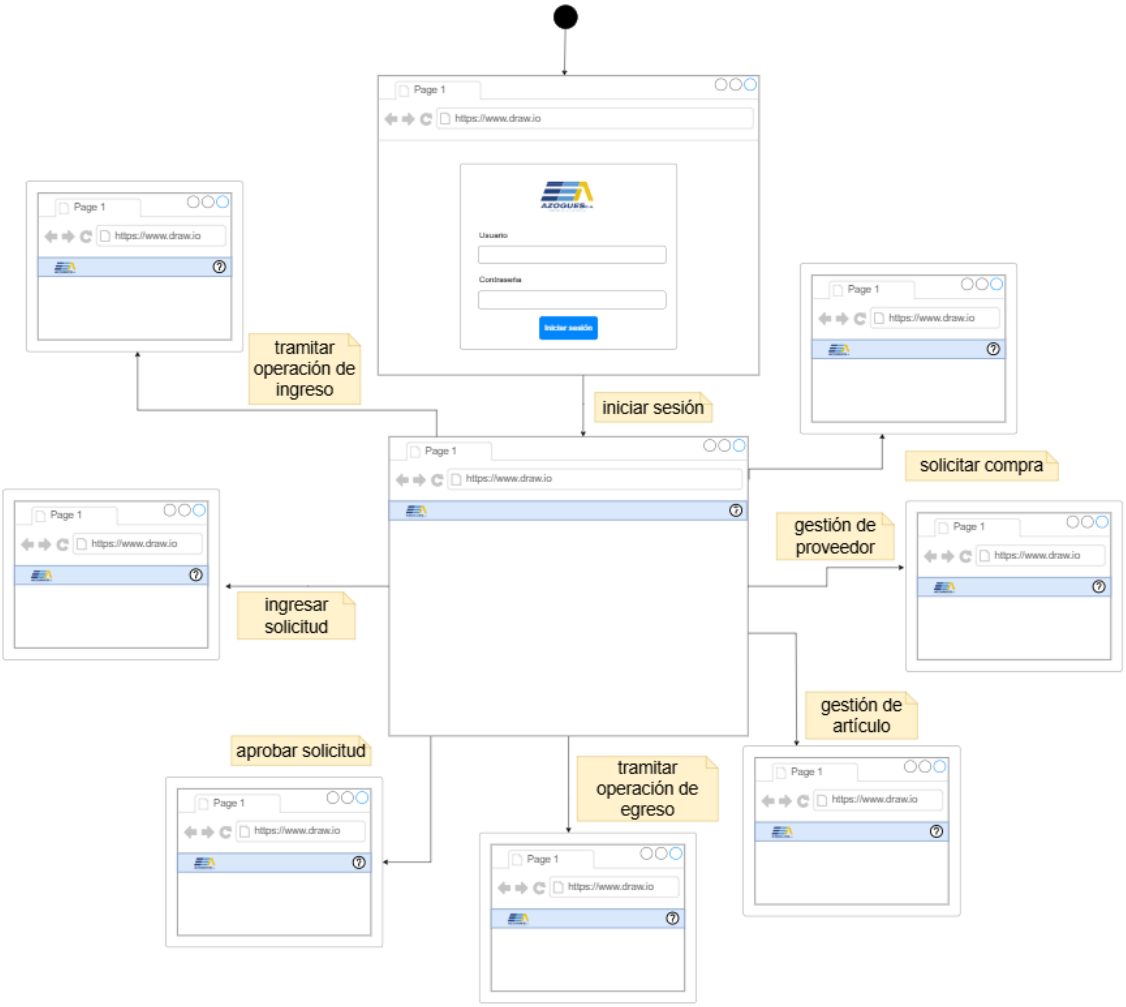


Figura 3.11: Prototipos de interfaces de usuario del sistema de gestión de bodega

Como en la sección de especificación de casos de uso se describió los casos de uso que detallaban los procesos de ingreso y egreso de bodega, en este apartado para continuar con la línea de explicación se adjuntan los prototipos de interfaz de usuario de los mismos casos de uso.

El prototipo de interfaz de usuario para el caso de uso de *Tramitar operación de ingreso* se muestra en la Figura 3.12.

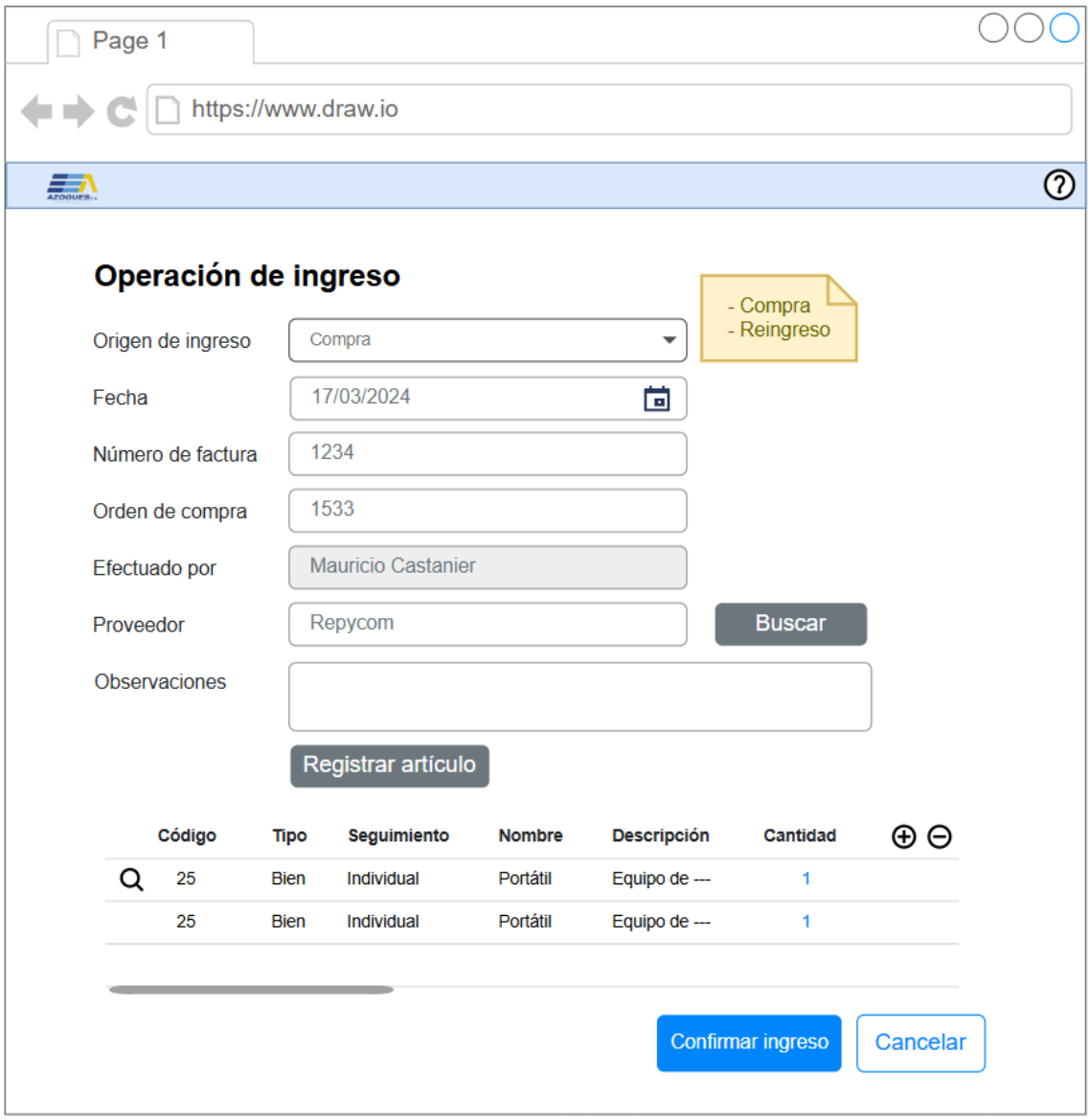


Figura 3.12: Prototipos de interfaz de usuario para el caso de uso CU1: Tramitar operación de ingreso

La Figura 3.13 muestra los prototipos de interfaz de usuario para el caso de uso de *Ingresar solicitud*.

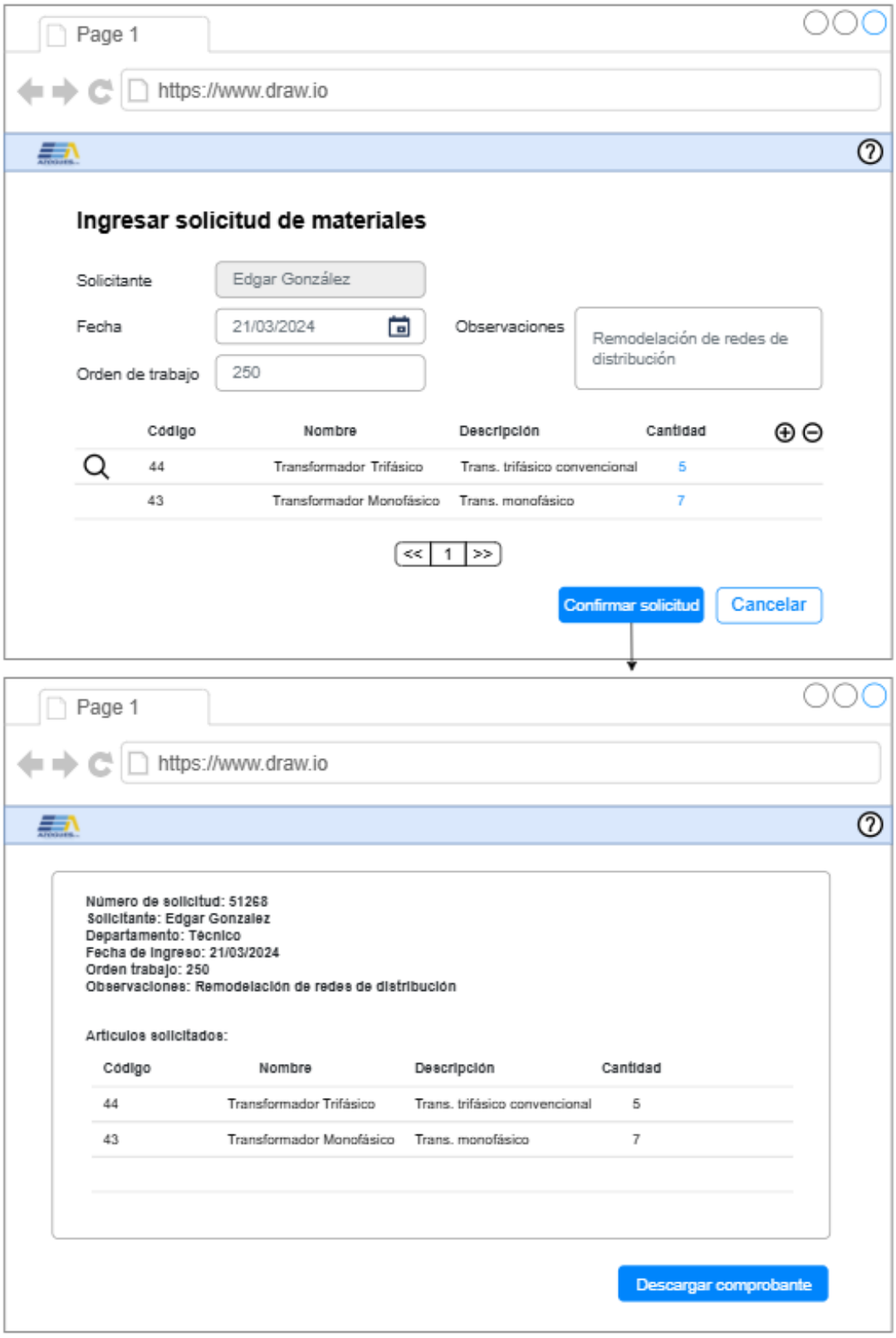


Figura 3.13: Prototipos de interfaz de usuario para el caso de uso CU2: Ingresar solicitud

La Figura 3.14 muestra el caso de uso de *Aprobar solicitud*.

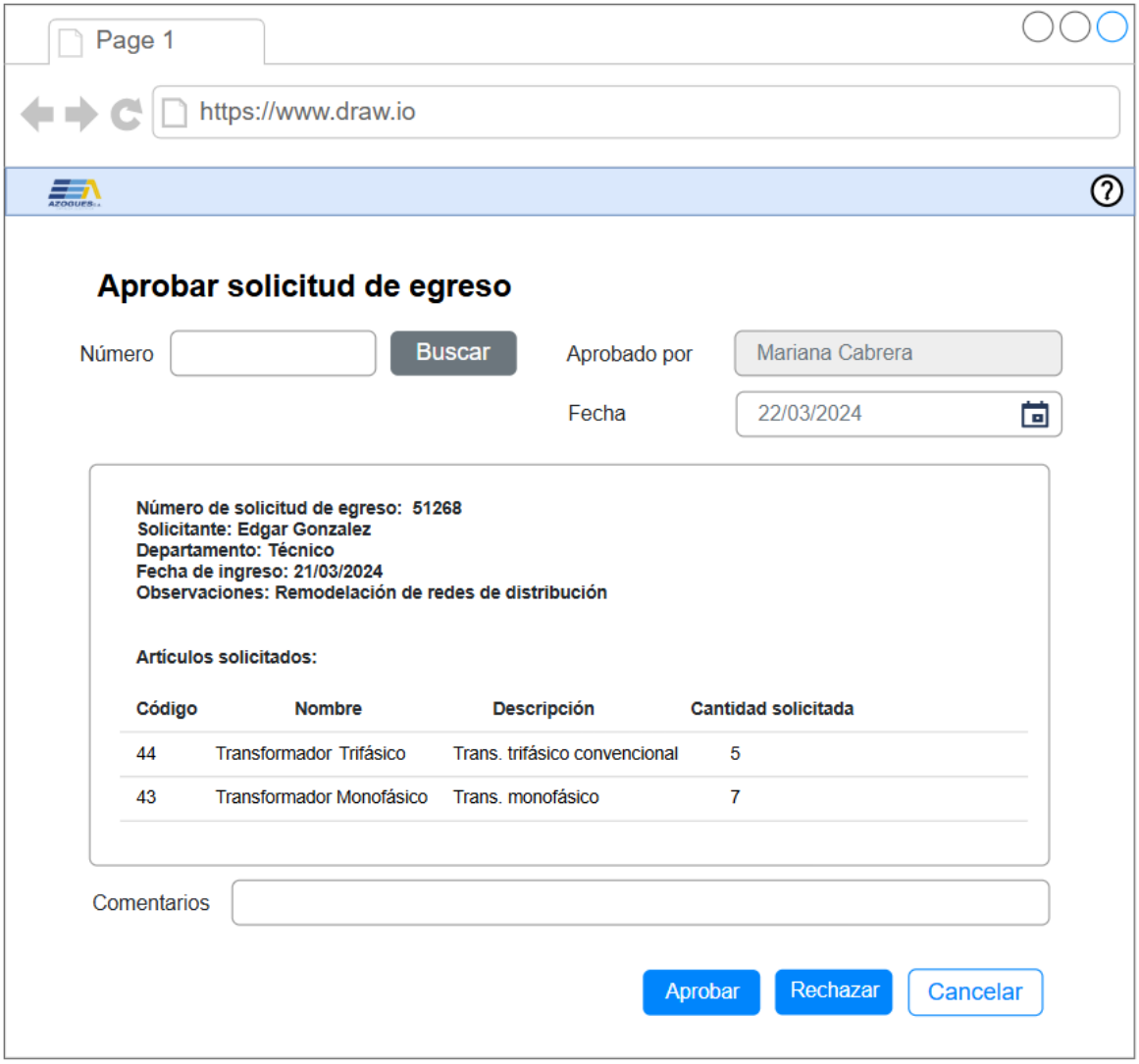


Figura 3.14: Prototipos de interfaz de usuario para el caso de uso CU3: Aprobar solicitud

Para representar el proceso de las operaciones de egreso se puede ver la Figura 3.15.

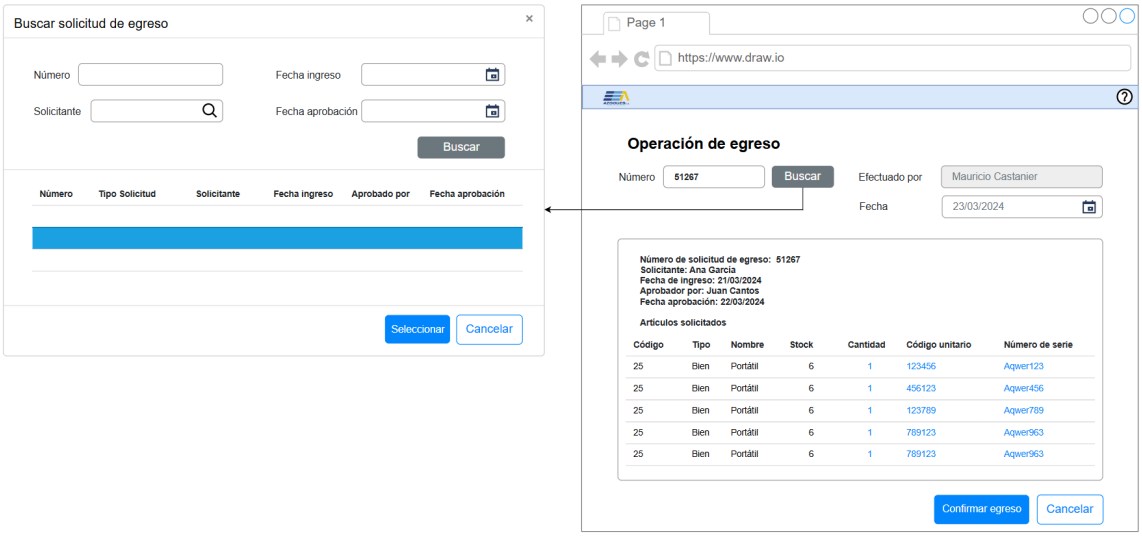


Figura 3.15: Prototipos de interfaz de usuario para el caso de uso CU4: Tramitar operación de egreso

3.2.4. Requisitos especiales

Los requisitos especiales, en este contexto hacen referencia a una descripción detallada de los requisitos no funcionales que están relacionados con los casos de uso identificados para el sistema de gestión de bodega en la Empresa Eléctrica Azogues C.A. Estos requisitos definen parámetros de calidad que aseguran la usabilidad, eficiencia, seguridad, y robustez del sistema. Estos parámetros son esenciales en los flujos de trabajo de análisis, diseño e implementación, proporcionando pautas para el desarrollo de un sistema que cumpla con las funcionalidades esperadas y garantice un rendimiento óptimo y una experiencia de usuario satisfactoria.

Entre los requisitos no funcionales más relevantes para este proyecto se encuentran el rendimiento, que asegura tiempos de respuesta rápidos durante las transacciones; la seguridad, que protege la integridad y la privacidad de los datos del sistema; la disponibilidad, que garantiza un acceso constante al sistema con mínimas interrupciones; la usabilidad, que facilita la interacción del usuario con el sistema; la escalabilidad, que permite al sistema crecer según las necesidades de la empresa; la interoperabilidad, que asegura la integración con otros sistemas; y el mantenimiento, que promueve una gestión eficiente a largo plazo del sistema. A continuación, se enumeran los requisitos no funcionales específicos para este proyecto; mismos que están descritos como metas que luego serán especificadas de tal manera que sean medibles y verificables:

- Rendimiento:

- El sistema debe ser capaz de procesar transacciones de ingreso y egreso de bodega en menos de dos segundos.
- La base de datos debe soportar realizar consultas simultáneas sin degradación significativa del rendimiento.

