UCUENCA

Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería en Ciencias de la Computación

Aplicación de un Enfoque Integral para el Desarrollo de Software basado en RUP: Sistema de Gestión de Incidentes y Administración de Servicios

Trabajo de titulación previo a la obtención del título de Ingeniero en Ciencias de la Computación

Autores:

Jorge Fabricio Criollo Criollo

Paul Andrés Villalta Heredia

Director:

Miguel Ángel Zúñiga Prieto

ORCID: 0000-0001-9369-1813

Cuenca, Ecuador

2024-09-16

Resumen

El crecimiento del consumo de software, impulsado por la transformación digital y la competencia, ha incrementado la demanda de soluciones personalizadas. Sin embargo, las empresas desarrolladoras enfrentan desafíos en la gestión de servicios y soporte, como la comunicación deficiente y la falta de canales confiables para el intercambio de información. Esta tesis propone un enfoque integral basado en el Proceso Unificado de Desarrollo (RUP) para crear un Sistema de Gestión de Incidentes y Administración de Servicios, utilizando como caso de estudio a Vimasistem.ltda, una empresa ecuatoriana que brinda servicios tecnológicos para los sectores financiero y comercial. La metodología fue incremental y basada en RUP, abarcando las fases de inicio, elaboración, construcción y transición.

Se realizó un análisis exhaustivo del modelo de negocio y los flujos de trabajo de Vimasistem.ltda para identificar los requisitos del sistema, lo cual permitió diseñar una arquitectura lógica detallada. El sistema desarrollado mejoró la gestión de cambios en productos de software y facilitó la administración de clientes, personal y productos. La validación empírica de usabilidad y experiencia de usuario, mediante los cuestionarios SUS y UEQ, mostró que la aplicación aborda las deficiencias de las soluciones actuales y mejora la eficiencia operativa y la satisfacción del cliente. En conclusión, la tesis demuestra la viabilidad de desarrollar una solución efectiva y adaptable para la gestión de incidentes y servicios en empresas de desarrollo de software, contribuyendo a su éxito y crecimiento en el competitivo mercado actual.

Palabras clave del autor: arquitectura lógica, usabilidad de sistemas, eficiencia operativa

Abstract

The growth in software consumption, driven by digital transformation and competition, has increased the demand for customized solutions. However, software development companies face challenges in service management and support, such as poor communication and the lack of reliable channels for information exchange. This thesis proposes a comprehensive approach based on the Rational Unified Process (RUP) to create an Incident Management and Service Administration System, using Vimasistem.ltda, an Ecuadorian company providing technology services for the financial and commercial sectors, as a case study. The methodology was incremental and based on RUP, covering the phases of inception, elaboration, construction, and transition.

An exhaustive analysis of the business model and workflows of Vimasistem.ltda was conducted to identify system requirements, which enabled the design of a detailed logical architecture. The developed system improved change management in software products and facilitated the administration of clients, personnel, and products. Empirical validation of usability and user experience, using the SUS and UEQ questionnaires, showed that the application addresses the shortcomings of current solutions and enhances operational efficiency and customer satisfaction. In conclusion, the thesis demonstrates the feasibility of developing an effective and adaptable solution for incident and service management in software development companies, contributing to their success and growth in the competitive current market.

Author keywords: logical architecture, system usability, operational efficiency

Índice de contenido

Índice de figuras

Índice de tablas

Agradecimientos

Jorge Fabricio Criollo Criollo

Quiero expresar mi gratitud a la Universidad de Cuenca por proporcionar un entorno que ha fomentado mi crecimiento académico a lo largo de estos años.

Agradezco profundamente a mi tutor, Ing. Miguel Ángel Zúñiga, por dedicar su tiempo y ofrecer sus conocimientos y orientación para la realización de este proyecto.

A mis padres y hermanos, les agradezco de corazón por su incondicional apoyo y constante aliento en todo momento.

Paul Andrés Villalta Heredia

Agradezco a la Universidad de Cuenca por haber propiciado un ambiente de crecimiento académico durante estos años de estudio.

A mi tutor, Ing. Miguel Ángel Zúñiga, por haber brindado su tiempo y orientación para el desarrollo de este trabajo.

A mis padres y hermanos por haber sido un apoyo incondicional en todo momento.

1. Introducción

El avance vertiginoso de la tecnología y la creciente necesidad de las empresas de mantenerse competitivas en un entorno globalizado han impulsado el consumo de software de manera exponencial (Trenkle, 2020). Este fenómeno es producto de la transformación digital, la automatización de procesos y la competencia feroz entre empresas (Lagerburg, 2023). Las pequeñas empresas, en particular, han encontrado en el software una herramienta poderosa para competir y ofrecer experiencias superiores a sus clientes (Sharma et al., 2008). En respuesta a esta demanda, han surgido numerosas empresas de desarrollo de software que ofrecen soluciones altamente personalizadas para satisfacer las necesidades específicas de cada negocio (Silvestro et al., s.d.).

El consumo creciente de software es una tendencia que se espera continúe en los próximos años, lo que generará una demanda cada vez mayor de soluciones de software a medida (Kruchten, 2004). Empresas como Foldcraft, Ubiquiti Networks, Monex, Bridgestone Corporation y Zoom han incorporado soluciones informáticas para mejorar la satisfacción de sus clientes y gestionar sus servicios de manera más eficiente (Nunes & Russo, 2019). Estas aplicaciones han optimizado operaciones y mejorado la gestión de procesos, desde la visibilidad y administración de procesos en Foldcraft (Rêgo et al., 2022) hasta la eficiencia en la gestión de incidentes en Ubiquiti Networks (Kallmuenzer et al., 2024), la reducción de tiempos de comercialización en Monex (*Managing Information Technology | SpringerLink*, 2024), el ahorro de horas de trabajo en Bridgestone Corporation (Osterwalder et al., 2010) y la mejora en la gestión de proyectos en Zoom (Ries, 2017).

No obstante, a pesar de este crecimiento constante, las empresas de desarrollo de software enfrentan desafíos significativos en la gestión de servicios y soporte (dotnet-bot, 2024). La falta de comunicación efectiva y la ausencia de canales confiables para el intercambio de información representan obstáculos importantes (*What Is a REST API?*, 2024). Estas deficiencias afectan directamente la capacidad de las empresas de desarrollo para ofrecer soluciones que satisfagan plenamente las necesidades de sus clientes, limitando el potencial del software en un entorno de crecimiento continuo (*Eclipse Documentation | The Eclipse Foundation*, 2024).

En el mercado actual, existen diversas soluciones de software para la gestión empresarial, como YouTrack (*Introduction to YouTrack | YouTrack Server*, s.d.), ServiceNow (*Product Documentation | ServiceNow*, s.d.), Freshworks (*Freshworks Developer Docs | Freshworks app ecosystem*, 2024) y Asana (*Build an App with Asana*, 2024). Cada una de estas herramientas se especializa en diferentes aspectos de la gestión de proyectos y servicios, desde el seguimiento de incidencias hasta la atención al cliente y la gestión de proyectos. Sin embargo, ninguna de estas soluciones logra cubrir al 100% las necesidades específicas de

cada empresa, ya sea por la falta de opciones de personalización o por los costos asociados (Olson & Kesharwani, 2010). Además, las políticas de protección de datos hacen que las empresas sean reacias a adoptar soluciones de terceros para el intercambio de información (Soveizi et al., 2023).

En este contexto, esta tesis propone un enfoque integral para el desarrollo de software basado en el Proceso Unificado de Desarrollo (RUP), orientado a la creación de un sistema de gestión de incidentes y administración de servicios para empresas de desarrollo de software (Kruchten, 2004). Utilizando como caso de estudio a la empresa Vimasistem.ltda, una compañía ecuatoriana que brinda servicios tecnológicos para el sector financiero y comercial (*Inicio | Vimasistem*, 2024), este sistema busca optimizar las operaciones relacionadas con la mesa de ayuda, la gestión de cambios en productos de software y la administración de clientes, personal y productos de software. El alcance de esta implementación estará limitado a las características de mayor prioridad, debido al tiempo disponible para finalizar este trabajo de titulación.

Metodológicamente, se adoptará un enfoque incremental basado en RUP, que comprende las fases de inicio, elaboración, construcción y transición (Kruchten, 2004). Durante la fase inicial se definirán el alcance y los requisitos del proyecto; en la fase de elaboración, estos se refinarán y se desarrollará un plan detallado; la fase de construcción se centrará en la implementación y prueba del software; y finalmente, la fase de transición se enfocará en la instalación del software en el entorno de producción. Cada fase incluirá disciplinas iterativas como el modelado de negocio, la definición de requisitos, el análisis, el diseño, la implementación, las pruebas, el despliegue, la gestión del cambio y la configuración, así como la gestión del proyecto y del entorno (Kruchten, 2004).