- Seguridad:
  - El sistema de gestión de bodega debe garantizar la protección del acceso mediante una autenticación y autorización robusta.
  - Los datos sensibles deben ser encriptados en tránsito y en reposo.
- Disponibilidad:
  - El sistema debe garantizar una disponibilidad del 99.9 %.
- Usabilidad:
  - La interfaz de usuario debe ser intuitiva y fácil de usar, debe permitir a los usuarios realizar sus tareas sin necesidad de extensas capacitaciones.
- Interoperabilidad:
  - El sistema debe facilitar la exportación e importación de datos en formatos estándar.
- Mantenimiento
  - El sistema debe ser fácil de mantener y actualizar, con una arquitectura modular que permita la implementación de cambios sin afectar otras partes del sistema.
  - Debe incluir documentación completa y actualizada para desarrolladores y administradores del sistema.

3.3. Análisis y Diseño

Durante la disciplina de Análisis y Diseño se toman los roles de arquitecto de software y diseñador de software respectivamente, el propósito principal de esta disciplina es convertir los requisitos del sistema en una especificación de cómo implementar el sistema. En las siguientes secciones, se aborda primero el análisis de la arquitectura del sistema, donde se define la estructura general y se establecen los cimientos sobre los cuales se construirán los elementos del sistema. Luego, se procede a examinar los casos de uso mediante la elaboración de

diagramas de clases y de colaboración, que ilustran la interacción entre los diversos componentes del sistema en la realización de las funciones requeridas. Seguidamente, se analizan las clases con la finalidad de identificar las responsabilidades, atributos y requisitos especiales. Finalmente, se analizan los paquetes que agrupan las clases y componentes relacionados, ofreciendo una visión organizada y modular del sistema.

3.3.1. Análisis de la Arquitectura

Se establece la organización del sistema, considerando tres paquetes, que agrupan las clases de modelos, vistas y controladores. Esta estructura modular facilita el desarrollo, mantenimiento y escalabilidad del sistema de gestión de bodega, asegurando una clara separación de responsabilidades. A continuación, se puede apreciar el diagrama de paquetes del análisis de la arquitectura en la Figura 3.16.

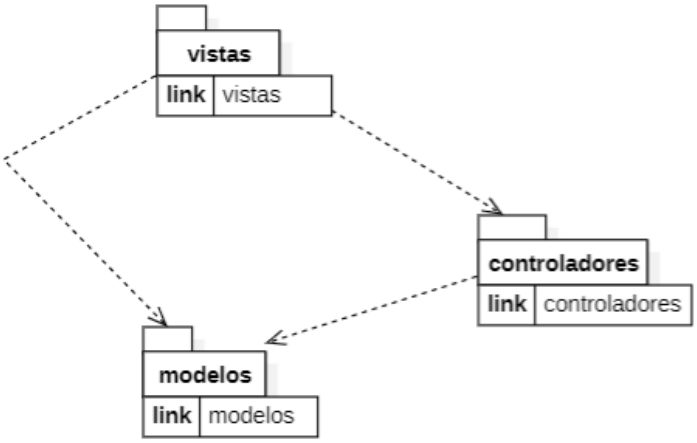


Figura 3.16: Diagrama de paquetes del análisis de arquitectura

Las clases de modelo son tomadas del modelo de dominio, con la diferencia que estas reflejan la información de forma que beneficia al diseño e implementación del sistema, como se puede ver en la Figura 3.17 se especifican las clases identificadas, sumando un total de 19 clases que abarcan las operaciones esenciales de la bodega, incluyendo ingresos, egresos, manejo de inventarios y solicitudes. Esta selección de clases asegura que todos los aspectos críticos de la gestión de bodega, desde la recepción hasta la distribución de artículos, estén adecuadamente representados y gestionados dentro del sistema.

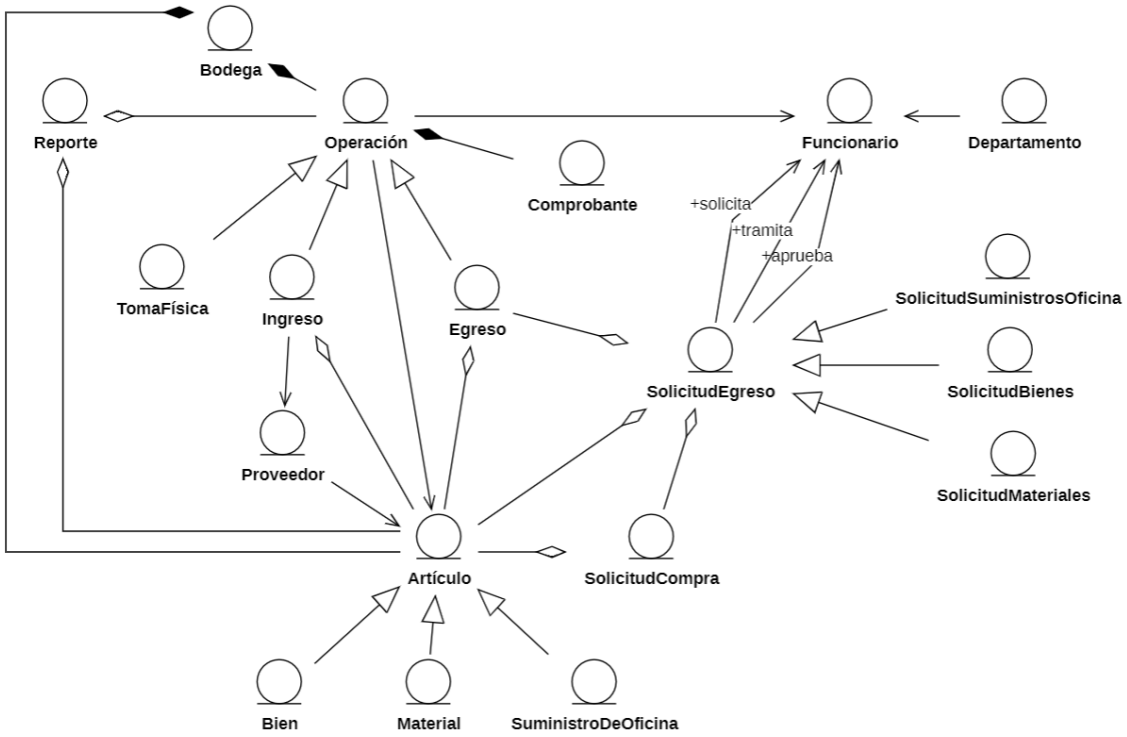


Figura 3.17: Diagrama de clases de las clases modelo

Las clases vistas son usadas para representar la interacción entre los actores y el sistema, estas se mantienen en un nivel alto ya que son abordadas en los flujos de trabajo siguientes. En la Figura 3.18 se especifican las clases vistas identificadas, cada una diseñada para facilitar tareas específicas dentro del sistema de gestión de bodega.

Figura 3.18: Diagrama de clases de las clases vistas

Finalmente, las clases controladoras manejan y coordinan las secuencias de acciones y flujos de control principales, son usadas para encapsular el control de cada caso de uso, también se utilizan para representar la lógica de negocio que implica a varios objetos de las clases de modelo y vista. En la Figura 3.19, se especifican las clases controladoras identificadas, cada

una encargada de gestionar procesos dentro del sistema de gestión de bodega. Inicialmente se consideró un controlador por cada caso de uso, sin embargo, luego de algunas iteraciones se obtuvo finalmente las clases que ilustra la Figura 3.19. Estas incluyen desde la tramitación de operaciones de ingreso y egreso hasta la generación de reportes y la gestión de proveedores.

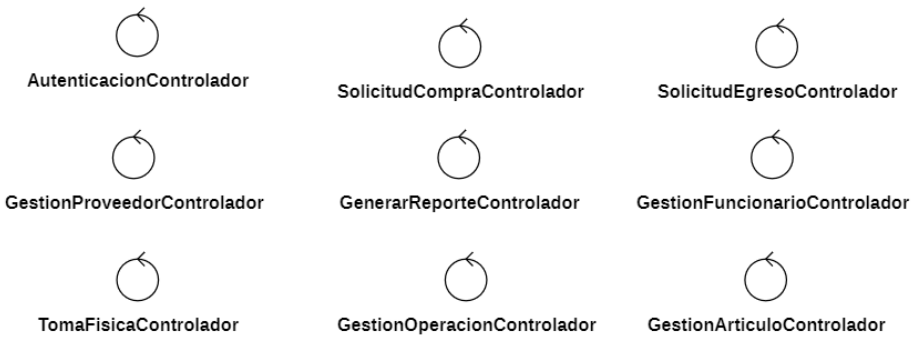


Figura 3.19: Diagrama de clases de las clases controladores

3.3.2. Análisis de Casos de Uso

Se identifican las clases de análisis: modelo, vista y controlador, necesarias para realizar cada caso de uso, además se esbozan las responsabilidades, atributos y relaciones de dichas clases, para ello se usó como apoyo a los diagramas de clases, posterior a esto se describen las interacciones entre objetos de análisis mediante diagramas de colaboración. En conjunto, ambos diagramas proporcionan una visión completa tanto de la estructura estática (diagrama de clases) como de la dinámica (diagrama de colaboración) de las interacciones necesarias para realizar los distintos casos de uso.
De la misma forma que se presentó en el capítulo de requisitos, en esta sección se incluyen los diagramas correspondientes a los casos de uso: *Tramitar operación de ingreso, Ingresar solicitud, Aprobar solicitud, Tramitar operación de egreso*; para proporcionar una representación visual clara y detallada de las interacciones y secuencias de acciones involucradas específicamente de las operaciones de ingreso y egreso.

CU1: Tramitar operación de ingreso

El diagrama de clases de la Figura 3.20 ilustra las clases involucradas y sus relaciones. En este diagrama se identifican las siguientes clases:

- Funcionario de bodega: Representa el actor que interactúa con el sistema para tramitar la operación de ingreso.

- BuscarArtículoVista: Esta clase de vista permite al funcionario buscar artículos.
- MenuVista: Clase de vista que proporciona el menú de opciones al funcionario.
- TramitarOperacionIngresoVista: Clase de vista que permite tramitar la operación de ingreso.
- BuscarProveedorVista: Clase de vista que permite buscar proveedores.
- RegistrarArticuloVista: Clase de vista que facilita el registro de artículos nuevos.
- GestionOperacionControlador: Clase controladora que maneja la lógica de negocio para tramitar una operación de ingreso.
- Artículo: Clase de modelo que representa los artículos en el inventario.
- Ingreso: Clase de modelo que registra los detalles de las operaciones de ingreso.
- Proveedor: Clase de modelo que representa a los proveedores.
- Funcionario: Clase de modelo que representa a los funcionarios de la bodega.

Estas clases interactúan para permitir al funcionario de bodega registrar una nueva operación de ingreso, buscando proveedores y artículos, y registrando los detalles del ingreso.

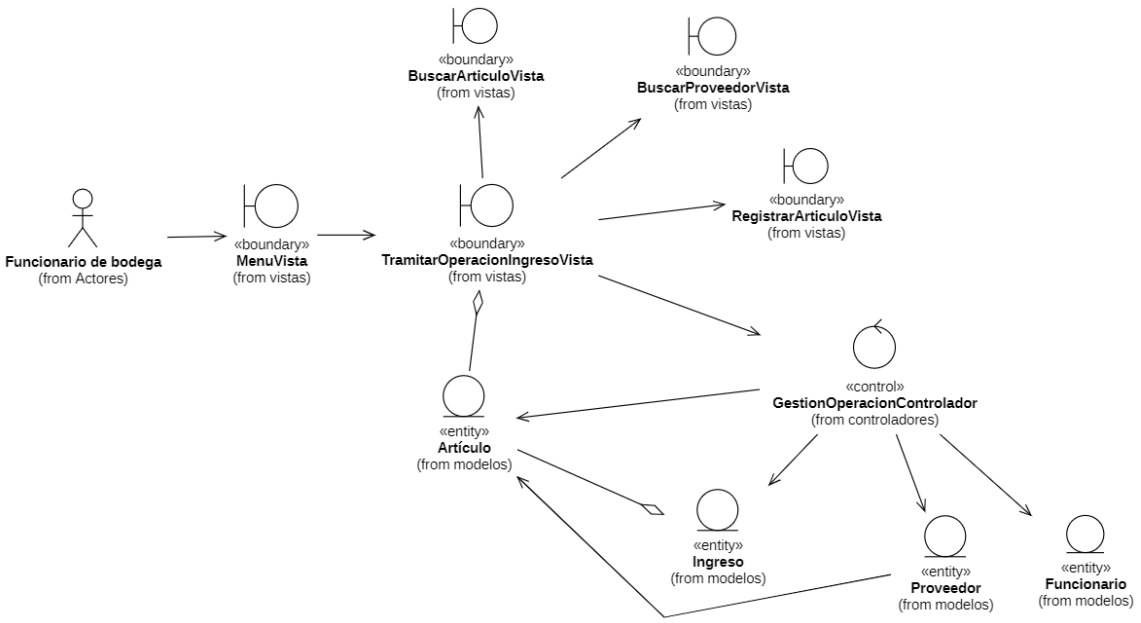


Figura 3.20: Diagrama de clases del caso de uso CU1: Tramitar operación de ingreso

El diagrama de colaboración de la Figura 3.21 muestra las interacciones dinámicas entre los objetos de las clases involucradas en el caso de uso *Tramitar operación de ingreso*. A continuación, se describen los pasos representados en el diagrama:

- Inicio de la operación: El funcionario de bodega abre el menú y selecciona la opción para tramitar la operación de ingreso.
- Búsqueda de Proveedor: El funcionario solicita buscar un proveedor mediante la vista BuscarProveedorVista, sobre esta vista se puede seleccionar búsqueda por RUC o nombre del proveedor.
- Registro de Artículo: El funcionario busca artículos mediante la vista BuscarArticuloVista. Una vez seleccionado el artículo, se utiliza la vista RegistrarArticuloVista para registrar los detalles del artículo en la operación de ingreso.
- Proceso de Ingreso: La clase GestionOperacionControlador valida y procesa los datos ingresados. Se registran los detalles del proveedor, artículos y el ingreso propiamente dicho. Por último, se confirma la operación, y se genera un comprobante de ingreso.

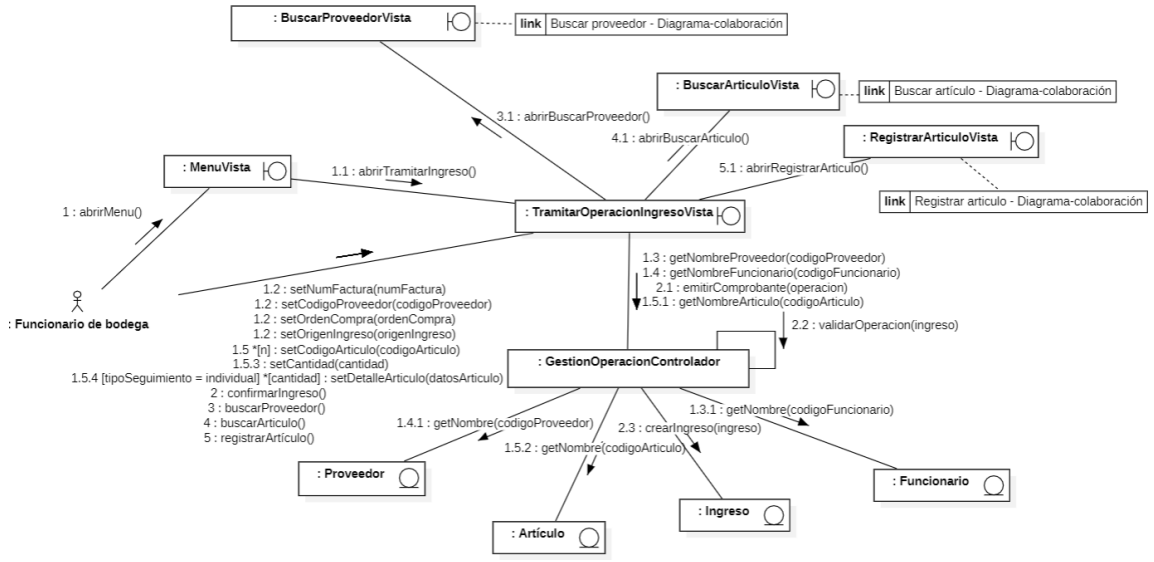


Figura 3.21: Diagrama de colaboración del caso de uso CU1: Tramitar operación de ingreso

CU2: Ingresar solicitud

El diagrama de la Figura 3.22 muestra la interacción entre clases, que permiten al funcionario ingresar una nueva solicitud de egreso, buscar artículos y registrar los detalles de la solicitud.

- Funcionario: El actor que interactúa con el sistema para ingresar una nueva solicitud.
- MenuVista: Clase de vista que proporciona el menú de opciones al funcionario.
- IngresarSolicitudEgresoVista: Clase vista que permite ingresar los detalles de una nueva solicitud de egreso.

- BuscarArticuloVista: Esta case permite al funcionario buscar artículos.
- SolicitudEgresoControlador: Esta clase maneja la lógica de negocio para ingresar la solicitud de egreso.
- Artículo: Clase de modelo que representa los artículos en el inventario.
- SolicitudEgreso: Clase de modelo que registra los detalles de las solicitudes de egreso.
- Funcionario: Clase de modelo que representa a los funcionarios.

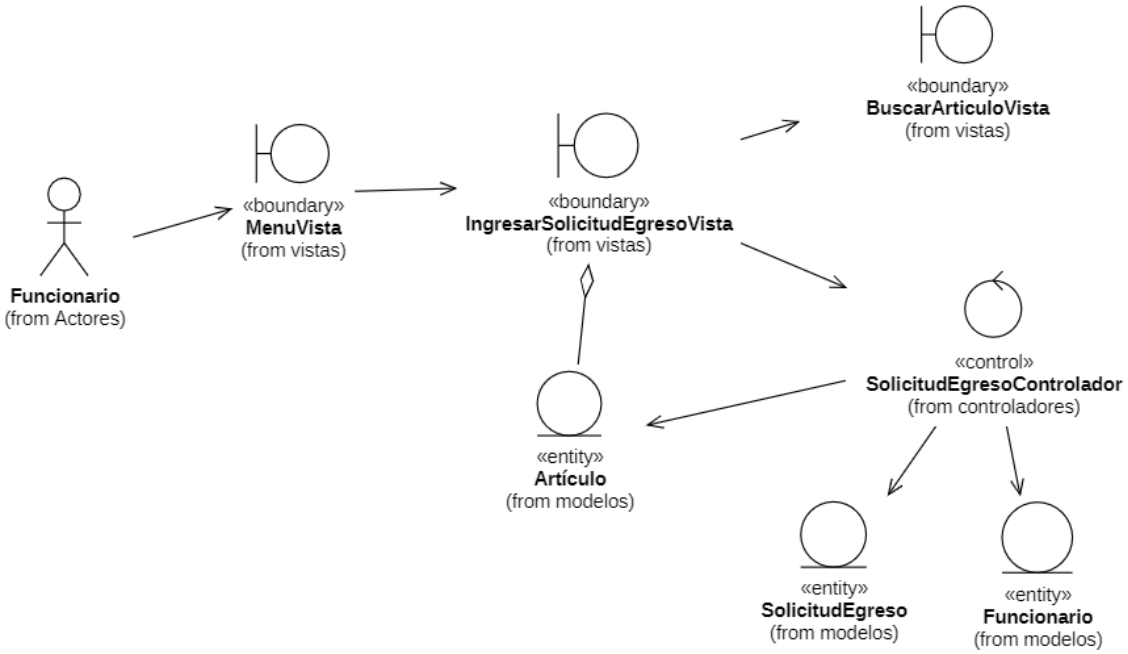


Figura 3.22: Diagrama de clases del caso de uso CU2: Ingresar solicitud

Adicionalmente, a continuación, se describen los pasos representados en el diagrama de colaboración de la Figura 3.23:

- Inicio de la solicitud: El funcionario de bodega abre el menú y selecciona la opción para ingresar una solicitud de egreso.
- Ingreso de Detalles de la Solicitud: El funcionario utiliza la vista IngresarSolicitudEgresoVista para ingresar los detalles de la solicitud, como el código del funcionario, la fecha, el tipo de solicitud, y las observaciones.
- Búsqueda de Artículo: El funcionario solicita buscar un artículo, iniciando la vista BuscarArticuloVista. La vista de artículo permite buscar y seleccionar un artículo, que se registra en la vista IngresarSolicitudEgresoVista.

- Proceso de Solicitud: La clase SolicitudEgresoControlador valida y procesa los datos ingresados. Se registran los detalles del funcionario, artículos y la solicitud de egreso. Como último paso se confirma la operación, y se genera un comprobante de la solicitud.

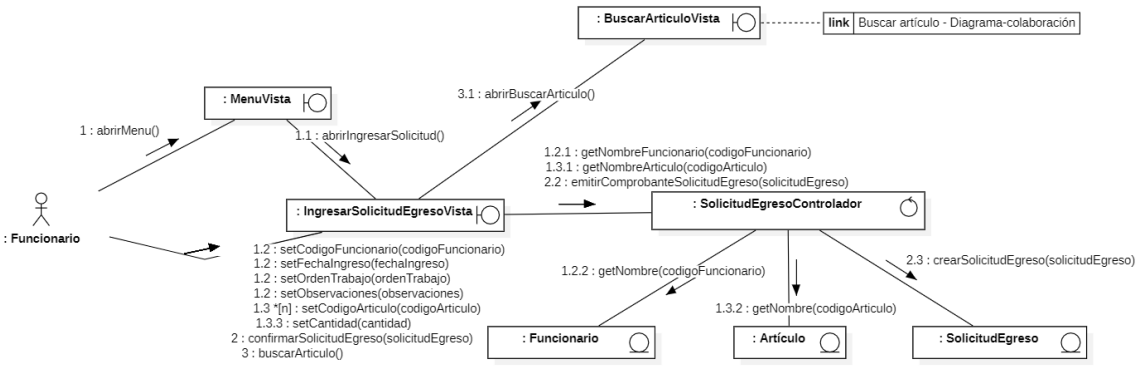


Figura 3.23: Diagrama de colaboración del caso de uso CU2: Ingresar solicitud

CU3: Aprobar solicitud

El diagrama de clases de la Figura 3.24 muestra las clases involucradas y sus interacciones. Estas clases colaboran para permitir al funcionario aprobar una solicitud de egreso, desde la visualización de las solicitudes pendientes hasta la confirmación de la aprobación. A continuación, se describen las clases identificadas:

- Funcionario: El actor que interactúa con el sistema para aprobar una solicitud de egreso.
- MenuVista: Clase de vista que proporciona el menú de opciones al funcionario.
- AprobarSolicitudEgresoVista: Clase que permite revisar y aprobar las solicitudes de egreso.
- SolicitudEgresoControlador: Clase controladora que maneja la lógica de negocio para aprobar la solicitud.
- SolicitudEgreso: Clase de modelo que contiene los detalles de las solicitudes de egreso.
- Funcionario: Clase de modelo que representa a los funcionarios involucrados en la aprobación de solicitudes.

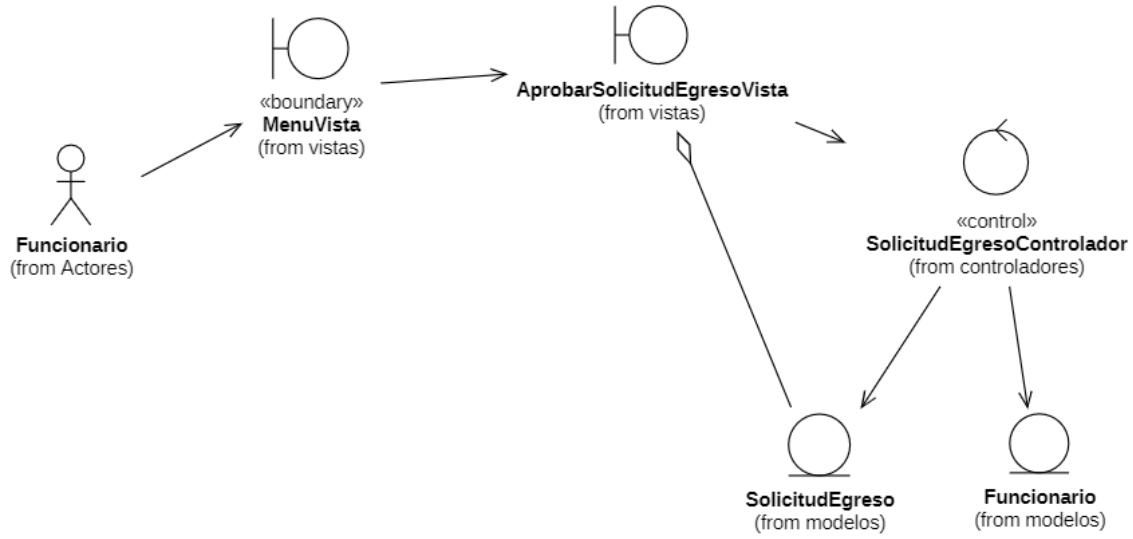


Figura 3.24: Diagrama de clases del caso de uso CU3: Aprobar solicitud

El diagrama de colaboración de la Figura 3.25 detalla las interacciones dinámicas entre los objetos de las clases para el caso de uso. Estas interacciones aseguran que todas las solicitudes de egreso sean revisadas y aprobadas de manera eficiente y precisa, manteniendo la integridad del sistema de gestión de bodega. Las siguientes son las interacciones clave para este caso de uso:

- Iniciar la Aprobación: El funcionario abre el menú y selecciona la opción para aprobar una solicitud de egreso.
- Revisión de la Solicitud: La vista AprobarSolicitudEgresoVista es utilizada para revisar los detalles de la solicitud de egreso, incluyendo lo siguiente: código del funcionario, código de la solicitud, tipo de solicitud, artículos solicitados y otros datos relevantes.
- Búsqueda de Solicitud de Egreso: El funcionario puede buscar solicitudes de egreso utilizando la vista BuscarSolicitudEgresoVista.
- Validación y Actualización: La clase SolicitudEgresoControlador valida la información y procesa la aprobación. Se actualizan los detalles de la solicitud de egreso, como el estado de aprobación y las fechas correspondientes.

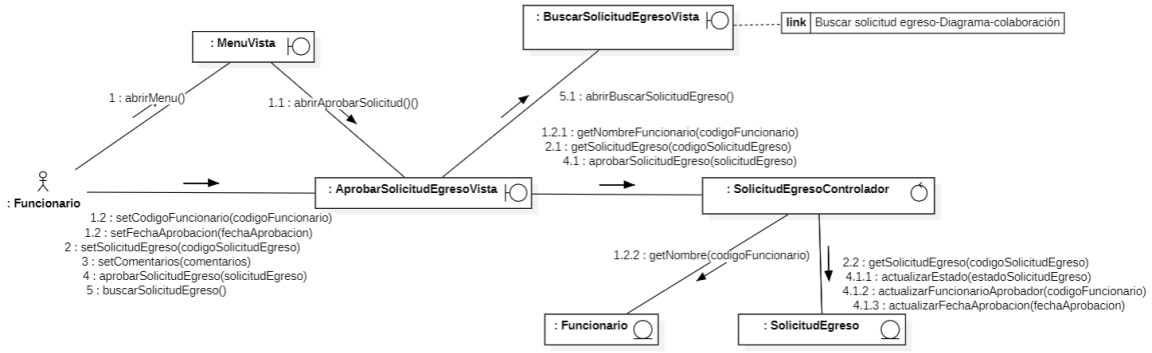


Figura 3.25: Diagrama de colaboración del caso de uso CU3: Aprobar solicitud

CU4: Tramitar operación de egreso

El diagrama de la Figura 3.26 muestra la colaboración de las distintas clases involucradas para permitir al funcionario de bodega tramitar una operación de egreso, desde la búsqueda de solicitudes hasta la confirmación del egreso. Las clases identificadas en este diagrama son:

- Funcionario de bodega: El actor que interactúa con el sistema para tramitar una operación de egreso.
- MenuVista: Clase de vista que proporciona el menú de opciones al funcionario.
- TramitarOperacionEgresoVista: Clase de vista que permite al funcionario tramitar una operación de egreso.
- BuscarSolicitudEgresoVista: Clase de vista que permite al funcionario buscar solicitudes de egreso.
- GestionOperacionControlador: Clase controladora que maneja la lógica de negocio para tramitar la operación de egreso.
- SolicitudEgreso: Clase de modelo que contiene los detalles de las solicitudes de egreso.
- Egreso: Clase de modelo que contiene los detalles de las solicitudes de egreso.
- Funcionario: Clase de modelo que representa a los funcionarios de la empresa.

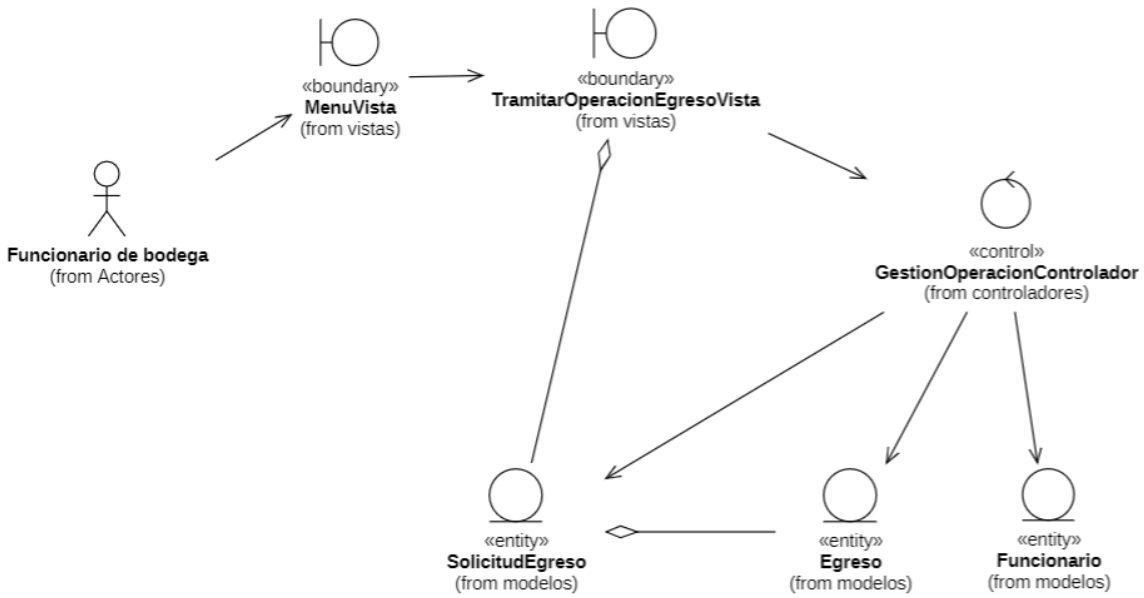


Figura 3.26: Diagrama de clases del caso de uso CU4: Tramitar operación de egreso

El diagrama de colaboración de la Figura 3.27 especifica las interacciones entre los objetos de las clases para el caso de uso *Tramitar operación de egreso*. Dichas interacciones aseguran que todas las operaciones de egreso sean gestionadas de manera eficiente y precisa, manteniendo la integridad del sistema de gestión de bodega. A continuación, se presenta una descripción textual de los pasos a seguir del caso de uso representado en la imagen.

- Iniciar la Operación: El funcionario abre el menú y selecciona la opción para tramitar una operación de egreso.
- Ingreso de Detalles del Egreso: Mediante la vista TramitarOperacionEgresoVista se puede ingresar los detalles de la operación de egreso.
- Búsqueda de Solicitud de Egreso: El funcionario puede buscar solicitudes de egreso mediante la vista BuscarSolicitudEgresoVista.
- Validación y Actualización: La clase GestionOperacionControlador se encargan de validar la información y procesar la operación de egreso, se actualizan los detalles de la solicitud de egreso y se registra la operación de egreso. Finalmente, se genera un comprobante de la operación de egreso y se confirma la operación.

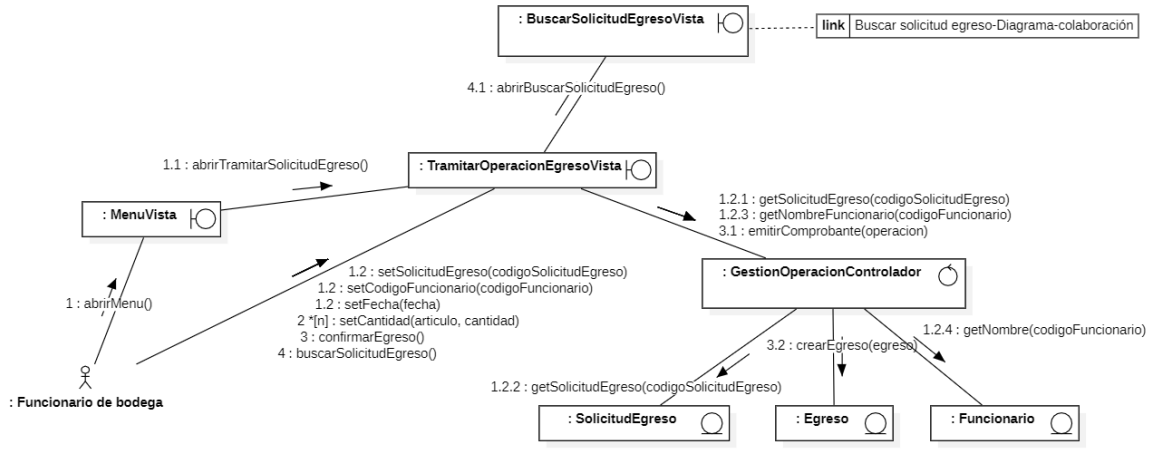


Figura 3.27: Diagrama de colaboración del caso de uso CU4: Tramitar operación de egreso

3.3.3. Análisis de Clases

Para realizar este análisis, se revisan los diagramas de clases y de colaboración obtenidos durante el análisis de casos de uso, y se identifican con qué clases se relacionan cada una y qué mensajes se intercambian entre ellas. En esta parte se determinan las dependencias y responsabilidades de las clases de análisis, conjuntamente con los atributos y relaciones.
Además, se puede identificar métricas importantes como el acoplamiento entre clases y actuar en consecuencia para optimizar la arquitectura del sistema. Como ejemplo de esta actividad se presentan los siguientes diagramas de clases.
En el primer diagrama de la Figura 3.28 se puede apreciar la clase vista TramitarOperacionIngresoVista y las clases con las que se relaciona.

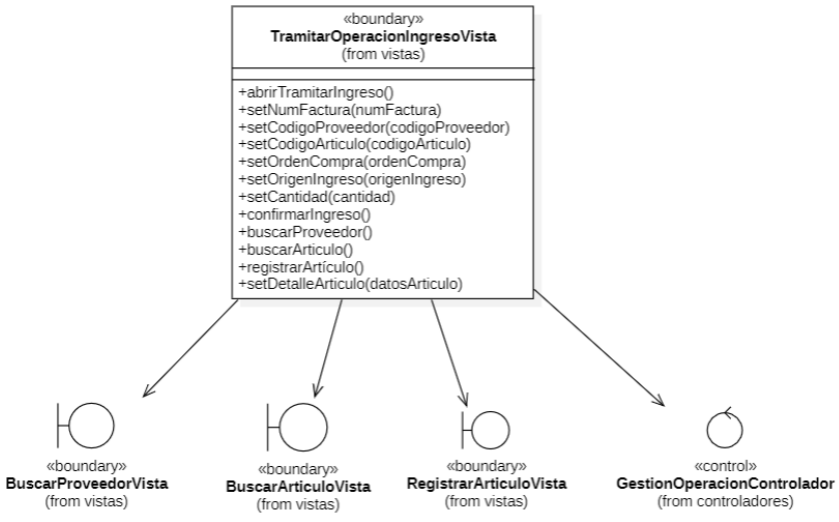


Figura 3.28: Diagrama de clases de la clase TramitarOperacionIngresoVista

Así mismo, en la Figura 3.29 se presenta la clase controlador GestionOperacionControlador con sus respectivas clases relacionadas.

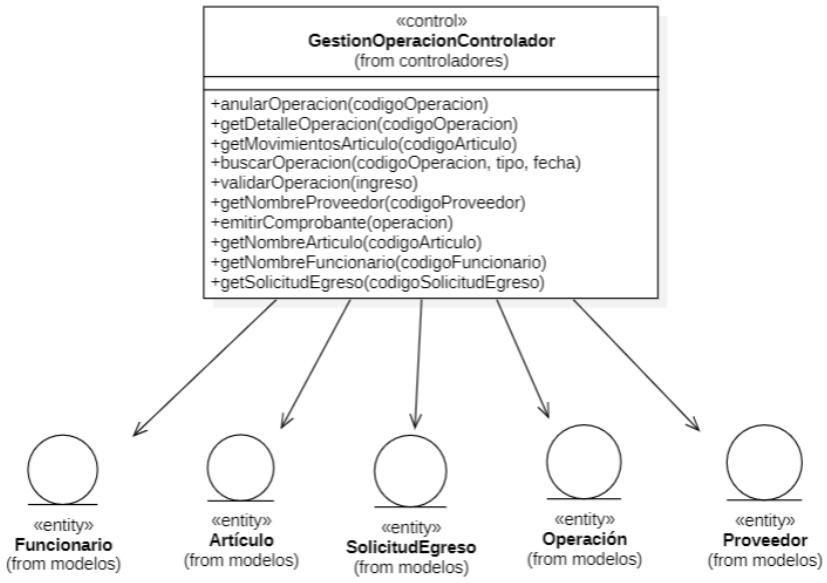


Figura 3.29: Diagrama de clases de la clase GestionOperacionControlador

3.3.4. Análisis de Paquetes

El análisis de paquetes se enfoca en el agrupamiento de clases en paquetes, para garantizar que dichos paquetes de análisis sean independientes entre sí, también pretende describir las dependencias de cierto modo que se pueda estimarse el efecto de cambios futuros. A continuación, se puede observar el diagrama de paquetes del sistema, el cual consta de cuatro paquetes: Gestión de usuarios, gestión de proveedores, gestión de solicitudes y gestión de bodega. Estos paquetes están relacionados mediante líneas punteadas, las cuales demuestran dependencia. Gestión de bodega depende de gestión de proveedor y gestión de usuario, gestión de solicitudes depende de gestión de bodega y gestión de usuario.