Para evaluar la usabilidad y la experiencia de usuario, se realizará una evaluación empírica en una población de ingenieros de software que trabajen en proyectos de Vimasistem (Ferreira & Acuña, s.d.). Se utilizarán el cuestionario de usabilidad SUS(Brooke, 1996) y el cuestionario de experiencia de usuario (UEQ)(*User Experience Questionnaire (UEQ)*, 2024) para recoger los datos necesarios. El objetivo principal de esta tesis es analizar, diseñar, desarrollar e implementar un sistema de gestión de incidentes y administración de servicios aplicando la metodología RUP, utilizando como caso de estudio a Vimasistem.ltda.

1.1. Objetivo General

Analizar, diseñar, desarrollar e implementar un sistema de gestión de incidentes y administración de servicios para empresas de desarrollo de software aplicando la metodología de Rational Unified Process (RUP): Caso de estudio empresa Vimasistem.

1.2. Objetivos Específicos

- Analizar los requerimientos y procesos empresariales: Identificar y documentar los requerimientos funcionales y no funcionales del sistema.
- Diseñar la Arquitectura del Sistema: Diseñar la arquitectura del sistema, incluyendo vistas arquitectónicas que se consideren relevantes para la comunicación con diferentes partes interesadas.
- Implementar el Sistema: Construir el sistema de acuerdo con la arquitectura definida.
- Probar y validar empíricamente: el sistema en un entorno de usuario final con una población conformada por ingenieros de software trabajando en proyectos de la empresa Vimasistem

1.3. Estructura de la Tesis

La presente tesis se estructura de la siguiente forma:

- Capítulo 1. Introducción: Contexto, identificación del problema, metodología, antecedentes, objetivo general y objetivos específicos.
- Capítulo 2. Marco Teórico y Trabajos Relacionados: Revisión de la literatura y análisis de trabajos previos relacionados con la gestión de incidentes y administración de servicios en el desarrollo de software.
- Capítulo 3. Construcción de la Herramienta: Descripción detallada del desarrollo del sistema de gestión de incidentes y administración de servicios, incluyendo los insumos, modelos de dominio, y casos de uso.
- Capítulo 4. Análisis: Análisis de clases y arquitectura del sistema.
- Capítulo 5. Diseño: Diseño de la arquitectura, incluyendo la capa de presentación, negocio y datos.
- Capítulo 6. Implementación: Detalles sobre la implementación del sistema.
- Capítulo 7. Planificación del Experimento: Objetivo, contexto y ejecución del experimento para validar el sistema.
- Capítulo 8. Resultados: Evaluación del sistema mediante cuestionarios de usabilidad y experiencia de usuario.
- Capítulo 9. Conclusiones: Resumen de los hallazgos, contribuciones y trabajos futuros.

2. Marco Teórico y Trabajos Relacionado

La comprensión de los elementos clave del desarrollo de software en el contexto de la transformación digital y su impacto en las pequeñas empresas es fundamental. Analizando aspectos como la gestión de incidentes y servicios, la metodología “Rational Unified Process (RUP)” y el análisis de modelos de negocio, se establece una base sólida para la creación de

un Sistema de Gestión de Incidentes y Servicios. A continuación, se presentan los seis componentes teóricos principales que sustentan este trabajo.

2.1. Marco Teórico

2.1.1. Transformación Digital y Software Empresarial

En las últimas décadas, la transformación digital ha redefinido la manera en que las empresas operan y compiten en el mercado global. El software se ha convertido en un componente esencial para incrementar la eficiencia operativa, la satisfacción del cliente y la efectividad en la toma de decisiones estratégicas. Según estudios recientes, las organizaciones que adoptan tecnologías digitales avanzadas pueden lograr un crecimiento significativo en productividad y rentabilidad. Este cambio ha impulsado a las pequeñas empresas a invertir en soluciones de software personalizadas para mantenerse competitivas y satisfacer las demandas de los clientes (Kallmuenzer et al., 2024).

Desarrollo de Software para Pequeñas Empresas

La gestión de incidentes y la administración de servicios son componentes críticos para el sector cuya meta principal es el desarrollo de software. La gestión de incidentes se refiere al proceso de identificar, analizar y corregir problemas que pueden surgir en un sistema de software, mientras que la administración de servicios abarca la planificación, entrega y soporte de servicios de TI. La implementación de sistemas eficaces para la gestión de incidentes y servicios puede aumentar notablemente la calidad del software, reducir tiempos de inactividad y aumentar la satisfacción del cliente (Kallmuenzer et al., 2024).

Gestión de Incidentes y Administración de Servicios

La gestión de incidentes y la administración de servicios son componentes críticos en el sector del desarrollo de software. La gestión de incidentes se refiere al proceso de identificar, analizar y corregir problemas que pueden surgir en un sistema de software, mientras que la administración de servicios abarca la planificación, entrega y soporte de servicios de TI. La implementación de sistemas eficaces para la gestión de incidentes y servicios puede mejorar significativamente la calidad del software, reducir tiempos de inactividad y aumentar la satisfacción del cliente (*Managing Information Technology* | *SpringerLink*, 2024).

2.1.2. Análisis del Modelo y Flujo del Negocio

El análisis del modelo de negocio es un proceso fundamental para entender cómo una empresa de desarrollo de software puede crear, entregar y capturar valor. Este análisis incluye la identificación de los “segmentos de clientes, las propuestas de valor, los canales de distribución, las relaciones con los clientes, las fuentes de ingresos, los recursos clave, las actividades clave, las asociaciones clave y la estructura de costos” (Osterwalder et al., 2010). Para nuevas empresas y proyectos emergentes en el sector del desarrollo de software, un modelo de negocio bien definido puede ser la diferencia entre el éxito y el fracaso. Un modelo

de negocio efectivo ayuda a las empresas a entender su posición en el mercado, a identificar oportunidades de crecimiento y a diseñar estrategias para la entrega eficiente y rentable de soluciones de software (Ries, 2017).

2.1.3. Rational Unified Process (RUP)

El Rational Unified Process (RUP) es una metodología iterativa de desarrollo de software que proporciona un marco estructurado para el análisis, diseño, implementación y prueba de sistemas de software. Desarrollado por Rational Software Corporation y ampliamente promovido por Philippe Kruchten, RUP se destaca por su capacidad para gestionar proyectos complejos y adaptarse continuamente a los cambios en los requisitos y el entorno de desarrollo. A continuación, se detallan los principales procesos y prácticas de RUP según "The Rational Unified Process: An Introduction" de Philippe Kruchten[5].

2.1.3.1. Principios Fundamentales de RUP:

Desarrollo Iterativo

El desarrollo iterativo es un pilar central de RUP, permitiendo una mejor gestión de riesgos y adaptabilidad. Cada iteración del proyecto produce versiones incrementales del sistema, lo que facilita la obtención de retroalimentación continua y temprana.

Gestión de Requisitos:

La gestión de requisitos en RUP asegura que todas las necesidades del cliente y objetivos del proyecto se documenten y gestionen adecuadamente. Esto incluye la creación de casos de uso detallados y la gestión de cambios en los requisitos.

Arquitectura Basada en Componentes:

RUP promueve una arquitectura basada en componentes, facilitando la reutilización y modularidad del software. Este enfoque contribuye a la creación de sistemas más robustos y flexibles, mejorando el mantenimiento y evolución del software.

Verificación Continua de la Calidad del Software:

La calidad del software se verifica continuamente a través de pruebas automatizadas, revisiones y otros mecanismos de aseguramiento de la calidad, incluyendo pruebas unitarias, de integración, de sistema y de aceptación.

2.1.3.2. Fases del RUP

La metodología RUP está organiza en cuatro etapas principales, cada una de ellas con objetivos y entregables específicos:

Inicio (Inception):

- Objetivo: Definir el alcance del proyecto y obtener la aprobación inicial.
- Actividades: Elaborar el caso de negocio, identificar riesgos clave, definir casos de uso principales y establecer un plan inicial del proyecto.

- Entregables: Documento de visión, lista de riesgos, plan del proyecto inicial y modelo de negocio preliminar.

Elaboración (Elaboration):

- Objetivo: Refinar los requisitos y establecer una arquitectura base, mitigando los riesgos principales.
- Actividades: Desarrollar la arquitectura base, completar los casos de uso detallados y preparar un plan detallado para la fase de construcción.
- Entregables: Modelo de arquitectura de software, modelo de análisis, especificaciones de casos de uso detallados y prototipos funcionales.

Construcción (Construction):

- Objetivo: Desarrollar el sistema completo basado en la arquitectura establecida y los requisitos detallados.
- Actividades: Implementar y probar componentes, integrar el sistema y realizar pruebas de sistema y aceptación.
- Entregables: Código fuente del sistema, documentación técnica, manuales de usuario y un sistema probado.

Transición (Transition):

- Objetivo: Transferir el sistema al entorno de producción y asegurar que cumple con las expectativas del usuario final.
- Actividades: Realizar pruebas beta, corregir errores, preparar la documentación final y capacitar a los usuarios.
- Entregables: Sistema implementado, documentación final y reporte de lecciones aprendidas.

2.1.3.3. Disciplinas del RUP

RUP incluye varias disciplinas que abarcan desde el principio al fin de la vida del desarrollo del software:

- Modelado de Negocios: Entender la estructura y dinámica del negocio, identificar procesos clave y establecer requisitos de alto nivel.
- Requisitos: Capturar y gestionar los requisitos del sistema, incluyendo la creación de casos de uso y el establecimiento de criterios de aceptación.
- Análisis y Diseño: Definir la solución técnica y diseñar la arquitectura del sistema, asegurando que cumple con los requisitos funcionales y no funcionales.
- Implementación: Construir y probar componentes del sistema, asegurando que la implementación cumple con el diseño y los requisitos.

- Pruebas: Verificar y validar que el sistema funciona según lo especificado, utilizando diferentes tipos de pruebas a lo largo del ciclo de desarrollo.
- Despliegue: Preparar el sistema para su entrega y uso en el entorno de producción, incluyendo instalación, configuración y capacitación de usuarios.
- Gestión de Configuración y Cambio: Controlar las versiones del software y gestionar los cambios de manera eficiente para asegurar la integridad del sistema.
- Gestión del Proyecto: Planificar y gestionar el proyecto, incluyendo estimaciones de tiempos y costos, asignación de recursos y seguimiento del progreso.
- Gestión del Entorno: Proporcionar y mantener el entorno de desarrollo, incluyendo las herramientas e infraestructura necesarias para soportar el proceso de desarrollo.

2.1.4. Entorno de Desarrollo

Para la creación de sistemas de gestión de incidentes y administración de servicios, existen diversas herramientas y entornos de desarrollo integrados (IDEs) que facilitan y optimizan el proceso de creación y despliegue de software. A continuación, se detallan varias opciones destacadas, junto con sus características y beneficios.

2.1.4.1. Herramientas para el Desarrollo Backend

.NET Framework (dotnet-bot, 2024):

El .NET Framework es una plataforma de desarrollo integral desarrollada por Microsoft, utilizada principalmente para construir aplicaciones empresariales robustas y escalables. Es ampliamente reconocido por su compatibilidad con aplicaciones empresariales existentes y su capacidad para ofrecer un rendimiento eficiente.

- Compatibilidad y Rendimiento: Altamente compatible con aplicaciones empresariales existentes y ofrece un rendimiento eficiente y escalable.
- Biblioteca de Clases: Proporciona una amplia biblioteca de clases predefinidas para simplificar el desarrollo de funciones complejas.
- Soporte para Servicios Web: Facilita la creación de servicios web y APIs RESTful(What Is a REST API?, 2024), esenciales para la arquitectura moderna de aplicaciones distribuidas.
- Seguridad: Incluye características de seguridad integradas que protegen contra amenazas comunes.

Eclipse(Eclipse Documentation | The Eclipse Foundation, 2024):

Eclipse es un entorno de desarrollo integrado (IDE) gratuito y de código abierto, conocido por su extensibilidad y su soporte para múltiples lenguajes de programación. Es una herramienta con gran popularidad en el mercado de desarrollo debido a su flexibilidad y capacidad de integración con una amplia variedad de herramientas y plugins.

- Extensibilidad: IDE gratuito y de código abierto conocido por su extensibilidad y soporte para múltiples lenguajes de programación como Java(Java Documentation, 2024), C++(C++ Documentation — DevDocs, s.d.), y Python(Welcome to Python.Org, 2024).
- Integración: Compatible con una amplia variedad de herramientas y plugins que mejoran la productividad.

IntelliJ IDEA(Getting started | IntelliJ IDEA Documentation, 2024):

IntelliJ IDEA es un IDE potente y versátil desarrollado por JetBrains, especialmente valorado por sus herramientas avanzadas de refactorización y su ecosistema robusto que soporta una amplia variedad de frameworks y tecnologías.

- Refactorización Potente: Ofrece herramientas avanzadas para la refactorización y desarrollo inteligente de aplicaciones en lenguajes como Java, Kotlin(Kotlin Docs | Kotlin, 2024) y Scala(Learn Scala, 2024).
- Ecosistema Robusto: Soporta una amplia variedad de frameworks y tecnologías, facilitando el desarrollo de aplicaciones complejas.

PyCharm(Getting Started | PyCharm, 2024):

PyCharm es un IDE especializado en el desarrollo en Python, desarrollado por JetBrains. Es ideal para el desarrollo backend con frameworks como Django y Flask, y ofrece herramientas avanzadas que mejoran significativamente la productividad del desarrollador.

- Especialización en Python: IDE especializado en Python, ideal para desarrollo backend con frameworks como Django(Django Documentation | Django Documentation, 2024) y Flask(Welcome to Flask — Flask Documentation (3.0.x), 2024).
- Herramientas Avanzadas: Proporciona características avanzadas para el desarrollo web, incluyendo depuración y gestión de entornos virtuales.

NetBeans(Apache NetBeans Wiki, 2024):

NetBeans es un IDE de código abierto y de acceso gratuito que soporta varios lenguajes de programación, por ello en una opción versátil para el desarrollo de aplicaciones. Ofrece herramientas integradas para edición, depuración y pruebas de aplicaciones.

- Multilenguaje: Soporta varios lenguajes de programación, incluyendo Java, JavaScript(JavaScript | MDN, 2024), PHP(PHP: Documentation, 2024), y C++.
- Funcionalidades Completa: Ofrece herramientas integradas para edición, depuración y pruebas de aplicaciones.

2.1.4.2. Herramientas para el Desarrollo Frontend

Angular(Angular - Introducción a la Documentación de Angular, 2024):

Angular es un framework de desarrollo frontend mantenido por Google, conocido por su capacidad para construir aplicaciones web dinámicas y responsivas. Su arquitectura basada en componentes y su ecosistema robusto lo hacen ideal para proyectos complejos.

- Arquitectura de Componentes: Promueve la reutilización de código y una mejor organización del proyecto a través de una arquitectura basada en componentes.
- Desempeño y Velocidad: Proporciona un alto rendimiento en la renderización de vistas y actualización de datos en tiempo real.
- Ecosistema Robusto: Cuenta con un ecosistema amplio de bibliotecas que facilitan el desarrollo, junto con herramientas que facilitan el mantenimiento de aplicaciones web.
- Soporte para SPA: Facilita la creación de aplicaciones de una sola página, mejorando la navegación y la interacción del usuario sin recargar la página completa.

React(Getting Started – React, 2024):

Biblioteca de JavaScript creada por Facebook, empleada para desarrollar interfaces de usuario. Es conocida por su flexibilidad y eficiencia en la actualización y renderización de componentes.

- Biblioteca Flexible: Biblioteca JavaScript flexible para construir interfaces de usuario, destacada por su eficiencia en la actualización y renderización de componentes.
- Comunidad Activa: Amplio soporte de la comunidad y numerosas bibliotecas complementarias.

Vue.js(Vue.Js, 2024):

Vue.js es un framework progresivo de JavaScript que se destaca por su facilidad de aprendizaje y flexibilidad, lo que facilita la creación de aplicaciones web interactivas y dinámicas.

- Curva de Aprendizaje Suave: Framework progresivo de JavaScript conocido por su facilidad de aprendizaje y flexibilidad.
- Componentes Reutilizables: Facilita la creación de componentes reutilizables y modularización del código.

WebStorm(Getting started with WebStorm | WebStorm Documentation, 2024):

WebStorm es un IDE desarrollado por JetBrains, optimizado para el desarrollo de aplicaciones frontend con frameworks como Angular, React y Vue.js. Ofrece herramientas avanzadas que facilitan la codificación, depuración y prueba de aplicaciones web.

- Optimización para Desarrollo Web: IDE de JetBrains optimizado para el desarrollo de aplicaciones frontend con frameworks como Angular, React y Vue.js.
- Herramientas Avanzadas: Proporciona herramientas avanzadas para codificación, depuración y pruebas de aplicaciones web.

2.1.4.3. Entornos de Desarrollo (IDEs)

Visual Studio(ghogen, 2024):

Visual Studio es un IDE integral desarrollado por Microsoft, conocido por su robustez y amplia gama de características que facilitan el desarrollo de aplicaciones backend.

- Entorno Integral: Ofrece un conjunto completo de herramientas para edición, depuración, compilación y publicación de aplicaciones.
- Depuración Avanzada: Herramientas de depuración avanzadas que mejoran la calidad del código y la eficiencia del desarrollo.
- Integración con .NET: Se integra perfectamente con .NET Framework, proporcionando plantillas y soporte para las últimas características del framework.
- Extensibilidad: Permite personalización mediante una amplia variedad de extensiones y complementos.