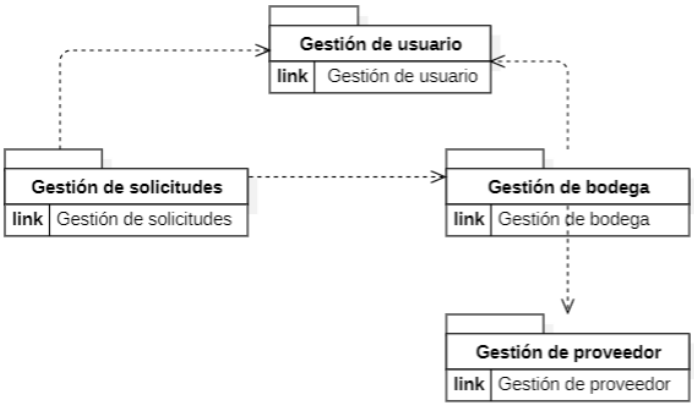


Figura 3.30: Diagrama de paquetes del sistema

Dentro de cada uno de los paquetes de la Figura 3.30 se presenta una organización divida en modelos, vistas y controladores como se puede apreciar en la Figura 3.31.

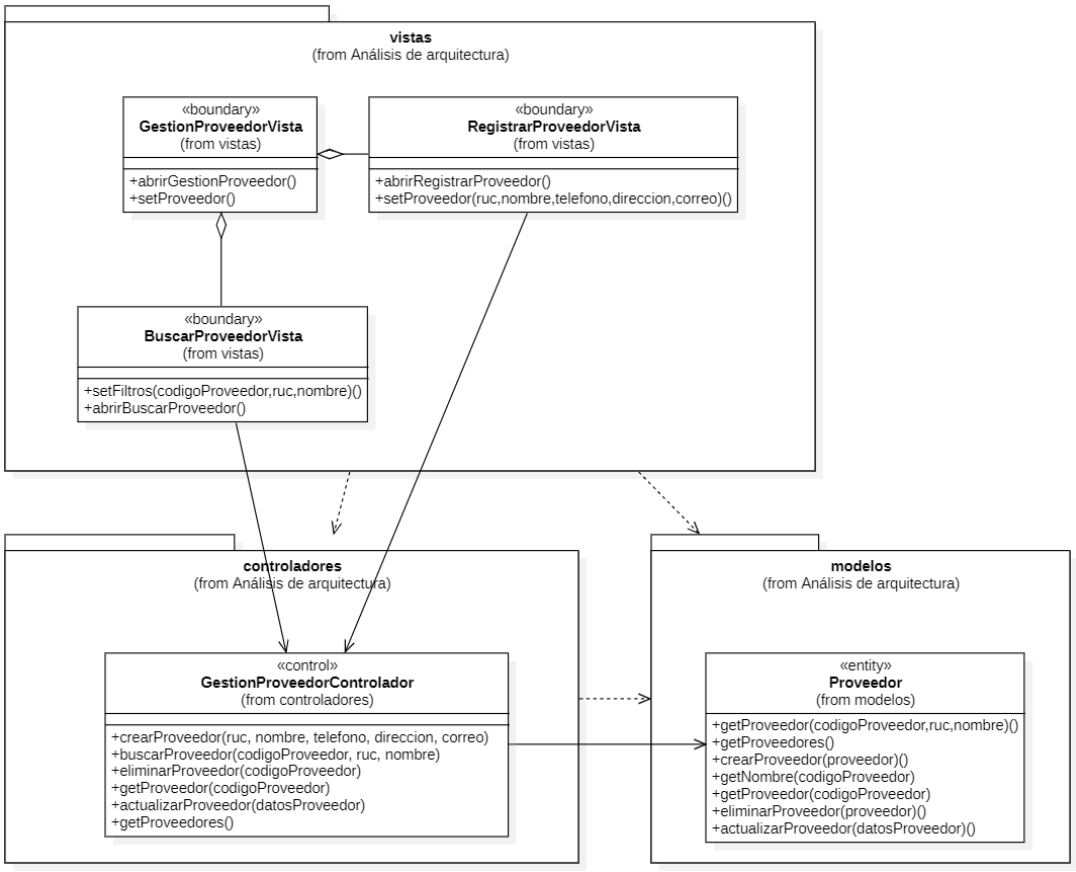


Figura 3.31: Diagrama de clases del paquete Gestión de proveedores

3.3.5. Diseño de la Arquitectura

La arquitectura para este proyecto es una arquitectura por capas, diseñada bajo el modelo cliente-servidor. Esta organización se compone de los siguientes componentes clave:

- Capa de presentación: Desarrollada en Angular, esta capa se ejecuta en el navegador del cliente y es responsable de la interfaz de usuario. Tiene contacto con la capa de negocio.
- Capa de negocio: Desarrollada en Spring Boot, esta capa se encarga de procesar la lógica de negocio, interactuar con la base de datos y devolver las respuestas adecuadas a la capa de presentación.
- Capa de datos: Gestiona la comunicación con la base de datos PosgreSQL. Esta base de datos se encuentra en un servidor independiente y se encarga de almacenar todos los datos necesarios para el funcionamiento del sistema.

La Figura 3.32 es una vista arquitectónica del sistema. Para realizarla se empleó una vista lógica, una de las 4 vistas propuestas por Kruchten (Jayawardene, 2021), usando un diagrama de paquetes para su representación. En este diagrama se puede apreciar el flujo de comunicación del sistema, descrito detalladamente a continuación:

- Interacción del usuario: El usuario interactúa con la aplicación a través de la interfaz proporcionada por Angular.
- Solicitudes del cliente: Angular envía solicitudes HTTP al backend de Spring Boot para realizar operaciones como obtener, crear, actualizar o eliminar datos.
- Procesamiento del Servidor: Spring Boot procesa estas solicitudes, ejecutando la lógica de negocio correspondiente y, si es necesario, interactúa con la base de datos PostgreSQL para obtener o modificar datos.
- Respuestas del Servidor: Spring Boot envía las respuestas de vuelta a Angular en formato JSON, para que la interfaz de usuario se actualice en consecuencia.

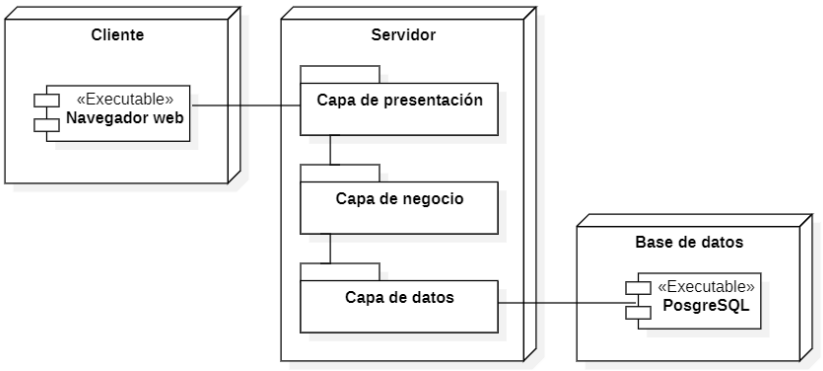


Figura 3.32: Diagrama de la arquitectura del sistema

Capa de presentación

Esta capa se comunica con la capa de negocio para gestionar los eventos del usuario, incluyendo la captura de entradas y la validación de datos. Cumple con las funciones de renderización de la interfaz de usuario, envío de solicitudes HTTP a la capa de negocio a través de la API REST, recibe y procesa las respuestas para actualizar la interfaz de usuario dinámicamente y gestiona el enrutamiento del lado del cliente para la navegación dentro de la aplicación.
En esta capa se organizan los paquetes de acuerdo con la organización de las pantallas como se puede apreciar en la Figura 3.33. El paquete Core contendrá los servicios centrales que se utilizarán en toda la aplicación, como la autenticación y la gestión de llamadas HTTP. El paquete Shared incluye componentes, modelos y funciones que se utilizarán en varios lugares de la aplicación. Esto con la finalidad de evitar duplicaciones y facilitar el mantenimiento. El paquete Features, se divide en submódulos que reflejan los subsistemas del backend (gestión de bodega, gestión de proveedores y gestión de usuarios). Cada uno de estos módulos contiene sus propios componentes, servicios y modelos, lo que facilita la escalabilidad y la separación de responsabilidades.

- Components: Cada carpeta bajo components contiene componentes específicos de ese módulo.
- Services: Contienen servicios que se utilizan para manejar la lógica de negocio y las solicitudes a las APIs.
- Models: Contienen interfaces para manejar los datos dentro de cada módulo.

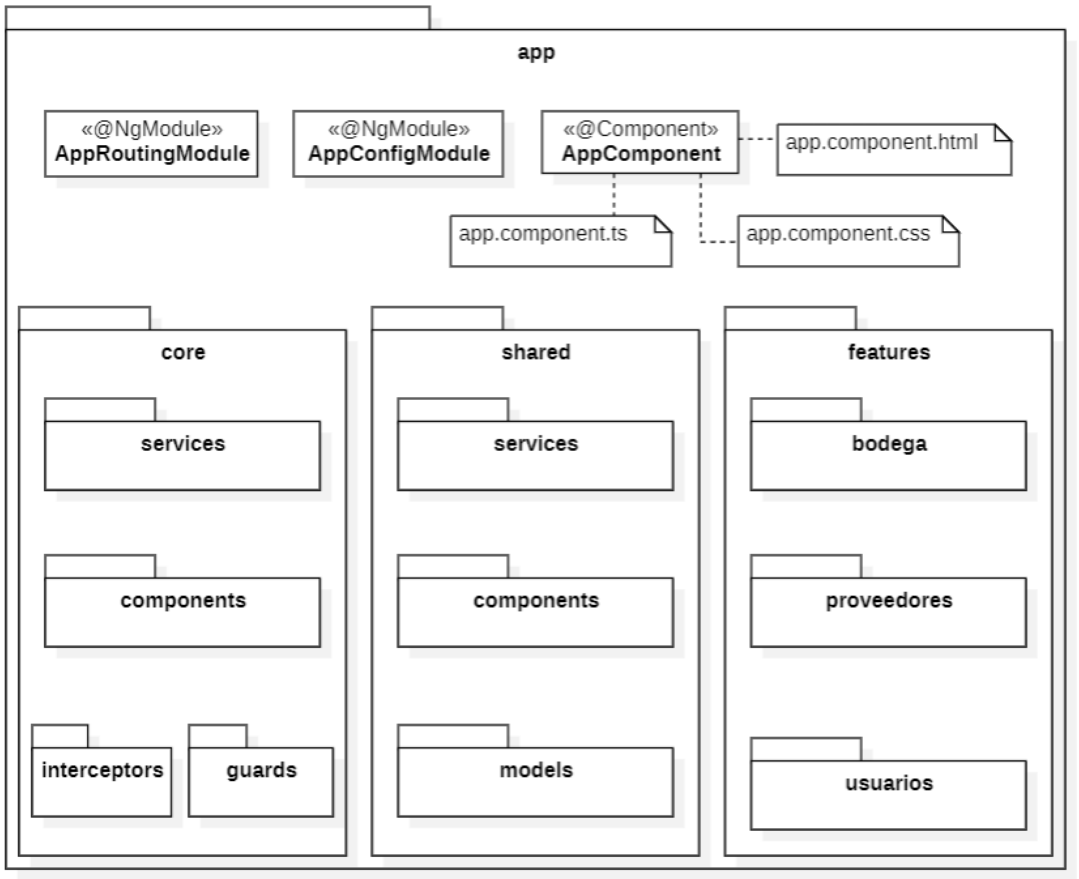


Figura 3.33: Diagrama de paquetes de la capa de presentación

Capa de negocio

Esta capa contiene la lógica de negocio que define cómo se deben procesar los datos, se encarga también de la validación y procesamiento de los datos recibidos desde la capa de presentación. Además, es responsable de la gestión de autenticación y autorización de usuarios y está en constante interacción con la capa de datos.

En esta capa se implementa una API REST, la comunicación con la capa de presentación se realiza mediante el protocolo HTTP y la información se envía en formato JSON. En la Figura 3.34 se puede ver el diagrama de paquetes de la capa de negocio del sistema, en el cual se ilustra su estructura.

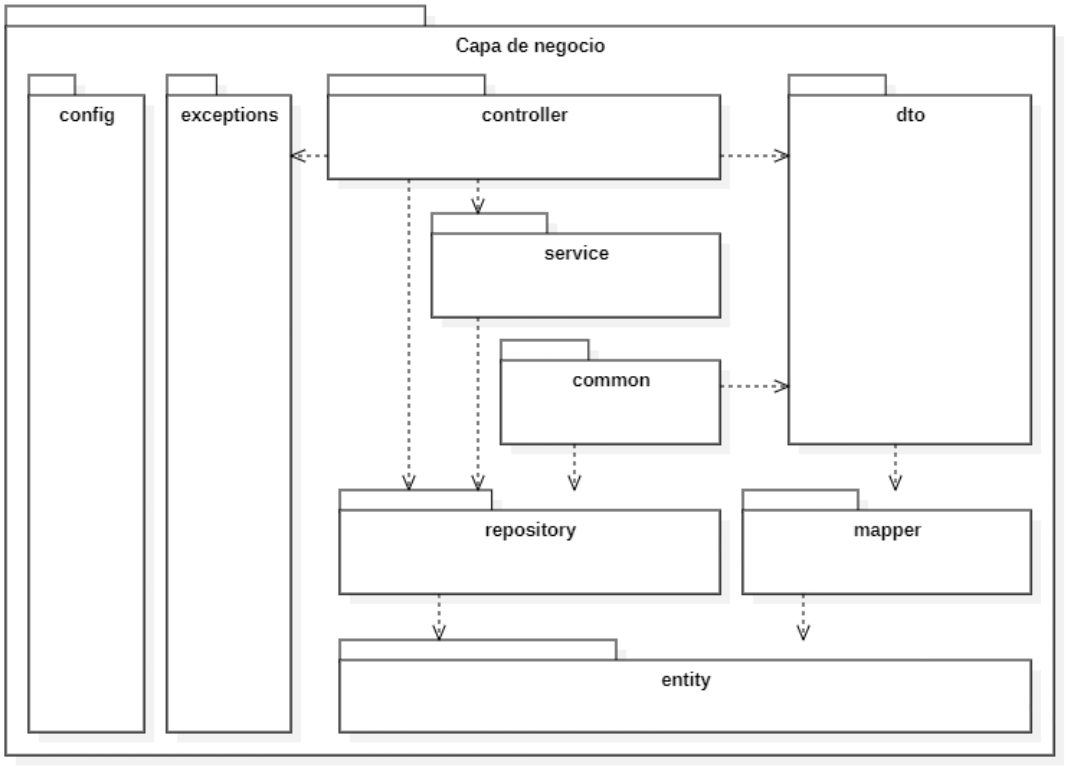


Figura 3.34: Diagrama de paquetes de la capa de negocio

Se consideran nueve paquetes especificados a continuación:

1. Paquete controller: Clases que manejan las solicitudes HTTP y las respuestas actuando como una interfaz entre la capa de presentación y la lógica de negocio.

2. Paquete service: Contiene la lógica de negocio del sistema.

3. Paquete repository: Interfaces para la capa de persistencia, extendiendo repositorios de Spring Data JPA, para realizar operaciones CRUD (Create, Read, Update, Delete) sobre la base de datos de manera abstracta, siguiendo el patrón Repository.

4. Paquete entity: Clases que representan entidades en la base de datos. Estas clases son mapeadas a tablas de la base de datos mediante anotaciones JPA (Java Persistence API). Este paquete encapsula el estado y el comportamiento relacionados con las entidades del dominio.

5. Paquete dto: Clases utilizadas para transferir datos entre las diferentes capas del sistema para este paquete se plantea el uso del patrón Data Transfer Object (DTO) para minimizar la cantidad de llamadas entre la capa de presentación y la capa de negocio, y facilitar la validación y transformación de los datos.

6. Paquete config: Clases de configuración específicas para el sistema, se definen configuraciones relacionadas con el contexto de Spring, la seguridad, la base de datos, entre otros aspectos del sistema.

7. Paquete common: Clases comunes que exponen interfaces y permiten la comunicación entre subsistemas. Este paquete incluye utilidades, constantes, y componentes compartidos que son utilizados en diferentes partes del sistema, promoviendo la reutilización de código y la cohesión.

8. Paquete mapper: Dentro de este paquete se plantea la creación de clases que mapean entre DTOs y entidades para asegurar que los datos se transfieran correctamente y facilitar la separación de la lógica de negocio.

9. Paquete exceptions: Clases para manejar excepciones personalizadas, para garantizar la gestión de errores de manera uniforme, proporcionar mensajes de error claros y mantener la consistencia en el manejo de excepciones.

Capa de datos

La capa de datos se encarga de gestionar la persistencia de los datos y asegurar la integridad de los mismos. Esta capa proporciona una abstracción sobre la base de datos para la comunicación con la capa de negocio.

Esta capa está compuesta por entidades que representan las tablas de la base de datos. En el diagrama de clases del modelo de la Figura 3.35 se pueden observar las principales entidades del sistema incluyendo *Funcionario, Artículo, SolicitudEgreso, Operación y Proveedor*, entre otras. Estas entidades están interrelacionadas para reflejar la estructura lógica del modelo de dominio permitiendo una gestión coherente y ordenada de los datos.

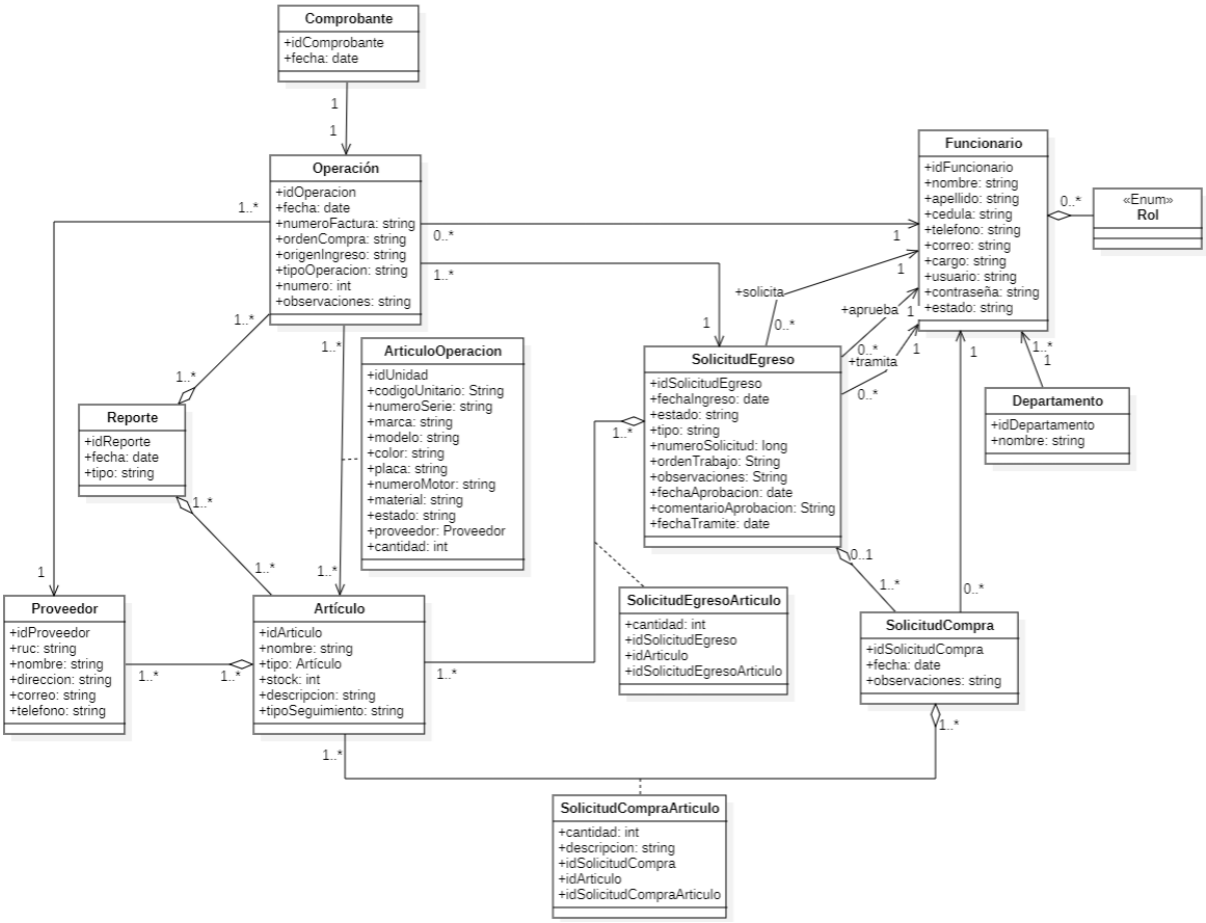


Figura 3.35: Diagrama de clases de la capa de datos

3.3.6. Diseño de Casos de Uso

El diseño de casos de uso se enfoca en transformar los modelos de análisis en una implementación detallada para la construcción del sistema. Esta actividad pretende asegurar que los requisitos funcionales y no funcionales identificados durante el análisis sean adecuadamente mapeados y representados en el diseño del sistema.