Visual Studio Code(Documentation for Visual Studio Code, 2024):

Visual Studio Code es un editor de código fuente creado por Microsoft que es ligero y poderoso, ideal para el desarrollo de aplicaciones frontend y backend.

- Ligero y Rápido: Editor de código fuente ligero que proporciona una experiencia de edición fluida sin sacrificar funcionalidad.
- Soporte Multilenguaje: Admite una amplia variedad de lenguajes de programación, con excelente soporte para HTML(HTML, 2024), CSS, JavaScript y TypeScript.
- Extensibilidad: Gran biblioteca de extensiones que permiten personalizar y ampliar sus capacidades.
- Integración con Git: Ofrece integración nativa con Git y otros sistemas de control de versiones.
- Terminal Integrado: Incluye un terminal integrado que mejora la eficiencia y productividad de los desarrolladores.
- Actualizaciones Frecuentes: Recibe actualizaciones frecuentes con nuevas características y mejoras.

2.1.5. Validación Empírica de Producto de Software

Las validaciones empíricas son una parte fundamental en el proceso de evaluación de la calidad, funcionalidad y usabilidad de los sistemas de software. Esta metodología se fundamenta en la observación y experiencia directa de los usuarios con el sistema, permitiendo recopilar datos precisos y relevantes sobre su rendimiento y aceptación(Ferreira & Acuña, s.d.). A continuación, se detallan los conceptos clave y la aplicación de las validaciones empíricas en la evaluación de software.

2.1.5.1. Concepto de Validaciones Empíricas

Las validaciones empíricas implican el uso de métodos prácticos y observacionales para evaluar un sistema de software a través de pruebas reales con usuarios finales. A diferencia de los métodos teóricos, que se basan en suposiciones y modelos abstractos, las validaciones empíricas recogen datos concretos provenientes de la interacción directa de los usuarios con la plataforma. Este enfoque tiene varios objetivos principales:

Identificación de Problemas Reales:

- Permite detectar y resolver problemas específicos que los usuarios pueden encontrar durante el uso del sistema, mejorando así la calidad general del producto.
- Medición de la Satisfacción del Usuario: Evalúa cómo los usuarios perciben la facilidad de uso, la eficiencia y la efectividad del sistema, proporcionando información valiosa para mejorar la experiencia del usuario.

Recopilación de Datos Objetivos:

- Obtiene información cuantitativa y cualitativa que refleja la experiencia real de los usuarios, lo que es crucial para tomar decisiones informadas sobre mejoras y ajustes necesarios.

Validación de Requisitos:

- Asegura que se cumple con los requisitos funcionales y de usabilidad definidos en las etapas iniciales del proyecto, garantizando que los objetivos del proyecto se cumplen adecuadamente y que las expectativas del usuario sean satisfactorias.

2.1.5.2. Herramientas Utilizadas en las Validaciones Empíricas

Sistema de Usabilidad de Software (SUS)

El Sistema de Usabilidad de Software (SUS) es una herramienta de evaluación que proporciona una medición rápida y fiable de la usabilidad de un sistema. Este cuestionario, desarrollado por John Brooke en 1986, consta de 10 ítems que los usuarios deben calificar en una escala de Likert de 7 puntos, que van desde "totalmente de acuerdo" hasta "totalmente en desacuerdo".

Ventajas del SUS

El SUS es conocido por su simplicidad y eficiencia. Es fácil de entender y de aplicar, lo que facilita su uso en diferentes contextos y permite obtener resultados rápidos y fácilmente interpretables. Además, su versatilidad permite aplicarlo a una amplia variedad de productos y servicios, incluyendo hardware, software, aplicaciones móviles y más(Brooke, 1996).

Proceso de Aplicación del SUS

- Selección de Participantes: Se seleccionará una muestra representativa de usuarios finales del sistema, asegurando una diversidad de perfiles y niveles de experiencia.

- Administración del Cuestionario: Los participantes utilizarán el sistema y luego completarán el cuestionario SUS.
- Análisis de Resultados: Los resultados se analizarán para calcular el puntaje SUS, que proporciona una medida cuantitativa de la usabilidad del sistema.

2.1.5.3. "Cuestionario de Experiencia de Usuario (UEQ)"

Este cuestionario es una herramienta de evaluación que mide diferentes aspectos de la experiencia del usuario con un producto o sistema. "El UEQ" consta de 26 ítems que se califican en una escala de 7 puntos", abarcando seis dimensiones principales: atracción, claridad, eficiencia, precisión, estimulación y novedad.

Ventajas del UEQ

El UEQ permite una evaluación integral de la experiencia del usuario al medir múltiples dimensiones, proporcionando una visión más completa. Su flexibilidad permite adaptarlo a diferentes contextos y tipos de productos, y su capacidad para comparar resultados con una base de datos de referencias facilita la interpretación de los resultados.

Proceso de Aplicación del UEQ:

Selección de Participantes: Se seleccionará una muestra representativa de usuarios finales del sistema, asegurando una diversidad de perfiles y niveles de experiencia.

Administración del Cuestionario: Los participantes utilizarán el sistema y luego completarán el cuestionario UEQ.

Análisis de Resultados: Los resultados se analizarán para obtener puntajes en cada una de las seis dimensiones, lo que permitirá identificar áreas de fortaleza y debilidad en la experiencia del usuario (User Experience Questionnaire (UEQ), 2024).

2.2. Trabajos Relacionados

En el mercado actual, existen diversas soluciones que abordan el desarrollo y soporte de software desde distintos enfoques. Cada una de estas herramientas se fundamenta en una estrategia específica, lo que puede dificultar la adaptación del flujo interno de la empresa a estas soluciones. La complejidad radica en que estas herramientas, a menudo, no pueden abarcar todos los procesos necesarios para satisfacer completamente las necesidades empresariales.

2.2.1. LogMeIn Rescue

LogMeIn Rescue ofrece una solución robusta para el soporte técnico remoto, permitiendo a los técnicos diagnosticar y resolver problemas en dispositivos de usuarios en tiempo real. Con funcionalidades avanzadas como la transferencia de archivos, chat en vivo, y la capacidad de manejar múltiples sesiones simultáneamente, LogMeIn Rescue mejora la eficiencia del soporte y minimiza el tiempo de inactividad del usuario (LogMeIn Rescue API User Guide – Overview of the LogMeIn Rescue API, s.d.).

2.2.2. Freshdesk

Herramienta de soporte técnico basada en la nube que facilita la gestión de tickets y la resolución de problemas de los clientes. Con características como la automatización de tareas repetitivas, un portal de autoservicio para usuarios y la integración con múltiples canales de comunicación, Freshdesk permite a los equipos de soporte operar de manera más eficiente y brindar una atención al cliente de alta calidad (Freshworks Developer Docs | Freshworks app ecosystem, 2024).

2.2.3. Jira

Herramienta líder en la gestión de proyectos y seguimiento de incidencias, especialmente popular en entornos ágiles. Ofrece una amplia gama de funcionalidades para planificar, rastrear y gestionar el desarrollo de software. Con su capacidad de personalización de flujos de trabajo y su integración con otras herramientas de desarrollo, Jira facilita la colaboración y la entrega continua de proyectos de software (Atlassian, 2024).

2.2.4. Trello

Herramienta de gestión de proyectos que utiliza tableros Kanban para ayudar a los equipos a visualizar y organizar tareas. Su interfaz intuitiva permite crear, asignar y seguir el progreso de tareas en tiempo real. Trello es especialmente útil para proyectos pequeños y equipos que buscan una solución simple pero efectiva para la gestión de tareas y proyectos(Trello Guides: Help Getting Started With Trello | Trello, 2024).

2.2.5. Asana

Plataforma que permite gestionar proyectos, diseñada para apoyar a los equipos a planificar, organizar y seguir el progreso de su trabajo. Ofrece diversas vistas como listas, tableros y calendarios, adaptándose a diferentes estilos de trabajo y necesidades de los proyectos. Asana mejora la colaboración y la transparencia, asegurando que todos los miembros del equipo estén alineados y enfocados en los objetivos comunes(Build an App with Asana, 2024).

2.2.6. Slack

Plataforma de comunicación y colaboración en tiempo real que centraliza la mensajería, el intercambio de archivos y la integración con otras herramientas de desarrollo. Facilita la coordinación entre equipos dispersos geográficamente, mejorando la comunicación y reduciendo el tiempo de respuesta. Con sus canales organizados por temas, Slack ayuda a mantener las conversaciones ordenadas y accesibles para todos los miembros del equipo.(Slack platform overview | Slack, 2024)

A pesar de las numerosas soluciones disponibles en el mercado que ofrecen diversas herramientas para la gestión de proyectos y la comunicación con clientes, todas implementan su propio flujo de trabajo, permitiendo poca flexibilidad en su adaptación. Esta rigidez dificulta

que las empresas puedan adecuarse completamente al flujo de trabajo de estas herramientas. Además, al ser soluciones especializadas, presentan costos muy elevados, lo que resulta prohibitivo para pequeñas empresas o proyectos en sus etapas iniciales.

3. Construcción de la herramienta "Sistema de Gestión de Incidentes y Administración de Servicios"

3.1. Insumos

Para asegurar una gestión estructurada y eficiente del proyecto, se realizó un análisis exhaustivo del modelo de negocio general de las empresas cuyo modelo de negocio tiene como objetivo desarrollar software, así como de sus flujos de trabajo típicos. Este análisis permitió centralizar y coordinar todos los procesos esenciales de desarrollo y soporte en un marco coherente. Los flujos de trabajo resultante se dividen en tres principales actores: Cliente, Supervisor y Desarrollo. Cada uno de estos actores desempeñan roles y responsabilidades específicas que se integran para garantizar la entrega de soluciones tecnológicas efectivas y de alta calidad.

3.1.1. Análisis del Modelo de Negocio y Flujo de Trabajo

El análisis del modelo de negocio de las compañías desarrolladoras de software mostró la necesidad de un enfoque centralizado que permita una coordinación efectiva entre los distintos departamentos y actores involucrados en el desarrollo y soporte de software. Estas empresas enfrentan desafíos significativos como la gestión de requisitos, la asignación de recursos, la comunicación entre equipos y la entrega de productos finales que cumplan con las expectativas del cliente.

Como resultado de este análisis, se identificaron y optimizaron dos flujos de trabajo especializados:

- Flujo de Desarrollo: Este flujo, representado en la Imagen 1, abarca cada una de las etapas del desarrollo de software, desde la solicitud inicial del cliente hasta la entrega y aprobación del producto final. Está diseñado para asegurar una revisión rigurosa y un control en cada fase del proceso, asegurando que el producto final satisfaga los requisitos del cliente y cumpla con los estándares de calidad establecidos.
- Flujo de Soporte: Este flujo, representado en la Imagen 2, se centra en la gestión de incidentes y problemas técnicos que puedan surgir después de la entrega del producto. Asegurar que los problemas sean identificados, analizados y resueltos de manera eficiente, manteniendo la calidad y funcionalidad del sistema en operación.

Ambos flujos de trabajo están estructurados para integrar roles y responsabilidades específicas de los actores involucrados (Cliente, Supervisor y Desarrollo), facilitando una coordinación efectiva y una gestión centralizada que aborda los desafíos comunes en la industria del desarrollo de software.

A continuación, se describirán en detalle estos dos flujos de trabajo, explicando cada una de sus etapas y la forma en que los actores interactúan dentro del sistema.

3.1.1.1. Desarrollo

Este flujo se inicia con una solicitud del cliente y finaliza con la aprobación y despliegue de la solución, abarcando todas las fases intermedias necesarias para garantizar la calidad y efectividad del producto final.

Descripción del Flujo de Trabajo para Desarrollo

Cliente:

- Solicita una solución tecnológica: El proceso se inicia cuando el cliente identifica una necesidad tecnológica y formaliza una solicitud. Esta etapa es crucial ya que define el punto de partida del proyecto y establece las expectativas iniciales.

  Supervisor:

- Revisa la solicitud y asigna al personal correspondiente: El supervisor revisa la solicitud del cliente para entender los requerimientos y determina el personal adecuado para atender la solicitud.
- Analiza el informe: Una vez asignado el personal, el supervisor analiza los informes generados durante el proceso de análisis de viabilidad y desarrollo.
- Notifica al cliente sobre la viabilidad: Si la solución es viable, el supervisor notifica al cliente y elabora una proforma que se envía para aprobación. Si no es viable, se comunica al cliente y se eliminan las solicitudes no viables.

Desarrollo:

- Analiza viabilidad: El equipo de desarrollo analiza la viabilidad técnica de la solución propuesta, asegurando que sea factible dentro de los recursos y capacidades disponibles.
- Realiza un informe detallado y lo envía a revisión: Se elabora un informe detallado sobre la solución propuesta y se envía al supervisor para su revisión.
- Desarrolla la solución: Una vez aprobada la viabilidad y la proforma, el equipo de desarrollo procede a la implementación de la solución.
- Proceso de pruebas: La solución desarrollada se somete a rigurosas pruebas para garantizar su calidad y funcionalidad.
- Modifica la solución: Si durante las pruebas se identifican problemas o áreas de mejora, se realizan las modificaciones necesarias.
- Despliega la solución: Finalmente, la solución se despliega en el entorno de producción, completando así el ciclo de desarrollo.

A continuación, se presenta el flujo de trabajo completo para atender una solicitud de desarrollo de software:

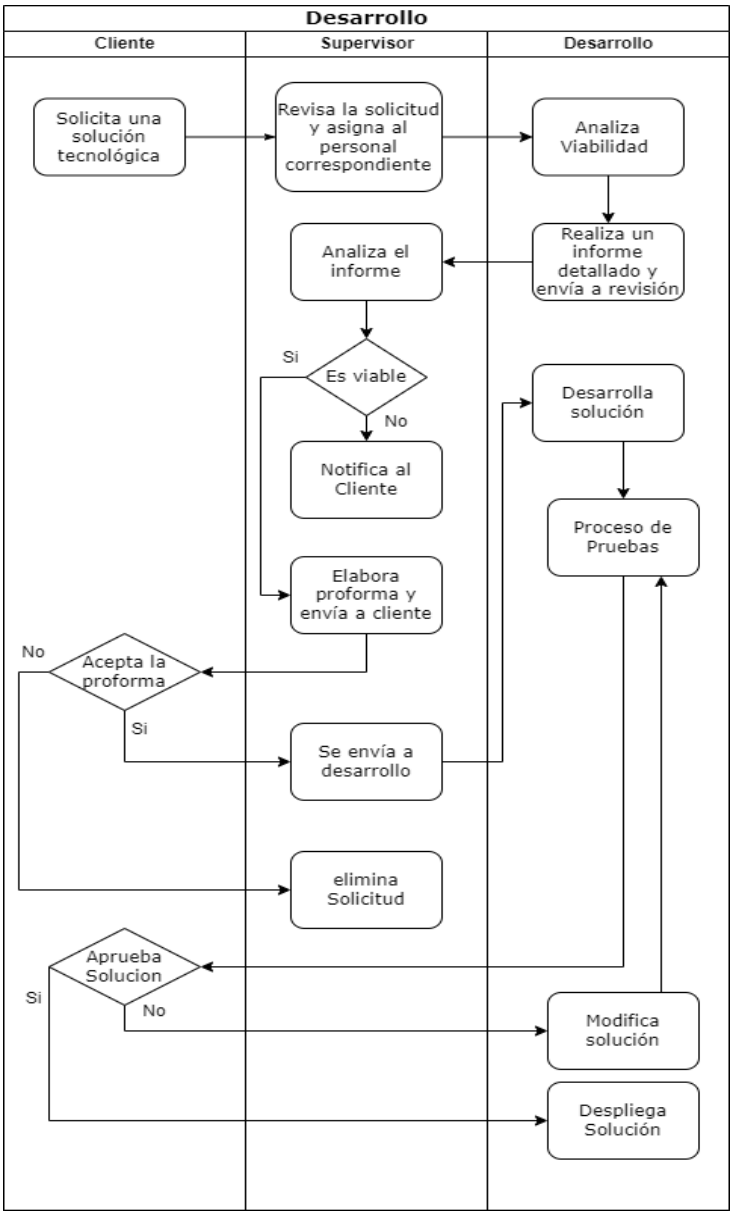


Ilustración 1. Flujo de trabajo completo para atender una solicitud de desarrollo de software

3.1.1.2. Soporte

Este flujo se inicia con una solicitud de soporte del cliente y finaliza con la actualización de la solución, abarcando todas las fases intermedias necesarias para garantizar la calidad y efectividad del producto final.

Descripción del Flujo de Trabajo para Soporte

El flujo de trabajo de soporte se divide en dos principales actores: Cliente y Desarrollador. A continuación, se describe cada etapa del proceso:

Cliente:

- Solicita un soporte: El proceso comienza cuando el cliente detecta un problema o necesita asistencia técnica y formaliza una solicitud de soporte.