El diagrama presentado en la Figura 3.36 muestra el mapeo desde el análisis al diseño, destacando la transición de las clases de análisis a las componentes de diseño. Este mapeo sigue un enfoque sistemático, donde cada elemento del modelo de análisis tiene su correspondiente en el diseño, garantizando coherencia y trazabilidad a lo largo del proceso de desarrollo.

- Paquete Análisis: Incluye las clases de vista, modelo y controlador identificadas durante el análisis. Estas clases representan una abstracción de los componentes del sistema y sus interacciones.

- Paquete Backend: Incluye los controladores, servicios, repositorios, entidades y objetos de transferencia de datos. Este paquete mapea las clases de análisis a componentes específicos que manejan la lógica de negocio, la persistencia de datos y la comunicación entre diferentes capas del sistema.
- Paquete Frontend: Incluye los componentes de interfaz de usuario, plantillas, estilos y servicios del frontend. Estos elementos están diseñados para interactuar con el backend.

El diagrama de mapeo ilustra cómo los elementos del análisis, tales como vistas y controladores, se traducen en componentes de frontend y backend, asegurando que todas las funcionalidades previstas sean implementadas correctamente. Las relaciones y dependencias entre estos componentes están claramente definidas para mantener la cohesión y reducir el acoplamiento, siguiendo los principios del diseño orientado a objetos (Jacobson et al., 1999).

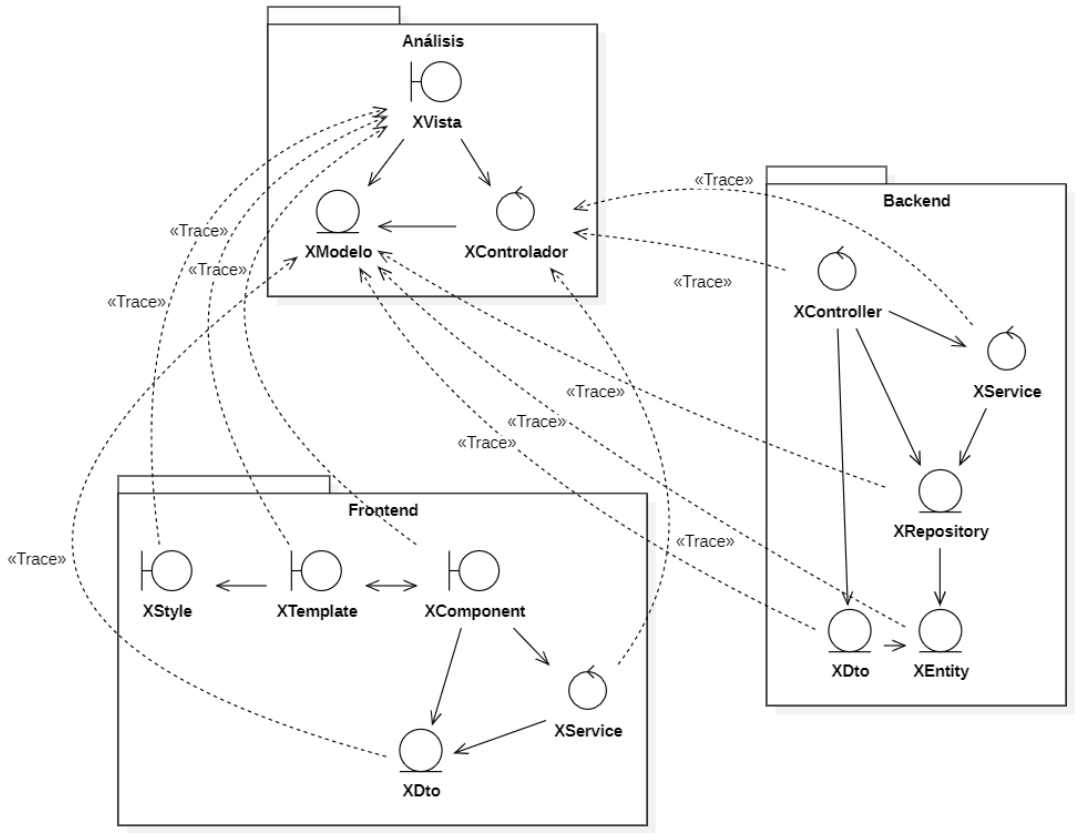


Figura 3.36: Diagrama del diseño de casos de uso

3.3.7. Diseño de Paquetes

En el diseño de paquetes, se aborda la estructuración del sistema en módulos cohesivos y bien definidos que facilitan su desarrollo, mantenimiento y escalabilidad. A continuación, se

presentan dos diagramas que ilustran este proceso de diseño.

El diagrama de la Figura 3.37 muestra el mapeo desde los paquetes de análisis hacia el diseño, se han consolidado los paquetes de Gestión de Solicitudes dentro del paquete de Gestión de Bodega para simplificar el manejo de información y para desacoplar las clases involucradas. Esta decisión se basa en la necesidad de mantener una estructura manejable y coherente que facilite la implementación y el mantenimiento del sistema.

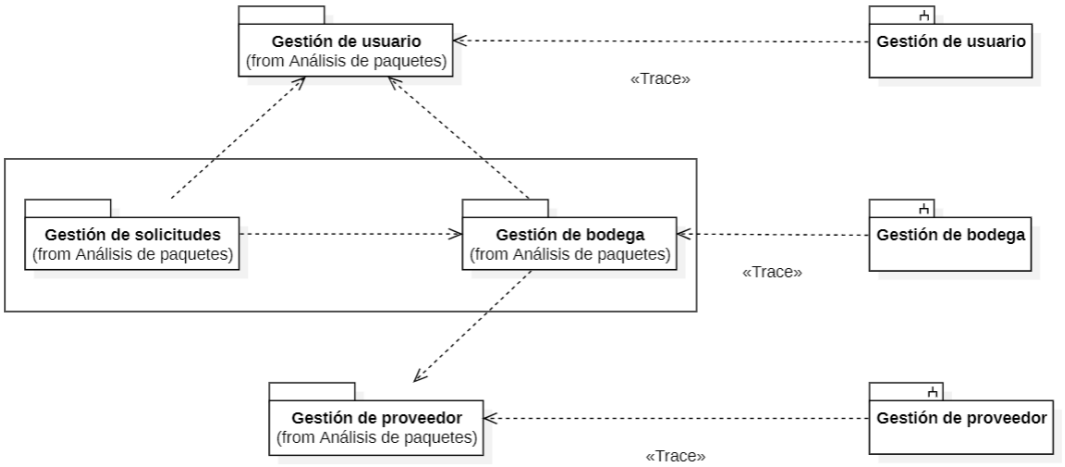


Figura 3.37: Mapeo desde los paquetes de análisis

- Gestión de usuario: Encargado de manejar todas las operaciones relacionadas con los usuarios del sistema.
- Gestión de bodega: Encargado de gestionar las solicitudes y las operaciones relacionadas con el inventario.
- Gestión de proveedor: Encargado de manejar todas las operaciones relacionadas con los proveedores.

El diagrama de componentes de la Figura 3.38 presenta una vista más detallada de los subsistemas que se van a implementar y se presentan las interfaces que sirven para comunicar los mismos. Este diagrama resalta la estructura modular del sistema y la interacción entre los distintos componentes.

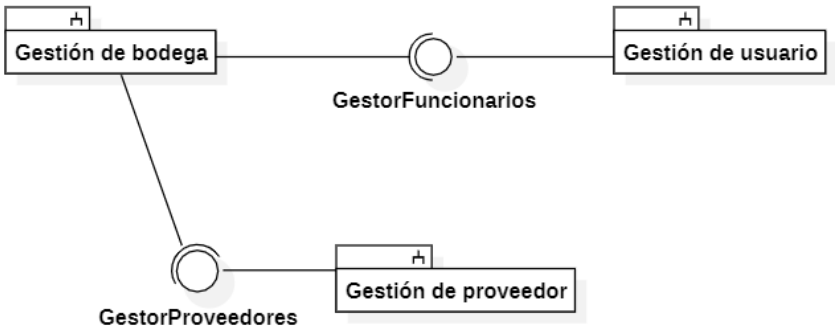


Figura 3.38: Diagrama de componentes de los subsistemas que conforman el sistema

Este diseño permite la separación de responsabilidades para mejorar la mantenibilidad y escalabilidad del sistema, cada subsistema está diseñado para ser independiente y relacionarse mediante interfaces de comunicación con los demás subsistemas. A continuación, se presenta una breve descripción de cada subsistema de la Figura 3.38.

- Gestión de usuario: Utiliza la interfaz GestorFuncionarios para manejar las operaciones relacionadas con los usuarios.
- Gestión de bodega: Interactúa con las interfaces GestorProveedores y GestorFuncionarios para realizar operaciones relacionadas con la bodega.
- Gestión de proveedor: Se comunica con GestorProveedores para gestionar las relaciones con los proveedores.

3.4. Implementación

En esta disciplina de Implementación, se traduce el diseño detallado en un sistema funcional para ello se toma el rol de desarrollador dentro del proyecto. En esta sección, se describen los pasos seguidos para la construcción del Sistema Integral de Gestión de Bodega para la Empresa Eléctrica Azogues C.A. Se detallan las herramientas y tecnologías utilizadas, la configuración del entorno de desarrollo, la estructura arquitectónica del sistema, y se presentan ejemplos concretos de la implementación del frontend, backend y la base de datos.

3.4.1. Herramientas de Desarrollo

Para el desarrollo del sistema se utilizaron diversas herramientas, las principales se muestran a continuación:

- Visual Studio Code: IDE utilizado para la edición de código fuente del frontend.
- IntelliJ IDEA: IDE utilizado para la edición de código fuente del backend.
- GitHub: Plataforma para el control de versiones.
- Angular CLI: Herramienta de línea de comandos para desarrollar aplicaciones en Angular.
- Spring Boot: Framework para el desarrollo de aplicaciones Java basadas en Spring.
- PostgreSQL: Sistema gestor de bases de datos.
- Docker: Plataforma para desarrollar y ejecutar aplicaciones dentro de contenedores.

La elección de estas herramientas está basada principalmente por la existencia de una amplia comunidad de soporte. Particularmente, la combinación de Spring Boot y Angular es utilizada y recomendada por su capacidad para desarrollar aplicaciones web modernas (dubey, 2024), por lo que se optó por ambas tecnologías. Además, esta combinación permite mantener separados el frontend del backend, lo que contribuye al desarrollo modular y el mantenimiento del sistema.

3.4.2. Implementación de la Capa de Presentación

La capa de presentación del Sistema Integral de Gestión de Bodega se desarrolló utilizando el framework Angular. A continuación, se describe la organización del proyecto, las principales funcionalidades implementadas y adicionalmente se incluirán capturas de pantalla para ilustrar la interfaz de usuario.

Organización del proyecto

La organización del proyecto sigue la estructura planificada en la disciplina de análisis y diseño. La Figura 3.39 presenta una captura de pantalla en donde se puede apreciar la estructura de carpetas y archivos principales del proyecto.

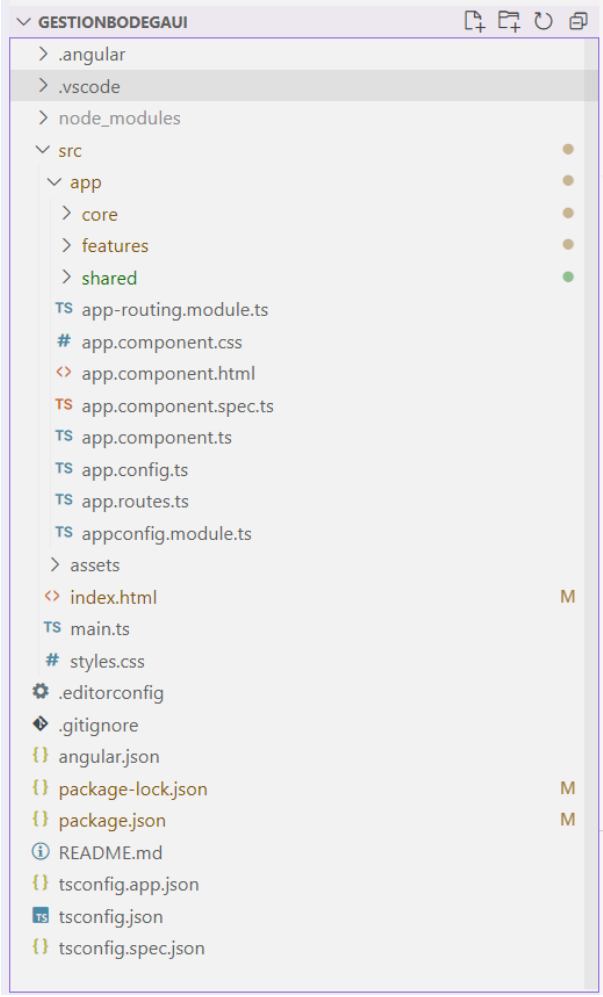


Figura 3.39: Organización del proyecto - capa de presentación

Como se puede observar, la estructura de carpetas está organizada considerando las carpetas core, shared y features, como se había planificado en la disciplina de análisis y diseño.

En la Figura 3.40 se puede apreciar específicamente la organización de carpetas y archivos dentro del módulo de gestión de bodega. Esta estructura refleja la planificación detallada realizada durante la disciplina de análisis y diseño del sistema, dentro de este módulo se pueden observar varias subcarpetas y archivos que organizan los componentes, modelos y servicios del sistema. Esta organización facilita la mantenibilidad y escalabilidad del proyecto, asegurando una clara separación de responsabilidades.

La carpeta components contiene los componentes del módulo de gestión de bodega, organizados por funcionalidades específicas. Por ejemplo, la subcarpeta solicitudes-egreso agrupa todos los componentes relacionados con la gestión de solicitudes de egreso, específicamente se puede ver el componente aprobar-solicitud y los archivos que lo conforman incluyendo: archivo de estilos CSS, archivo HTML que define la vista, archivo de pruebas unitarias y archivo

TypeScript que contiene la lógica del componente. La carpeta models contiene los modelos de datos utilizados en el módulo y la carpeta services que incluye los servicios que manejan la lógica de negocio y la comunicación con el backend.

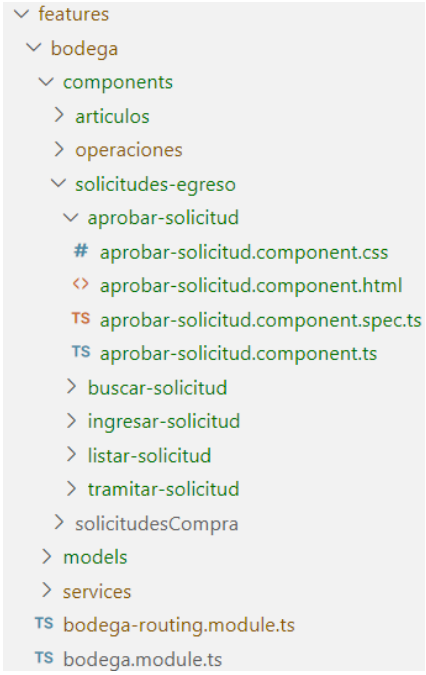


Figura 3.40: Estructura de carpetas y archivos del Módulo de Gestión de Bodega

Funcionalidades implementadas

La capa de presentación incluye varias funcionalidades clave que permiten a los usuarios interactuar con el sistema de manera eficiente. A continuación, se describen las principales funcionalidades y componentes implementados:

- LoginComponent: Este componente gestiona la autenticación de usuarios, permitiendo el acceso seguro al sistema.
- GestionArticulosComponent: Este componente está dedicado a la gestión de artículos, es decir, permite registrar, actualizar, visualizar y eliminar artículos de inventario.
- OperacionIngresoComponent: Este componente se encarga de realizar una operación de ingreso a bodega.
- OperacionEgresoComponent: Este componente permite realizar una operación de egreso a bodega.

Interfaz de usuario

Para ilustrar la interfaz de usuario desarrollada se incluyen capturas de pantalla de los principales componentes.

La interfaz de usuario para iniciar sesión se puede ver en la Figura 3.41, esta pantalla permite a los usuarios autenticarse en el sistema ingresando su usuario y contraseña.

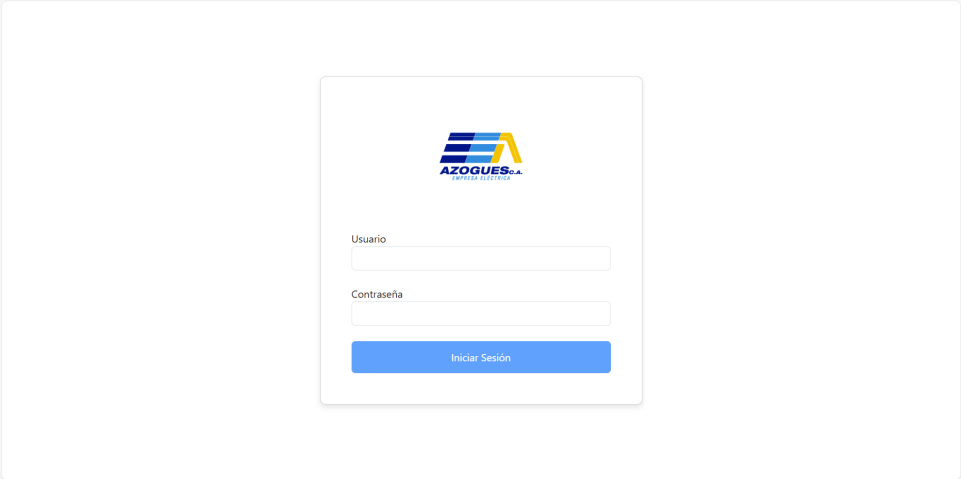


Figura 3.41: Interfaz de usuario para Iniciar sesión

La Figura 3.42 muestra la interfaz para registrar nuevos artículos en el sistema, ingresando detalles como el nombre del artículo, tipo, y descripción.

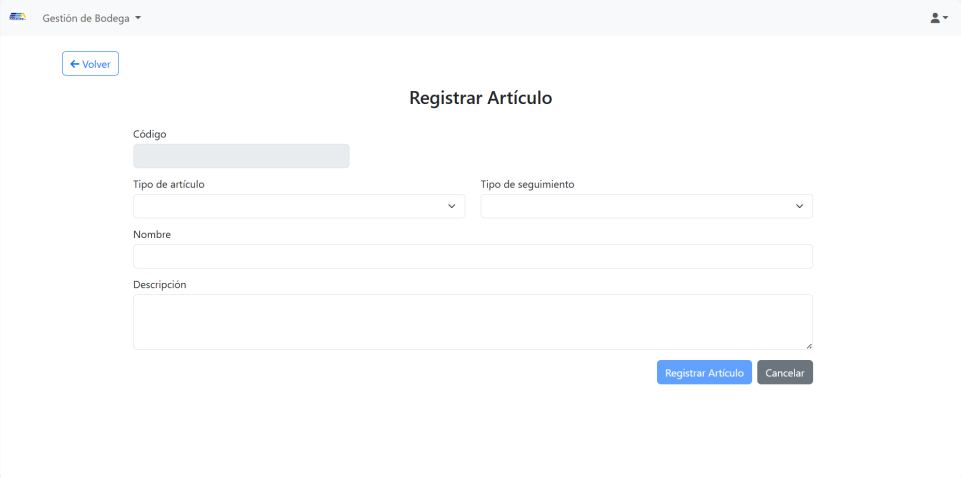


Figura 3.42: Interfaz de usuario para Registrar un artículo

En la interfaz de usuario de la Figura 3.43, permite a los usuarios registrar proveedores, incluyendo información relevante como nombre del proveedor, RUC, dirección y contacto.

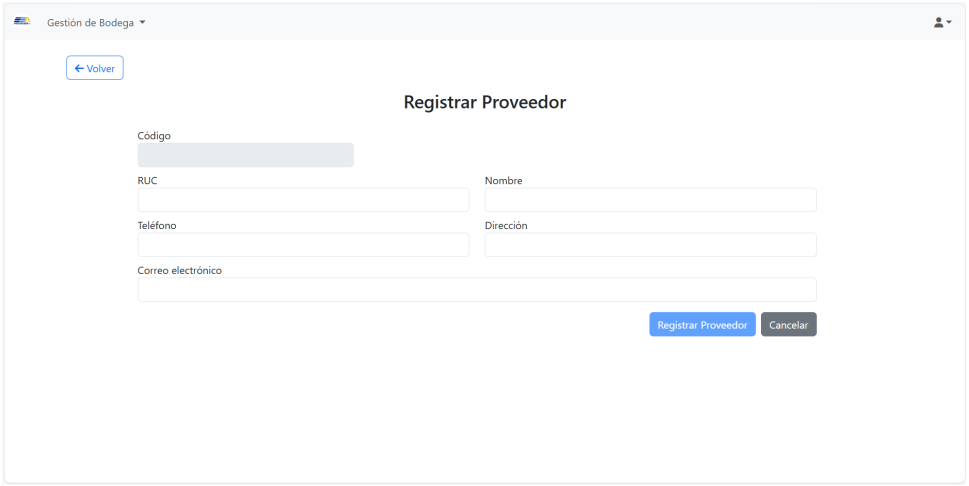


Figura 3.43: Interfaz de usuario para Registrar un proveedor

La pantalla de la Figura 3.44 permite registrar el ingreso de solicitudes de egreso, incluyendo información como orden de trabajo, detalles de los artículos requeridos, entre otros.

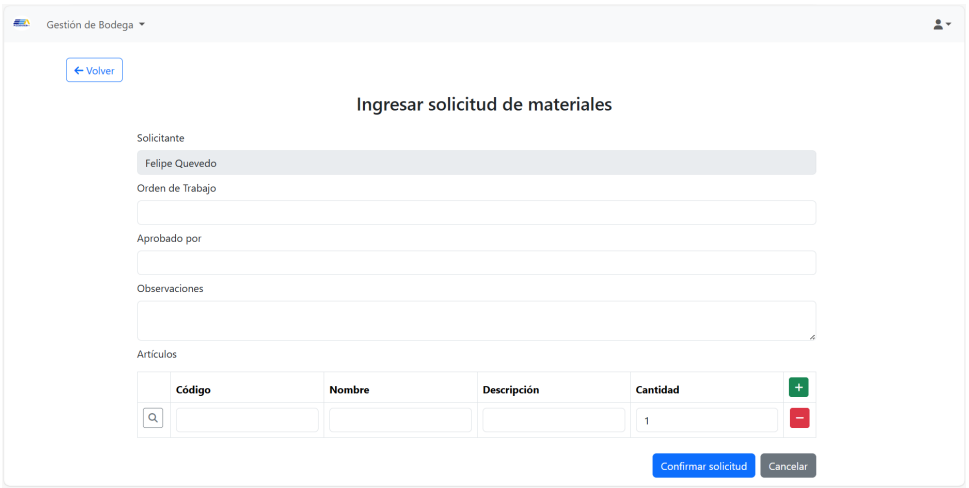


Figura 3.44: Interfaz de usuario para Ingresar una solicitud

La interfaz de la Figura 3.45 facilita el proceso de ingreso de artículos a la bodega, gestionando la entrada de nuevos bienes, materiales y suministros de oficina.

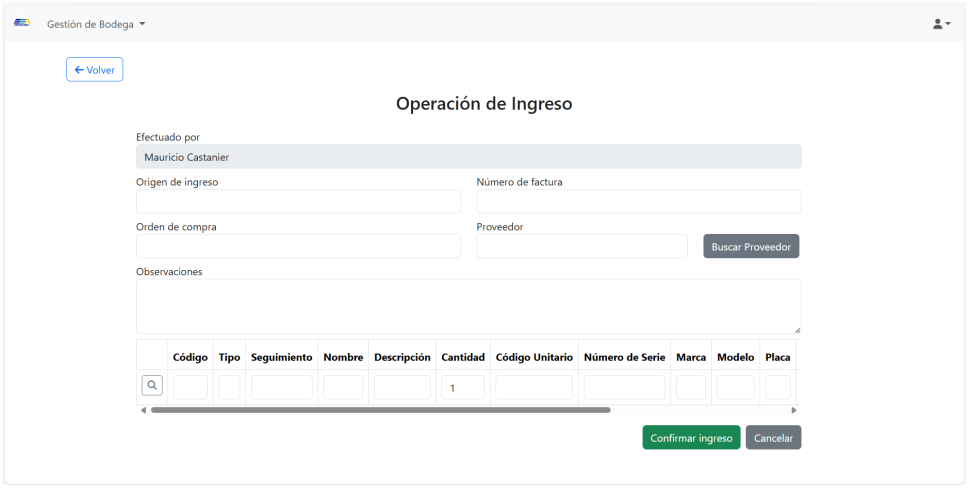


Figura 3.45: Interfaz de usuario para Tramitar una operación de ingreso

Por último, la interfaz de la Figura 3.46 permite gestionar la salida de artículos de la bodega, asegurando el registro y control de egresos.

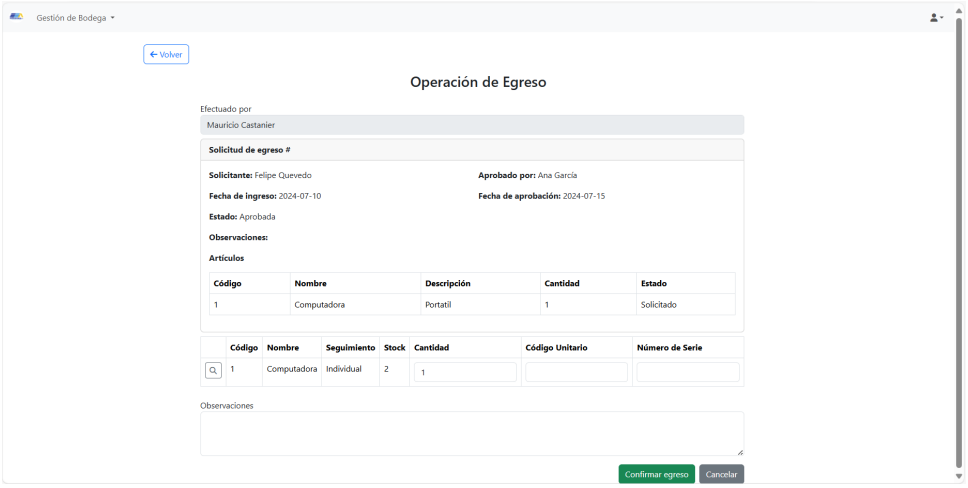


Figura 3.46: Interfaz de usuario para Tramitar una operación de egreso

3.4.3. Implementación de la Capa de Negocio

La capa de negocio está encargada de gestionar la lógica de negocio y las reglas que rigen las operaciones del sistema. En esta sección, se describe la implementación de la capa de negocio utilizando Spring Boot.

Se destacan las tecnologías y patrones de diseño utilizados, incluyendo Spring Security para asegurar que solo los usuarios autorizados puedan acceder a ciertas partes del sistema, conjuntamente se usa JSON Web Token (JWT) para crear tokens de acceso seguros y de esta forma validar las identidades de los usuarios y permitir la autenticación segura entre el cliente

y el servidor. Con respecto a los patrones de diseño utilizados, se puede mencionar al patrón Repository y Data Transfer Object (DTO). El patrón Repository es usado para la interacción con la base de datos a través de interfaces de repositorio y el patrón Data Transfer Object para transferir datos entre la capa de presentación y la capa de negocio, simplificando la estructura de datos y mejorando el rendimiento.

Organización del proyecto

La organización de la capa de negocio en el proyecto sigue la estructura establecida en la disciplina de análisis y diseño. En la Figura 3.47 se presenta la estructura de carpetas utilizada para la implementación de la capa de negocio.

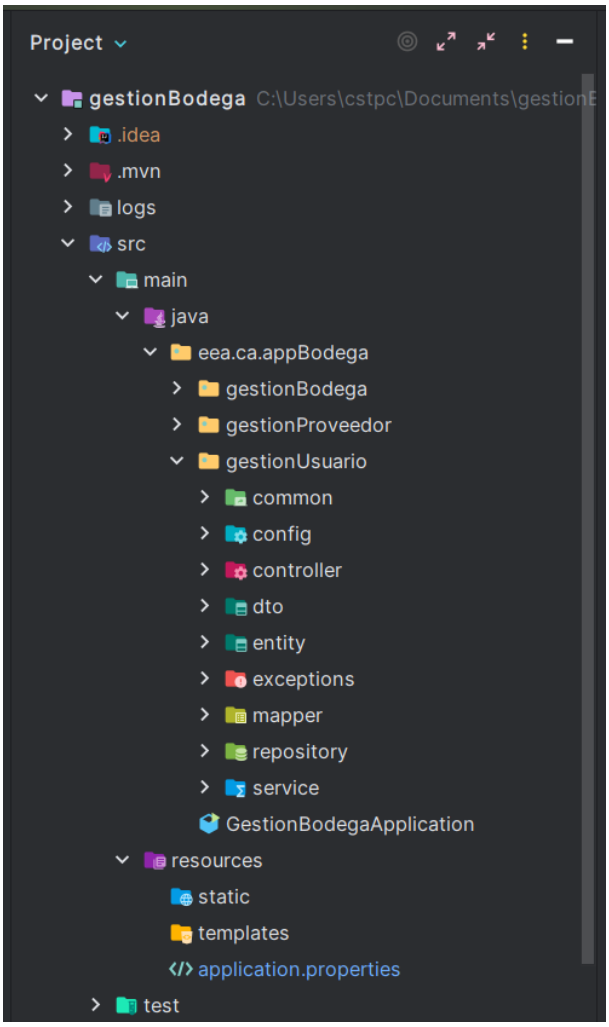


Figura 3.47: Organización del proyecto - capa de negocio

La Figura 3.48 muestra la estructura de las carpetas controller y entity, donde se puede observar los nombres de los archivos los cuales coinciden con los nombres de las clases diseñadas. Como se puede, cada clase sigue la nomenclatura definida en la disciplina de análisis y diseño,

asegurando de esta forma la consistencia con el diseño planificado.

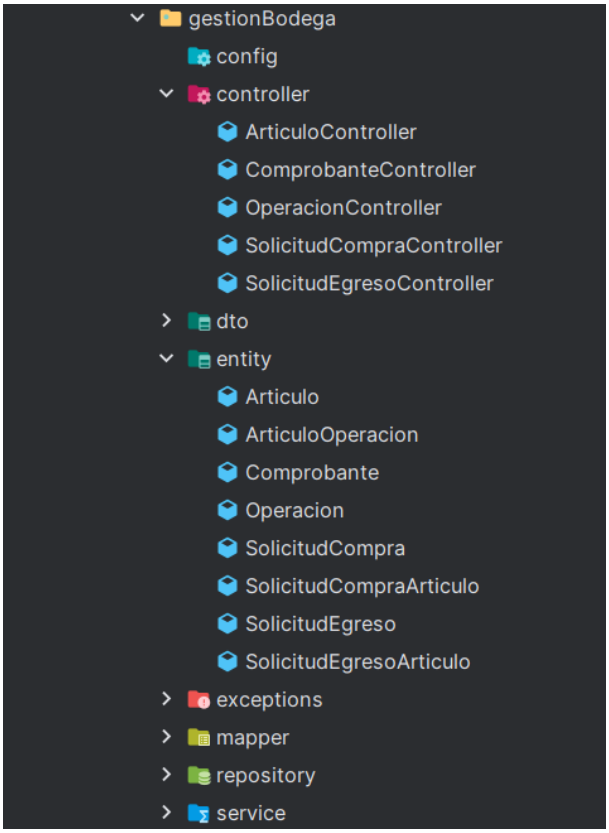


Figura 3.48: Estructura de carpetas y archivos del subsistema de Gestión de Bodega

El diseño del sistema, definido a través de diferentes artefactos de la disciplina de análisis y diseño, ha sido traducido en código de manera estructurada. Cada entidad y controlador corresponde a una clase que implementa la lógica y las relaciones definidas durante la fase de diseño.

Por ejemplo, en el diseño se definió una entidad SolicitudEgreso y un controlador SolicitudEgresoController. En las Figuras 3.49 y 3.50, se presenta una porción de código de ambas clases, mostrando cómo se han traducido las relaciones de diseño en sentencias de código.

En la clase SolicitudEgreso de la Figura 3.49 se pueden apreciar los import necesarios para la definición de la entidad, seguidamente entre las anotaciones se tiene la anotación @Entity, la cual indica que la clase es una entidad JPA que se mapeará a una tabla en la base de datos, las demás anotaciones de la librería Lombok facilitan la escritura de la clase, ya que eliminan el código repetitivo, como los getters, setters, y constructores.

```java
package eea.ca.appBodega.gestionBodega.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;

37 usages  Betty Flores *
@Entity
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class SolicitudEgreso {
    @Id
    private Long idSolicitudEgreso;
    private LocalDate fechaIngreso;
    private String estado;
    private String tipo;
    private String numeroSolicitud;
```

Figura 3.49: Clase SolicitudEgreso

En el SolicitudEgresoController de la Figura 3.50, se observa cómo los import (import de Spring y de las clases de la entidad y servicio) permiten que el controlador maneje las solicitudes HTTP y utilice el servicio SolicitudEgresoService para realizar operaciones sobre la entidad SolicitudEgreso. Las anotaciones @RestController y @RequestMapping definen la clase como un controlador REST y mapean las solicitudes HTTP a los métodos correspondientes.

```java
package eea.ca.appBodega.gestionBodega.controller;

import eea.ca.appBodega.gestionBodega.dto.SolicitudEgresoDto;
import eea.ca.appBodega.gestionBodega.service.SolicitudEgresoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

no usages  bettyfr +1 *
@RestController
@RequestMapping("/api/v1/solicitudes-egreso")
public class SolicitudEgresoController {

    12 usages
    @Autowired
    private SolicitudEgresoService solicitudEgresoService;

    no usages  new *
    @PostMapping
    public ResponseEntity<SolicitudEgresoDto> crearSolicitudEgreso(@RequestBody SolicitudEgresoDto solici
        SolicitudEgresoDto nuevaSolicitud = solicitudEgresoService.crearSolicitudEgreso(solicitudEgresoDt
        return new ResponseEntity<>(nuevaSolicitud, HttpStatus.CREATED);
    }
```

Figura 3.50: Clase SolicitudEgresoController

Cabe mencionar también la utilización de las clases de Servicio y Dto, las cuales facilitan la lógica de negocio y la transferencia de datos entre las diferentes capas de la aplicación. El servicio SolicitudEgresoService define las operaciones que pueden realizarse sobre las solicitudes de egreso, incluyendo métodos para crear, aprobar y tramitar solicitudes. Esto asegura que la lógica de negocio se maneje de manera centralizada y estructurada.
Este diseño modular y la consistencia en la implementación garantizan que el sistema sea escalable y mantenible, cumpliendo con los objetivos de la disciplina de análisis y diseño, y la disciplina de implementación.

3.4.4. Implementación de la Capa de Datos

La capa de datos se encarga de acceder a los datos del sistema. A continuación, se describe cómo la implementación de la capa de datos utilizando PostgreSQL, se abordarán los aspectos de la configuración y gestión de la base de datos, así como la integración con la capa de negocio mediante el uso de repositorios.

Diseño de la base de datos

La base de datos se diseñó para soportar todas las operaciones de la gestión de bodega, incluyendo la recepción, almacenamiento, y despacho de bienes, materiales y suministros de oficina. Se crearon diversas tablas que reflejan las entidades clave del dominio, como funcionarios, artículos de inventario y operaciones de bodega.
A partir de los diagramas desarrollados durante la disciplina de análisis y diseño se diseñó el esquema de la base de datos relacional.

Integración con la capa de negocio

Para integrar la capa de datos y la capa de negocio se utilizó el patrón Repository, el uso de este patrón facilitó el acceso y manipulación de los datos sin necesidad de escribir consultas SQL directamente en la lógica de negocio.
La creación de repositorios proporciona métodos estándar para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar), permite la extensión de consultas personalizadas según las necesidades, facilita la gestión de los datos y garantiza que las operaciones complejas puedan ser manejadas de manera eficiente y segura.

Capítulo 4: Pruebas y Evaluación del Sistema

4.1. Pruebas

En esta subsección se detalla el proceso de pruebas llevado a cabo para el sistema de gestión de bodega. Las pruebas son una actividad fundamental en el ciclo de vida del desarrollo de software, ya que aseguran que el sistema cumple con los requisitos especificados y satisface las necesidades del usuario final.

4.1.1. Enfoque de Pruebas

Dado el tiempo limitado disponible para la realización de pruebas exhaustivas mediante casos de pruebas detallados, se adoptó un enfoque basado en la especificación de requerimientos. Se utilizaron los flujos alternativos de los casos de uso identificados en la fase de especificación para diseñar y ejecutar las pruebas.

4.1.2. Participación de los Usuarios

Las pruebas del sistema fueron realizadas por cinco usuarios seleccionados de las áreas directamente involucradas con el uso del sistema. Estos usuarios proporcionaron retroalimentación valiosa y aseguraron que las pruebas cubrieran las funcionalidades más críticas y relevantes del sistema. Del área de bodega participaron dos usuarios, cuyos cargos son: jefe de bodega y auxiliar de bodega. Por otra parte, del área comercial tres usuarios incluyendo al jefe de acometidas y medidores, técnico electricista y auxiliar de acometidas y medidores.

4.1.3. Métodos de Pruebas

- Se analizaron los casos de uso del sistema y se identificaron los flujos alternativos relevantes. Estos flujos proporcionaron escenarios diversos para la ejecución de pruebas, permitiendo cubrir diferentes caminos posibles dentro del sistema. Para esto se consideraron las funcionalidades correspondientes a los casos de uso: *Tramitar operación de ingreso, Ingresar solicitud de egreso, Aprobar solicitud de egreso y Tramitar operación de egreso.*
- Se ingresaron datos en el sistema que provocaran la ejecución de diferentes flujos alternativos. Este método aseguró que se probaran todas las rutas críticas y alternativas

definidas en los casos de uso.