Desarrollador:

- Analiza la problemática: El desarrollador recibe la solicitud de soporte y analiza la problemática para comprender la naturaleza del problema.
- ¿Se necesita más información?: Si el problema no está claro o se necesita más información, el desarrollador solicita una aclaración al cliente.
- Solicita aclaración: El cliente proporciona más información o detalles necesarios para entender completamente la problemática.
- Realiza modificación en el sistema correspondiente: Con la información completa, el desarrollador realiza las modificaciones necesarias en el sistema para solucionar el problema.
- Valida el funcionamiento: Una vez realizada la modificación, el desarrollador valida que el sistema funcione correctamente tras el cambio.
- Solicita comprobación: El desarrollador solicita al cliente que valide la solución implementada para asegurarse de que el problema ha sido resuelto satisfactoriamente.
- Válida solución: El cliente comprueba si la solución es satisfactoria. Si no lo es, proporciona retroalimentación adicional y el ciclo se repite.
- Se actualiza el servicio: Si la solución es validada por el cliente, el servicio se actualiza y se considera el problema resuelto.

A continuación, se presenta el flujo de trabajo completo para atender un soporte sobre un producto solicitado por un cliente:

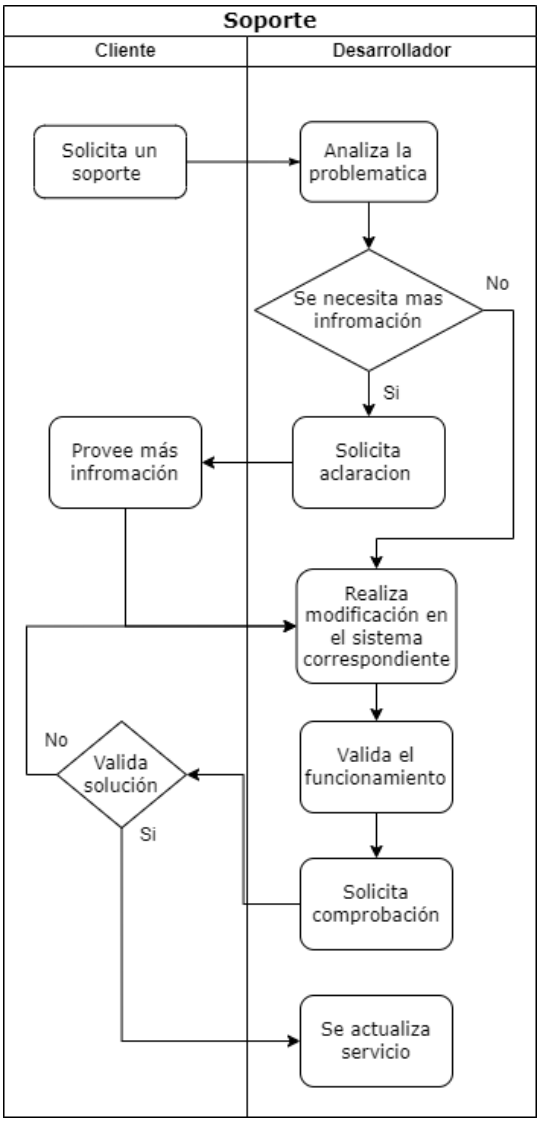


Ilustración 2 Flujo de trabajo completo para atender un soporte sobre un producto solicitado por un cliente.

3.1.2. Modelo de dominio

Durante la etapa de levantamiento de requisitos del sistema, se identificaron las principales entidades y conceptos que representan su funcionalidad. Esto permitió comprender gráficamente la interacción de las diferentes partes del sistema.

Para una mejor organización, el modelo de dominio se ha dividido en dos partes: una enfocada en la gestión de requerimientos y otra en la administración del soporte técnico, ya que estas áreas tienen necesidades distintas.

3.1.2.1. Modelo de dominio - Gestión de Requerimientos

Un requerimiento es una entidad que involucra varios elementos como productos, clientes, proformas, supervisores, comentarios, técnicos, prioridades, tareas, documentación, estados, entre otros.

Relación de entidades:

Productos y Clientes:

- Un requerimiento puede estar relacionado con un producto específico.
- Un cliente puede tener varios productos y puede subir requerimientos para cada uno de ellos.

Supervisores y Proformas:

- El supervisor de la empresa está asignado a cada requerimiento.
- El supervisor también elabora y envía la proforma al cliente.

Técnicos:

- Los requerimientos tienen técnicos asignados, quienes son responsables de realizar lo solicitado en los requerimientos.

Prioridades:

Los requerimientos se clasifican por prioridades (alta, media, baja) según la urgencia indicada por el cliente.

Estados del Requerimiento:

- Solicitud: Estado inicial cuando el cliente solicita un requerimiento.
- Análisis: Fase en la que el supervisor asigna técnicos para revisar la solicitud e indicar lo que implica el desarrollo.
- Revisado: Estado en el que los técnicos han realizado el análisis y el supervisor genera y envía la proforma al cliente.
- Proceso de Aprobación: Estado en el que el supervisor ha enviado la proforma, pero aún no hay respuesta del cliente.
- Aprobado: Estado cuando el cliente acepta la proforma.
- Desarrollo: Fase en la que el requerimiento ha sido aprobado, el supervisor establece el estado en desarrollo, y los desarrolladores comienzan a trabajar en lo solicitado.
- Pruebas: Fase en la que el desarrollo ha finalizado y se ha cargado la documentación requerida.
- Producción: Fase en la que se ha cargado la documentación de pruebas.
- Facturado: Proceso realizado por el contador para emitir la factura.
- Pagado: Estado final cuando el cliente ha pagado la factura.

A continuación, se presenta el modelo de dominio de gestión de requerimientos, donde se describe cada estado, dependencia y herencia necesaria para su futuro funcionamiento en el sistema:

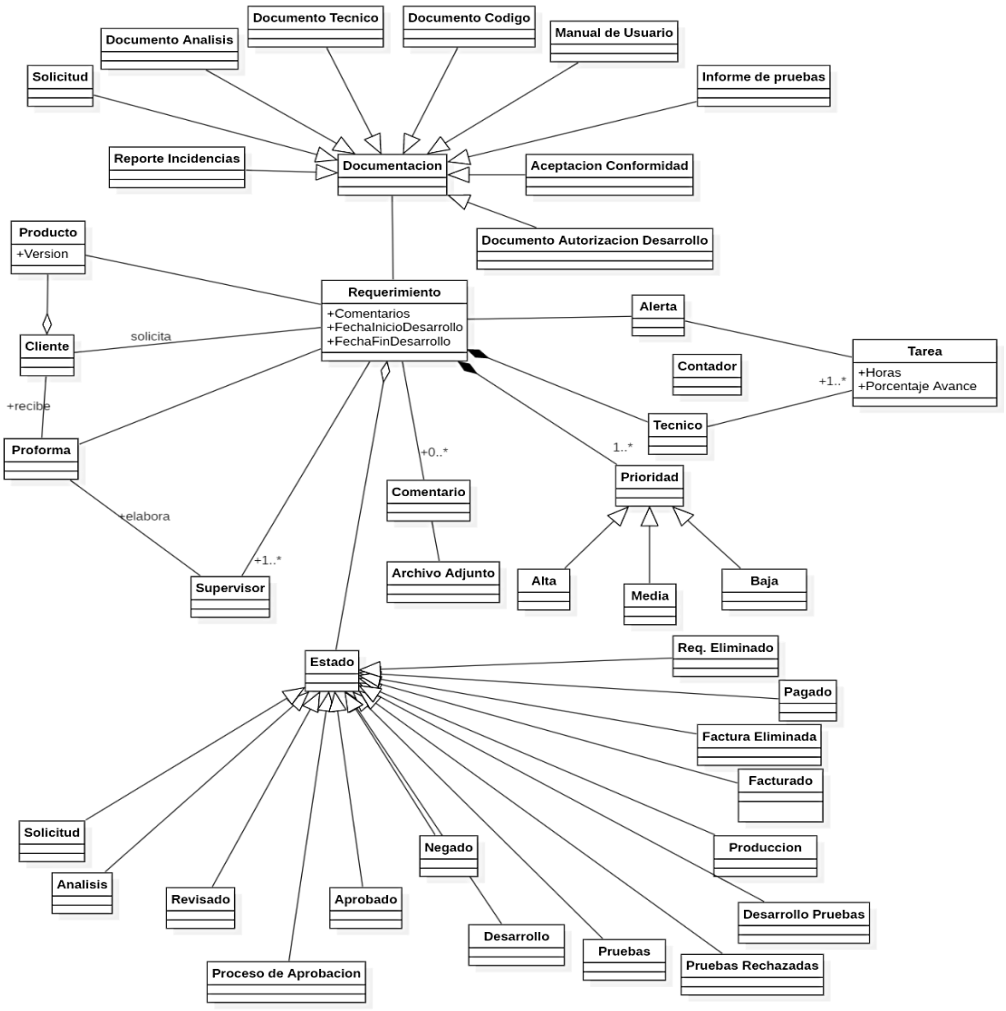


Ilustración 3 Modelo de dominio de gestión de requerimientos, describe cada estado, dependencia y herencia necesaria para su futuro funcionamiento en el sistema

3.1.2.2. Modelo de dominio - Gestión de Soporte Técnico

En la gestión de soporte técnico, un ticket es una entidad central que se relaciona tanto con un producto como con un cliente. Esta entidad permite la gestión y seguimiento de incidencias reportadas por los clientes sobre los productos.

Cada ticket tiene un conjunto de atributos y estados que facilitan su seguimiento y resolución:

Relación de entidades:

Productos y Clientes:

- Un ticket está asociado a un producto específico.
- Un cliente puede tener varios productos y puede generar tickets para cada uno de ellos.

Técnicos:

- Los tickets tienen técnicos asignados, quienes son responsables de resolver las incidencias reportadas.

Prioridades:

- Los tickets se clasifican por prioridades: baja, media, alta y crítica, según la urgencia indicada por el cliente.

Mensajes:

- Dentro de un ticket, tanto el cliente como el técnico asignado pueden enviar mensajes. Esta funcionalidad facilita la comunicación directa y continua para la resolución del problema.

Estados del Requerimiento:

- Abierto: Estado inicial cuando el cliente reporta una incidencia.
- Cerrado Empresa: Estado cuando el técnico ha trabajado en el ticket y considera que el problema ha sido resuelto.
- Cerrado Cliente: Estado final cuando el cliente confirma que la solución implementada es satisfactoria.

A continuación, se presenta el modelo de dominio de gestión de ticket de soporte, describe cada estado, dependencia y herencia necesaria para su futuro funcionamiento en el sistema:

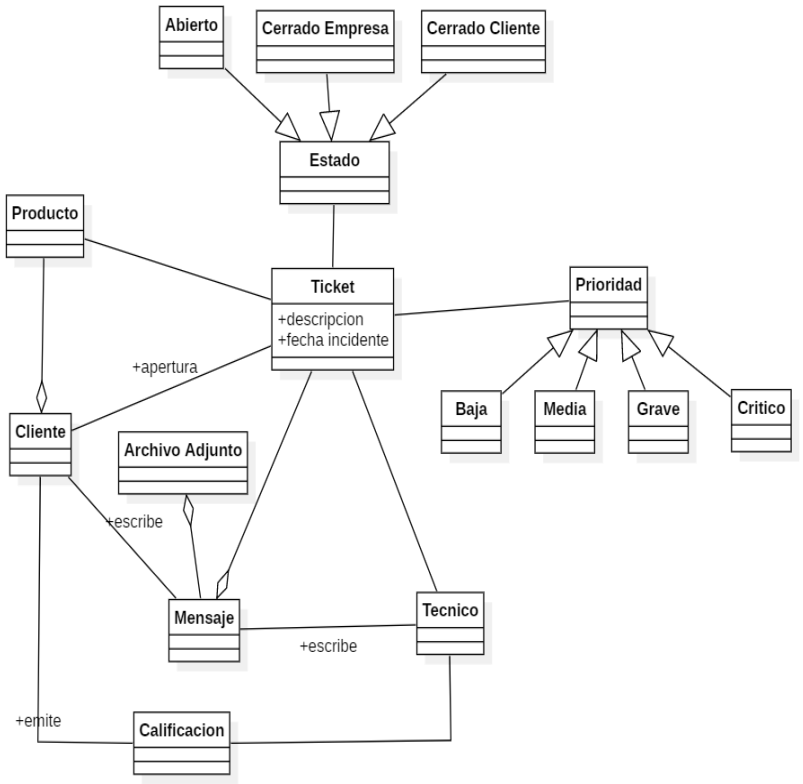


Ilustración 4 Modelo de dominio de gestión de ticket de soporte, describe cada estado, dependencia y herencia necesaria para su futuro funcionamiento en el sistema

Una vez que se identificó y refinó el flujo de trabajo, se procedió a la aplicación del RUP comenzando con la identificación de los actores involucrados en el sistema.

3.2. Requisitos

Después de haber generado todos los insumos descritos con anterioridad, incluyendo los flujos de trabajo para la gestión de requerimientos y la gestión de tickets de soporte, obtuvimos una comprensión mucho más clara de los requisitos del sistema. Estos insumos nos permitieron identificar y definir con precisión los actores involucrados en cada proceso.

3.2.1. Actores

El sistema incluye los siguientes actores:

- Usuario: Persona que utiliza el sistema para reportar y gestionar incidentes. Este rol puede incluir empleados internos y clientes externos.
- Cliente: Entidad que solicita una solución tecnológica y puede interactuar con el sistema para realizar seguimiento de sus solicitudes y ver el estado de los incidentes reportados.
- Supervisor: Monitorea el desempeño del equipo de desarrollo y asegura que las solicitudes y los incidentes sean gestionados de manera eficiente. Este actor también asigna tareas y revisa los informes generados.
- Técnico: Encargado de implementar las soluciones tecnológicas solicitadas, corregir incidentes reportados y realizar pruebas necesarias para asegurar la calidad del software. Hace alusión al actor Desarrollador identificado en la sección anterior de análisis y adaptación de flujo de trabajo.
- Contador: Aunque toda el área de facturación y cobros se maneja externamente al sistema, el Contador utiliza el sistema para gestionar los estados relacionados con la facturación y los pagos. Este rol incluye generar y enviar facturas a los clientes (gestionado fuera del sistema), realizar el seguimiento de los pagos y actualizar los estados en el sistema. El Contador puede registrar si ya se ha generado una factura y cambiar el estado a "pagado" una vez que el cliente cancela la factura por un desarrollo realizado.
- Tiempo: Representa la variable temporal, crucial para la gestión de plazos y tiempos de respuesta dentro del sistema. Aunque no es un actor humano, es fundamental para medir la eficiencia y el cumplimiento de los tiempos establecidos.

3.2.2. Descripción de Roles y Responsabilidades

- Cliente: Solicita nuevas soluciones, aprueba proformas. También puede realizar un seguimiento del estado de sus solicitudes.
- Supervisor: Asigna tareas a los desarrolladores, revisa los informes de viabilidad y de desarrollo, y asegura que se cumplan los plazos establecidos.

- Desarrollador: Analiza la viabilidad técnica de las solicitudes, desarrolla soluciones tecnológicas, realiza pruebas y despliega las soluciones aprobadas.
- Contador:
  - Fuera del Sistema:
    - Maneja toda la facturación y cobros de manera externa, incluyendo la generación de facturas y el seguimiento de los pagos.
  - Dentro del Sistema:
    - Actualiza el estado de las solicitudes y desarrollos, indicando si ya se ha generado una factura.
    - Marca los estados de las facturas como "pagadas" una vez que el cliente ha realizado el pago.
    - Utiliza el sistema para tener un registro actualizado del estado de los pagos y facilitar la gestión administrativa.
- Tiempo: Utilizado para medir y gestionar los tiempos de respuesta y los plazos de entrega de las soluciones tecnológicas.

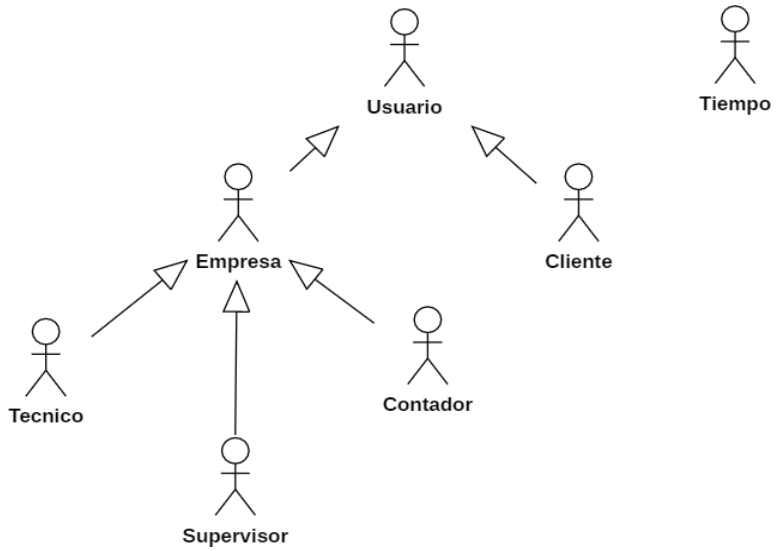


Ilustración 5 Esquema de la organización de los actores y sus herencias

3.2.3. Casos de Uso

En esta sección se muestran los diagramas de caso de uso que describen las principales interacciones de los actores con el Sistema de Gestión de Incidentes y Administración de Servicios. Estos diagramas proporcionan una perspectiva precisa y detallada de cómo se gestionan los diferentes aspectos del sistema, específicamente la gestión de requerimientos y el soporte de tickets.