4.1.4. Resultados de las Pruebas

Al realizar las pruebas, se logró identificar y solucionar varios defectos del sistema, garantizando que todos los flujos definidos funcionen correctamente y cumplan con los requisitos. Además, cabe mencionar que la especificación de requisitos fue una herramienta valiosa que permitió organizar y estructurar las pruebas de tal forma que se asegure la cobertura completa de las funcionalidades del sistema.

4.2. Evaluación del sistema

La evaluación del sistema se realizó utilizando el cuestionario SUS (System Usability Scale), el cuál sirve medir la usabilidad del sistema. Esta evaluación permitió obtener una visión cuantitativa de la experiencia del usuario.

4.2.1. Metodología de Evaluación

El cuestionario SUS fue utilizado para evaluar qué tan fácil de usar es el sistema. Este cuestionario permite obtener una medida cuantitativa de la usabilidad del sistema y de esta forma comprender la percepción de los usuarios finales.

4.2.2. Participantes de la Evaluación

Para evaluar el sistema se consideraron los mismos cinco funcionarios que participaron en las pruebas de aceptación. La elección de este grupo reducido de participantes se justifica por las siguientes razones:

- Los participantes fueron seleccionados considerando su experiencia y la relación que tienen con la gestión de bodega. Por lo tanto, son considerados representativos de los usuarios finales del sistema.
- La disponibilidad del personal es limitada debido a las responsabilidades operativas de los funcionarios.

4.2.3. Resultados de la Evaluación

Los resultados del cuestionario SUS mostraron que el sistema es fácil de usar, con una puntuación promedio de 91, dicho valor calculado a partir de los puntajes individuales de cada participante. En la Figura 4.1 se puede apreciar los resultados obtenidos.

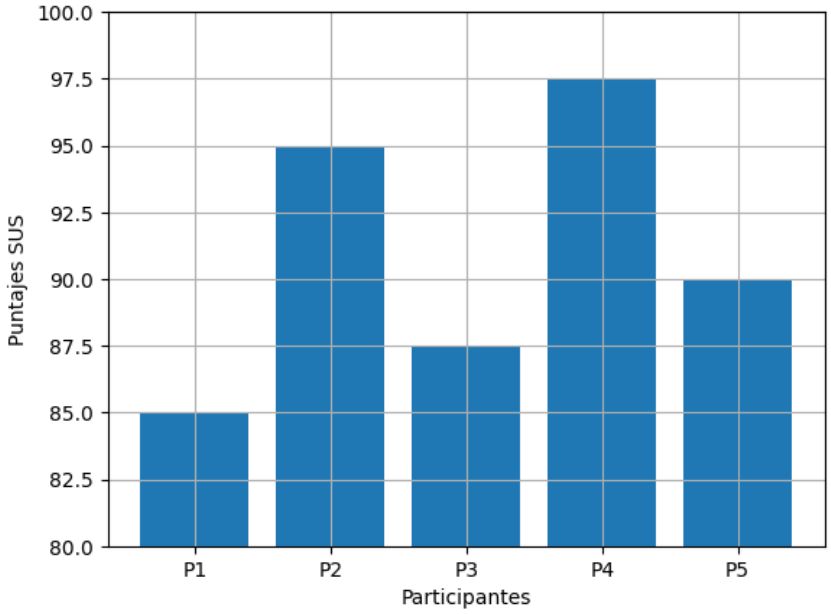


Figura 4.1: Resultados del cuestionario SUS

Estos puntajes sugieren que los usuarios encontraron el sistema fácil de usar y accesible. Considerando las respuestas obtenidas, se puede decir lo siguiente con respecto a la claridad y simplicidad, consistencia, facilidad de aprendizaje y satisfacción:

- Claridad y simplicidad: Los usuarios apreciaron la claridad de la interfaz y la simplicidad de las operaciones básicas.
- Consistencia: La consistencia en el diseño de las pantallas y la navegación fue destacada positivamente.
- Facilidad de aprendizaje: Los usuarios manifestaron que el sistema es fácil de aprender destacando la simplicidad de sus pantallas.
- Satisfacción: Los usuarios participantes expresaron satisfacción con el sistema en general.

4.2.4. Análisis de Resultados

El análisis de los resultados del cuestionario SUS sugiere que el Sistema Integral de Gestión de Bodega cumple con los objetivos de usabilidad establecidos. Los usuarios encontraron el sistema intuitivo y fácil de usar, lo cual es crucial para su adopción y efectividad en el entorno de trabajo. Sin embargo, se identificaron algunas áreas de mejora basadas en los comentarios de los usuarios:

- Mejora de la Navegación: Simplificar la navegación entre diferentes secciones del sistema para mejorar la eficiencia.
- Ampliación de Opciones de Informes: Incluir opciones de informes personalizados para satisfacer mejor las necesidades de los usuarios y proporcionar una mayor visibilidad de los datos.

La evaluación del Sistema Integral de Gestión de Bodega utilizando el cuestionario SUS demostró que el sistema proporciona una experiencia de usuario positiva. Las sugerencias identificadas durante la evaluación proporcionan un camino claro para optimizar aún más el sistema y asegurar su éxito continuo en la empresa.
Con respecto a los requisitos no funcionales especificados, no se realizaron pruebas específicas para medir los atributos de calidad importantes para el sistema debido a limitaciones de tiempo, sin embargo, se tomaron decisiones de diseño e implementación para solventar estos aspectos desde el inicio del desarrollo del sistema.

- Seguridad: Se emplearon herramientas y prácticas recomendadas para garantizar la seguridad del sistema, lo que incluyó una robusta implementación de la autenticación y autorización del sistema y el uso de prácticas seguras de manipulación de datos sensibles.
- Rendimiento: Se emplearon patrones de diseño para optimizar el rendimiento, lo que incluyó cachés para mejorar la velocidad de recuperación de los datos utilizados frecuentemente y la optimización de las consultas a la base de datos para responder rápidamente a las mismas.
- Mantenibilidad: Se utilizaron patrones de diseño para garantizar que el sistema sea fácil de mantener, separando las responsabilidades de manera que cualquier cambio o adición se pueda llevar a cabo de manera eficiente.

Estas decisiones fueron guiadas por la especificación de requerimientos, que destacó la importancia de estos atributos de calidad. Aunque las pruebas específicas no se llevaron a cabo, las decisiones de diseño e implementación reflejan un compromiso con la calidad del sistema en estas áreas.

Capítulo 5: Conclusiones

Este capítulo presenta las conclusiones derivadas del proyecto, evaluando el cumplimiento de los objetivos planteados y los resultados obtenidos. Los objetivos del proyecto se establecieron con el propósito de abordar y solucionar las necesidades específicas de la gestión de bodega en la Empresa Eléctrica Azogues C.A. A continuación, se evalúa el cumplimiento de cada objetivo:

En relación con el primer objetivo específico “Analizar los requerimientos funcionales y no funcionales del sistema”, se logró identificar y documentar exhaustivamente los requisitos del sistema. Este análisis se realizó mediante entrevistas, revisión de procesos y documentos existentes, lo que proporcionó una base sólida para las etapas subsecuentes de análisis, diseño y desarrollo del sistema. La especificación detallada de los requisitos sirvió para guiar el desarrollo y asegurar que el sistema cumpla con las expectativas de los usuarios finales. Dicha especificación de requerimientos generada en este trabajo de titulación puede ser utilizada por cualquier empresa o equipo de desarrollo como base para crear sistemas de gestión de bodega, ofreciendo una referencia versátil y útil para diferentes necesidades y contextos.

Para asegurar el cumplimiento del segundo objetivo específico “Diseñar una arquitectura de software que se adapte a las necesidades específicas de la gestión de bodega, definiendo los componentes y sus interacciones, garantizando una estructura modular y escalable del sistema” se diseñó una arquitectura de software que pueda adaptarse fácilmente a los requisitos únicos de la gestión de bodega y que pueda ampliarse o modificarse fácilmente según sea necesario.
El análisis sirvió como base para el diseño de la arquitectura del sistema. Esta arquitectura se basa en la separación por capas y el uso de patrones de diseño, garantizando una estructura modular y escalable que facilita el mantenimiento del código.

En cuanto al tercer objetivo específico, “Construir el sistema de acuerdo con la arquitectura definida”, se desarrolló el sistema utilizando tecnologías modernas como Angular para el frontend, Spring Boot para el backend y PostgreSQL para la gestión de bases de datos. La implementación se llevó a cabo siguiendo estrictamente las directrices del modelo de diseño, asegurando que el código desarrollado estuviera completamente alineado con la arquitectura

establecida, lo que facilitó la creación de un sistema modular y escalable.
Para las pruebas de aceptación se contó con la participación de usuarios expertos en el negocio, quienes son los usuarios finales del sistema. Las pruebas se realizaron con datos reales de la empresa, lo cual permitió localizar y corregir errores, como resultado final se reportó que el sistema cumple con los requerimientos establecidos y considerando la retroalimentación de los usuarios se proporcionó una guía para futuras mejoras y adaptaciones del sistema. Además, se realizó una evaluación del sistema a través del cuestionario SUS. Los resultados de esta evaluación indicaron que el sistema resulta ser intuitivo y fácil de usar.

Finalmente, con respecto al cumplimiento del objetivo general “Construir un sistema de software para la gestión de bodegas que automatice los procesos del área de bodega de la Empresa Eléctrica Azogues C.A.” se puede concluir que el sistema desarrollado ha permitido una automatización notable de los procesos críticos de la gestión de bodega, disminuyendo la dependencia de procedimientos manuales y por otro lado la digitalización y seguimiento de las solicitudes contribuyó al control automatizado de los inventarios. Estas funcionalidades permitieron un seguimiento preciso y en tiempo real de los bienes, materiales y suministros de oficina, así como eficiencia en el trámite de las solicitudes emitidas. Por último, con respecto a la metodología utilizada cabe mencionar que demostró ser la adecuada para asegurar la calidad del producto final. El Proceso Racional Unificado (RUP) al caracterizarse por ser iterativo e incremental, garantizó que cada disciplina desde el modelado del negocio hasta la implementación se llevara a cabo con un alto nivel de calidad y control, así como también, facilitó la identificación temprana de riesgos y la implementación de soluciones oportunas.

5.1. Recomendaciones para Investigaciones Futuras

A partir de la experiencia adquirida y los resultados obtenidos, se proponen las siguientes recomendaciones para investigaciones o desarrollos futuros: Ampliación de funcionalidades, que implica incorporar las funcionalidades adicionales identificadas como necesarias durante la evaluación, tales como opciones avanzadas de informes y personalización de la interfaz; optimización para dispositivos móviles con el propósito de mejorar la interfaz y la experiencia de usuario en dispositivos móviles y de esta forma aumentar la accesibilidad y flexibilidad del sistema; integración con otros sistemas corporativos de la empresa para proporcionar una solución más eficiente. Finalmente, realizar evaluaciones continuas con los usuarios finales para identificar áreas de mejora y asegurar que el sistema continúe satisfaciendo las necesidades de la empresa.

Referencias

¿Qué es OAuth 2.0 y para qué sirve? (s.f.). Consultado el 26 de julio de 2024, desde https://auth0.com/es/intro-to-iam/what-is-oauth-2

¿Qué es PostgreSQL? | IBM. (2024, abril). Consultado el 24 de junio de 2024, desde https://www.ibm.com/mx-es/topics/postgresql

¿Qué es un WMS (sistema de gestión de almacenes)? (s.f.). Consultado el 26 de julio de 2024, desde https://www.oracle.com/es/scm/logistics/warehouse-management/what-is-warehouse-management/

¿Qué son SSL, TLS y HTTPS? | DigiCert. (s.f.). Consultado el 26 de julio de 2024, desde https://www.digicert.com/es/what-is-ssl-tls-and-https

Abuchar Porras, A. (2023). *Metodologías ágiles para el desarrollo de software*. Universidad Distrital Francisco Jose de Caldas. Consultado el 9 de junio de 2024, desde https://search.worldcat.org/es/title/1388501884

Angular. (s.f.). Consultado el 28 de mayo de 2024, desde https://docs.angular.lat/

Anwar, A. (2014). A Review of RUP (Rational Unified Process). *International Journal of Software Engineering*, *5*. Consultado el 27 de junio de 2024, desde https://es.slideshare.net/slideshow/a-review-of-rup-rational-unified-process/59301856

Business Software, Business Management Software | NetSuite. (s.f.). Consultado el 27 de diciembre de 2023, desde https://www.netsuite.com/portal/home.shtml

Cabrera Pérez, R. J. (2020, enero). *Sistema de procesamiento de transacciones (TPS) orientado a la web para el control de bodega proveeduría de la universidad regional autónoma de los andes "UNIANDES" del cantón Santo Domingo* [bachelorThesis] [Accepted: 2020-01-22T22:22:36Z]. Consultado el 14 de diciembre de 2023, desde https://dspace.uniandes.edu.ec/handle/123456789/10860

Cárdenas Lara, L. A., Chalco Loya, A. P., & Rosas Lara, M. L. (2019). Control de Existencias de Bodega para la Universidad Central Del Ecuador.

Carrera Proaño, H. A., & Quito Chuquin, D. A. (2020, enero). *Desarrollo de un sistema interno para gestión y control de insumos, en la Zona 9 Distrito 17D04 (Ministerio de Salud Pública) ubicado en el sector La Vicentina* [bachelorThesis] [Accepted: 2020-02-07T17:15:22Z]. Consultado el 26 de diciembre de 2023, desde http://dspace.ups.edu.ec/handle/123456789/18372

Castillo De La Torre, O. D., & Guinet Huerta, S. A. (2023). *Sistema de gestión de almacenes para el control de existencias de una empresa comercializadora a través de solucio-*

nes móviles, integrada con el ERP [Tesis doctoral, Universidad Peruana de Ciencias Aplicadas (UPC)].

Castillo Sarzosa, F. D. (2017). *Desarrollo de un sistema de inventarios para la empresa Aldera Diseños usando la metodología del Proceso Unificado Racional RUP* [Tesis doctoral, Pontificia Universidad Católica del Ecuador].

Contreras Paredes, J. C. (2017). *Desarrollo de un sistema encargado de la gestión de información de clientes para la fundación FUDRINE usando la metodología del Proceso Racional Unificado RUP* [Tesis doctoral, Pontificia Universidad Católica del Ecuador].

Daniele, M., & Romero, D. (2006). CONSTRUCCIÓN DE UN FRAMEWORK PARA LA ENSEÑANZA DE PATRONES DE DISEÑO.

Dragos, P. (2021). Journal of Business and Economics. *Overview of the Agile Rational Unified Process (Rup) in the Context of Software Development Projects*, *12*, 110. Consultado el 9 de junio de 2024, desde https://repository.unpak.ac.id/tukangna/repo/file/files-20211015150215.pdf#page=107

draw.io. (s.f.). Consultado el 27 de junio de 2024, desde https://app.diagrams.net/

dubey, A. (2024, mayo). Building Modern Web Applications With Angularjs And Spring Boot. Consultado el 27 de julio de 2024, desde https://medium.com/@amandubey_6607/building-modern-web-applications-with-angularjs-and-spring-boot-fc720d8c4428

Dynamics 365 Business Central. (s.f.). Consultado el 27 de diciembre de 2023, desde https://insightsoftware.com/microsoft/dynamics-365-business-central/

Empresa Eléctrica Azogues C.A. (s.f.). Consultado el 26 de diciembre de 2023, desde https://eea.gob.ec/

Englander, I. (2021). *The architecture of computer hardware, systems software, & networking: an information technology approach* (6. ed). John Wiley.

Enn, M. (2023, julio). Spring Boot vs Spring MVC. Consultado el 24 de junio de 2024, desde https://medium.com/@mohamed.enn/spring-boot-vs-spring-mvc-6a7981e46062

ERP y CRM de fuente abierta | Odoo. (s.f.). Consultado el 27 de diciembre de 2023, desde https://www.odoo.com/es_ES

Fernández Samprieto, I. I. (2017). Implementació d'un sistema ERP per a una empresa de serveis informàtics.

Garibay, A. (2023, febrero). Notas Sobre el Modelo Cliente-Servidor. Consultado el 25 de junio de 2024, desde https://medium.com/@aarongaribay/notas-sobre-el-modelo-cliente-servidor-b29c30d2e855

Garrido Tejero, A. (2021). Uso del patrón repositorio y unidad de trabajo [Accepted: 2021-06-09T08:28:12Z Publisher: Universitat Politècnica de València]. Consultado el 9 de junio de 2024, desde https://riunet.upv.es/handle/10251/167635

Getting started with Angular | MDN. (2024, abril). Consultado el 24 de junio de 2024, desde https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_getting_started

Group, P. G. D. (2024, junio). PostgreSQL. Consultado el 24 de junio de 2024, desde https://www.postgresql.org/

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The unified software development process*. Addison-Wesley.

Jaramillo, W. (2016). *Aplicación de la metodología RUP y el patrón de diseño MVC en la construcción de un sistema de gestión académica para la Unidad Educativa Ángel De La Guarda* [Tesis doctoral, Pontificia Universidad Católica del Ecuador]. Consultado el 17 de junio de 2024, desde https://www.studocu.com/pe/document/servicio-nacional-de-adiestramiento-en-trabajo-industrial/calidad-de-software/documento-disertacion-wendy-jaramillo/62716936

Kruchten, P. (1999). *The rational unified process*. Addison-Wesley.

LDAP.com. (s.f.). Consultado el 26 de julio de 2024, desde https://ldap.com/

Márquez Reyes, D. J. (2023, marzo). Todo lo que necesitas saber sobre API Rest: Glosario de términos esenciales y más. Consultado el 24 de junio de 2024, desde https://dev.to/dennysjmarquez/todo-lo-que-necesitas-saber-sobre-api-rest-glosario-de-terminos-esenciales-y-mas-29pc

Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: a comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504-1522. https://doi.org/10.1007/s13198-023-01958-5

Patrones de diseño / Design patterns. (s.f.). Consultado el 26 de junio de 2024, desde https://refactoring.guru/es/design-patterns

Ruiz Salvatierra, A. O. (2019). *Diagramación de los procesos de la bodega para la elaboración de un software de gestión, en la empresa Electconstru S.A.* [bachelorThesis]. Babahoyo, UTB 2019 [Accepted: 2019-10-11T00:00:05Z]. Consultado el 13 de diciembre de 2023, desde http://dspace.utb.edu.ec/handle/49000/6898

SAP ERP: ERP probado y testeado en el tiempo. (s.f.). Consultado el 26 de diciembre de 2023, desde https://www.sap.com/latinamerica/products/erp/what-is-sap-erp.html

Spring Boot. (s.f.). Consultado el 28 de mayo de 2024, desde https://spring.io/projects/spring-boot

Spring Framework. (s.f.). Consultado el 24 de junio de 2024, desde https://spring.io/projects/spring-framework

Spring Security. (s.f.). Consultado el 26 de junio de 2024, desde https://spring.io/projects/spring-security

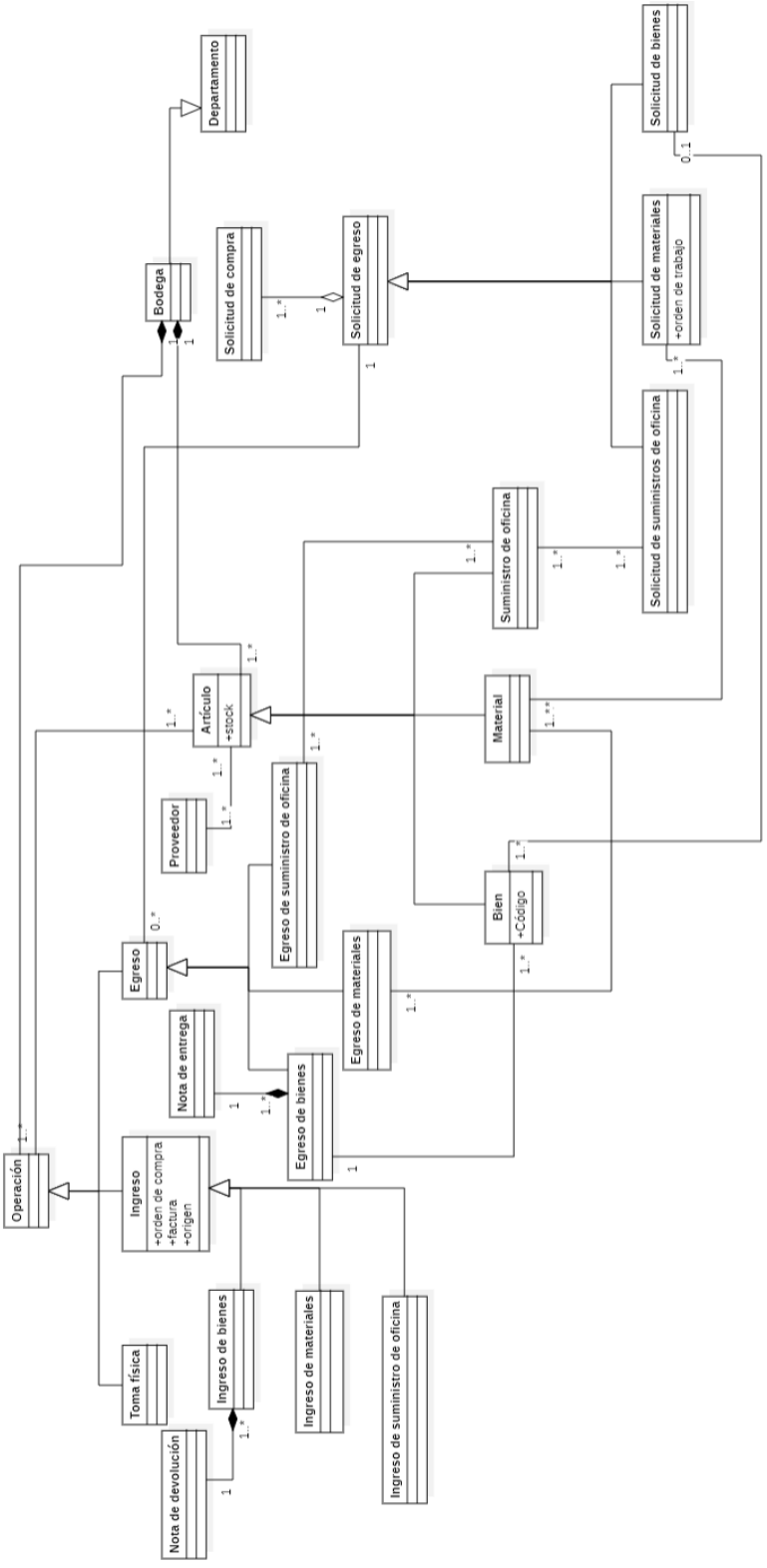
Token web JSON (JWT) | IBM. (2024, junio). Consultado el 26 de junio de 2024, desde https://www.ibm.com/docs/en/cics-ts/6.x?topic=cics-json-web-token-jwt

Tulaskar, D. P., Kale, P. D., Nemane, S. G., Sharma, A. J., & More, P. (2022). An Automated Warehouse Management System. *Journal of Scientific Research and Reports*, 41-49. https://doi.org/10.9734/jsrr/2022/v28i730534

Vivar Maldonado, M. J. (2015). *Sistema de gestión de bodegas.- Aplicación web y móvil* [bachelorThesis]. Espol [Accepted: 2017-08-04T20:51:52Z]. Consultado el 17 de junio de 2024, desde http://www.dspace.espol.edu.ec/handle/123456789/39924

Anexos

Anexo A: Modelo de Dominio



Anexo B: Cuestionario SUS

Cuestionario de Usabilidad

El objetivo de este cuestionario es comprender la opinión de los participantes acerca del Sistema de Gestión de Bodega, mediante el análisis de su percepción.

* Indica que la pregunta es obligatoria

Creo que me gustaría utilizar este sistema con frecuencia. *

	1	2	3	4	5	
Completamente en Desacuerdo	◯	◯	◯	◯	◯	Completamente de Acuerdo

El sistema me pareció innecesariamente complejo. *

	1	2	3	4	5	
Completamente en Desacuerdo	◯	◯	◯	◯	◯	Completamente de Acuerdo

Creo que el sistema es fácil de usar. *

	1	2	3	4	5	
Completamente en Desacuerdo	◯	◯	◯	◯	◯	Completamente de Acuerdo

Creo que necesitaría el soporte de un técnico para poder utilizar este sistema. *

	1	2	3	4	5	
Completamente en Desacuerdo	◯	◯	◯	◯	◯	Completamente de Acuerdo

Creo que las diversas funciones del sistema se encuentran bien integradas. *

	1	2	3	4	5	
Completamente en Desacuerdo	◯	◯	◯	◯	◯	Completamente de Acuerdo

Opino que hubo demasiada inconsistencia en el sistema. *

	1	2	3	4	5	
Completamente en Desacuerdo	○	○	○	○	○	Completamente de Acuerdo

Me imagino que la mayoría de la gente aprendería a utilizar este sistema rápidamente. *

	1	2	3	4	5	
Completamente en Desacuerdo	○	○	○	○	○	Completamente de Acuerdo

El sistema me pareció muy complicado de utilizar. *

	1	2	3	4	5	
Completamente en Desacuerdo	○	○	○	○	○	Completamente de Acuerdo

Me sentí muy seguro al usar el sistema. *

	1	2	3	4	5	
Completamente en Desacuerdo	○	○	○	○	○	Completamente de Acuerdo

Necesitaba aprender muchas cosas antes de poder empezar a utilizar este sistema. *

	1	2	3	4	5	
Completamente en Desacuerdo	○	○	○	○	○	Completamente de Acuerdo

Anexo C: Manual de Usuario

2024

Manual de Usuario

SISTEMA INTEGRAL DE GESTIÓN DE BODEGA

BEATRIZ FLORES ROSALES

Contenido

1. Introducción

Este manual está diseñado para ayudar a los usuarios a utilizar el Sistema Integral de Gestión de Bodega, desarrollado para la Empresa Eléctrica Azogues C.A. Este sistema permite gestionar de manera eficiente las operaciones de bodega, incluyendo la entrada y salida de artículos, solicitudes de egreso, y la gestión de proveedores y artículos.

2. Requisitos del Sistema

- Navegador Web: Compatible con los navegadores más recientes (Chrome, Firefox, Edge).
- Conexión a Internet: Requerida para acceder al sistema a través del navegador.
- Credenciales de Usuario: Proporcionadas por el administrador del sistema.

3. Acceso al Sistema

1. Abra su navegador web y diríjase a la URL del sistema.
2. Ingrese su nombre de usuario y contraseña en la pantalla de inicio de sesión.
3. Haga clic en "Iniciar sesión" para acceder al sistema.

Ilustración 1. Inicio de sesión

4. Módulos Principales

4.1. Gestión de Artículos

- Registrar Artículo: Permite registrar nuevos artículos en el inventario.
  - Navegue a la sección de "Gestión de Artículos".
  - Haga clic en "Registrar Artículo".
  - Complete los campos requeridos: Tipo de artículo, Tipo de seguimiento, Nombre, Descripción.

- Haga clic en "Registrar artículo" para añadir el artículo.

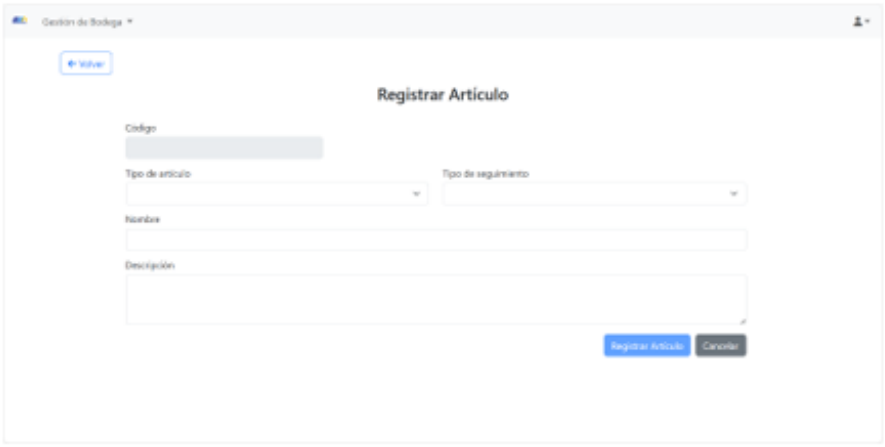


Ilustración 2. Registrar artículo

- Actualizar Artículo: Modifica la información de un artículo existente.
  - Navegue a la sección de "Gestión de Artículos".
  - Haga clic en "Actualizar Artículo".
  - Haga clic en "Buscar artículo" y haga clic en "Seleccionar" en el artículo requerido.
  - Realice los cambios necesarios y guarde los ajustes.

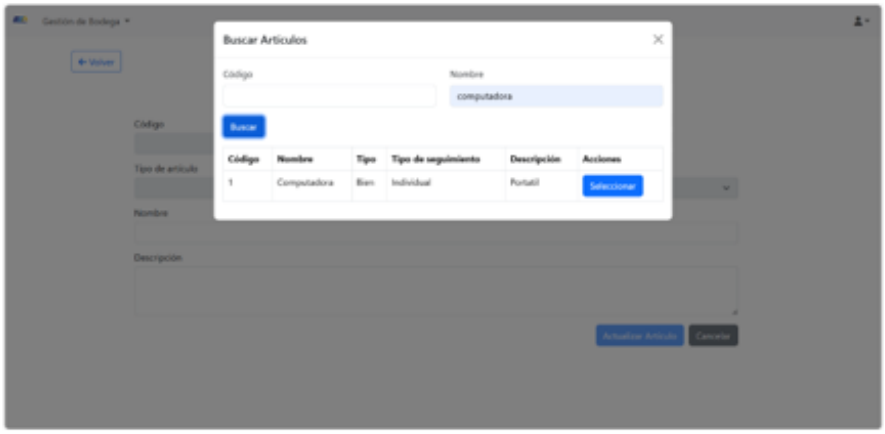


Ilustración 3. Buscar artículos

3

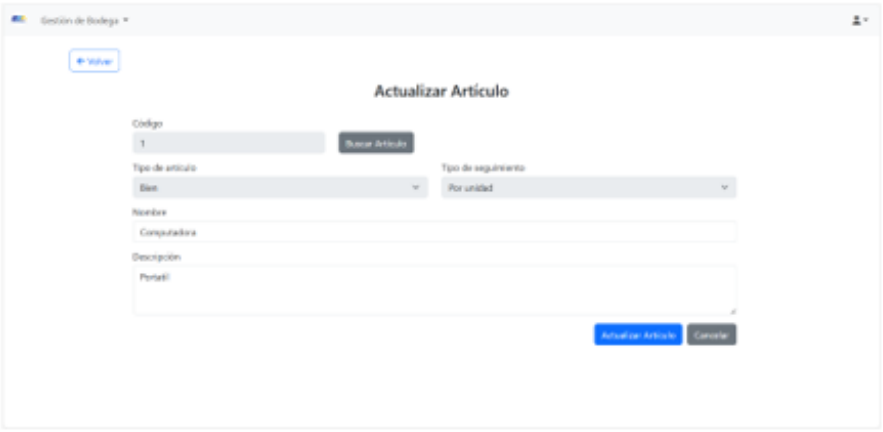


Ilustración 4. Actualizar artículo

4.2. Gestión de Proveedores

- Registrar Proveedor: Añada nuevos proveedores al sistema.
  - Vaya a "Gestión de Proveedores" y haga clic en "Nuevo Proveedor".
  - Complete la información del proveedor, incluyendo nombre, RUC, y datos de contacto.
  - Guarde los detalles para finalizar el registro.

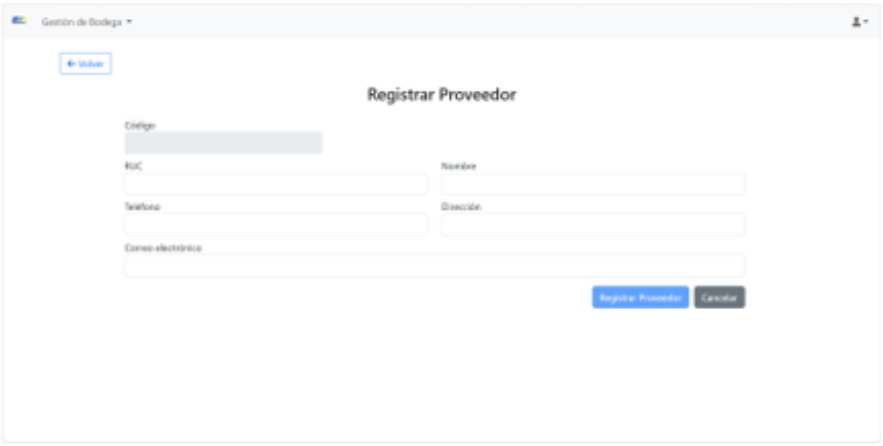


Ilustración 5. Registrar proveedor

4

4.3. Solicitudes de Egreso

- Crear Solicitud de Egreso: Solicite artículos del inventario para proyectos o actividades específicas.
  - Ingrese a "Solicitudes de Egreso".
  - Haga clic en "Nueva Solicitud".
  - Complete los detalles de la solicitud, seleccionando los artículos y cantidades requeridas.
  - Envíe la solicitud para su aprobación.

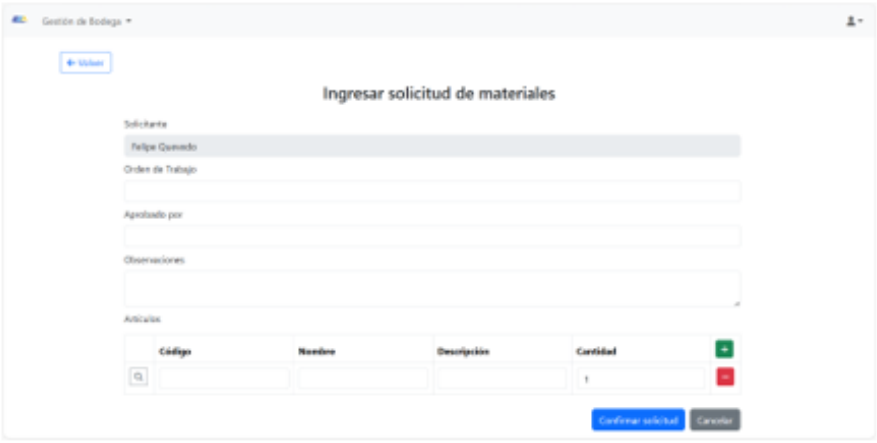


Ilustración 6. Ingresar solicitud de egreso

- Aprobar Solicitudes: Gestión de solicitudes pendientes de aprobación. Esta funcionalidad estará accesible para los usuarios asignados con el rol de Aprobar solicitud.
  - Seleccione a "Aprobación de Solicitudes".
  - Revise las solicitudes pendientes y apruébelas o rechácelas según corresponda.

4.4. Operaciones de Bodega

- Ingreso de Artículos: Registra la entrada de artículos en la bodega.
  - Navegue a "Tramitar Operación de Ingreso".
  - Complete la información del ingreso, como proveedor y detalles de los artículos.
  - Guarde el ingreso para actualizar el inventario.

5

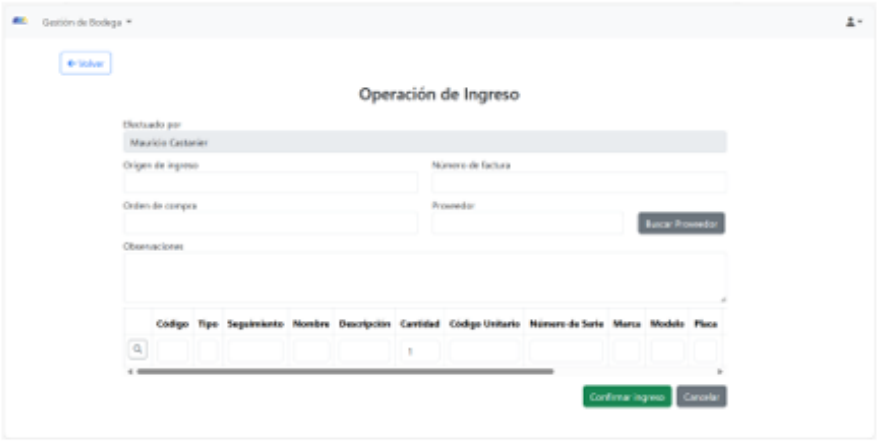


Ilustración 7. Tramitar operación de ingreso

- Egreso de Artículos: Registra la salida de artículos de la bodega.
  - Navegue a "Tramitar Operación de Egreso".
  - Seleccione los artículos a egresar y confirme la operación.

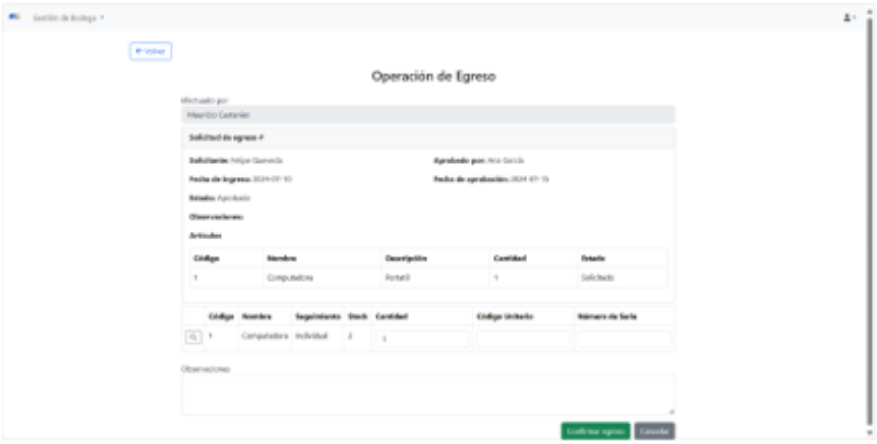


Ilustración 8. Tramitar operación de egreso

5. Seguridad y Acceso

El sistema utiliza autenticación basada en JWT (JSON Web Token) para asegurar que solo los usuarios autorizados puedan acceder a las funcionalidades del sistema. Cada usuario tiene permisos asignados según su rol, los cuales definen las acciones que pueden realizar dentro del sistema.

6

6. Resolución de Problemas

- Problemas de Acceso: Si no puede acceder al sistema, verifique sus credenciales o contacte al administrador del sistema.
- Errores en Operaciones: Revise que todos los campos obligatorios estén completos y correctamente llenados. En caso de errores persistentes, comuníquese con el soporte técnico.

7. Contacto y Soporte

Para cualquier problema técnico o solicitud de asistencia, contacte al departamento de soporte de TI al correo electrónico mzambrano@eea.gob.ec o llame al 2240377 extensión 170.

7