Los diagramas de caso de uso ayudan a ilustrar las funciones y responsabilidades de los actores involucrados, facilitando una comprensión integral de los procesos y flujos de trabajo

dentro del sistema. A continuación, se detallan los diagramas para cada área clave del sistema, proporcionando una estructura organizada para la gestión eficiente de requerimientos y soporte técnico.

3.2.3.1. Diagrama de Caso de Uso: Gestión de Requerimientos

El diagrama de caso de uso de gestión de requerimientos se centra en las interacciones relacionadas con la solicitud y gestión de desarrollos tecnológicos.

Descripción:

Actores: Usuario, Cliente, Empresa, Técnico, Contador, Supervisor, Tiempo

Casos de Uso Principales:

- Ingresar Solicitud: El usuario puede ingresar una nueva solicitud de requerimiento.
- Buscar Requerimiento: El usuario y el cliente pueden buscar requerimientos específicos en el sistema.
- Gestionar Requerimiento: El usuario gestiona los requerimientos asignados, incluyendo seguimiento y actualización del estado.
- Enviar Comentario con Cliente: Comunicación entre el usuario y el cliente para aclarar detalles del requerimiento.
- Solicitar Revisión: El cliente solicita una revisión del requerimiento.
- Buscar Técnico: La empresa busca técnicos disponibles para asignar al requerimiento.
- Asignar Técnicos: La empresa asigna técnicos a los requerimientos.
- Planificar: La empresa planifica las actividades necesarias para cumplir con el requerimiento.
- Adjuntar Documentación: Los usuarios adjuntan documentación relevante al requerimiento.
- Ver Documentación: Los usuarios visualizan la documentación adjunta.
- Enviar Comentario Interno: Los usuarios envían comentarios internos relacionados con el requerimiento.
- Ver Proforma: El supervisor y el cliente pueden ver las proformas generadas.
- Enviar Proforma: El supervisor envía la proforma al cliente.
- Notificar Cambio: El sistema notifica a los actores sobre cualquier cambio relevante en el estado del requerimiento.

Este diagrama detalla las interacciones complejas y colaborativas que son esenciales para la gestión eficaz de requerimientos dentro del sistema.

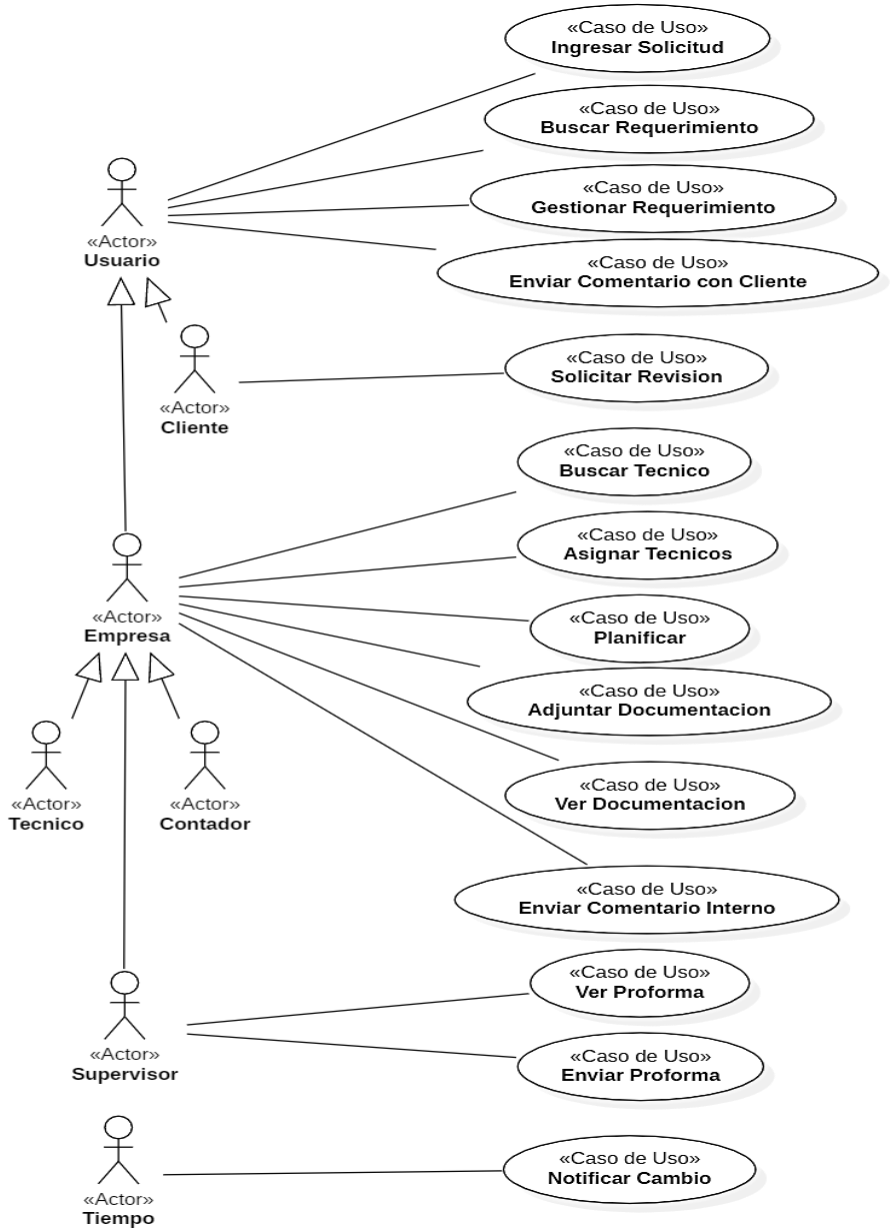


Ilustración 6 Diagrama de caso de uso sobre la gestión de un requerimiento, describen las maneras en las que cada actor interactúa, ya sea con el sistema o con otro actor

3.2.3.2. Diagrama de Caso de Uso: Soporte de Tickets

El diagrama de caso de uso de soporte se enfoca en la gestión de tickets de soporte, permitiendo a los usuarios y clientes interactuar con el sistema para resolver problemas y recibir asistencia.

Descripción:

Actores: Usuario, Cliente

Casos de Uso Principales:

- Buscar Ticket de Soporte: El usuario puede buscar tickets de soporte específicos para revisarlos.

- Gestionar Ticket de Soporte: El usuario gestiona los tickets de soporte, incluyendo la actualización del estado y la resolución de problemas.
- Aperturar Ticket de Soporte: El cliente abre un nuevo ticket de soporte cuando necesita asistencia técnica.
- Calificar Servicio: El cliente califica el servicio recibido una vez que el ticket de soporte ha sido resuelto.

Este diagrama muestra cómo el sistema facilita la interacción entre los clientes y el equipo de soporte para asegurar una resolución eficiente y satisfactoria de los problemas reportados.

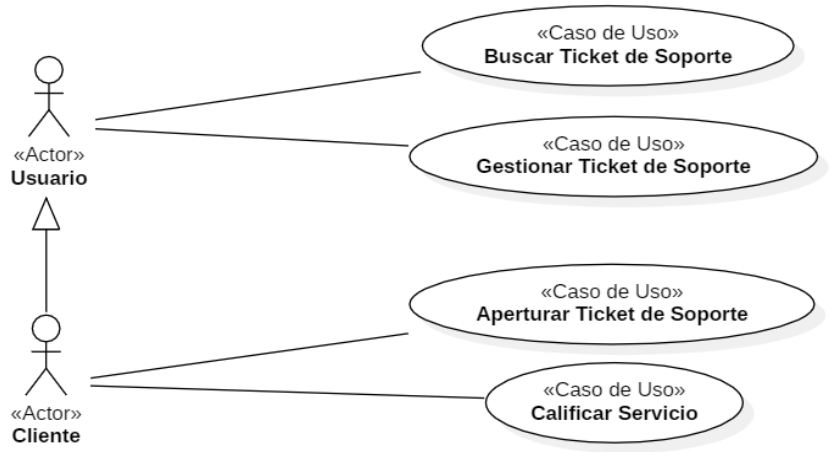


Ilustración 7 Diagrama de caso de uso sobre la gestión de un Ticket de Soporte, describen las maneras en las que cada actor interactúa, ya sea con el sistema o con otro actor

3.2.3.3. Diagrama de Caso de Uso General

El diagrama de caso de uso general describe los casos de uso básicos que todos los usuarios del sistema deben realizar para acceder y utilizar las funcionalidades del sistema.

Descripción:

Actor: Usuario

Casos de Uso:

- Iniciar Sesión: El usuario ingresa sus credenciales para acceder al sistema.
- Cerrar Sesión: El usuario cierra su sesión para salir del sistema de manera segura.
- Recuperar Clave: El usuario recupera su clave en caso de olvido, siguiendo un proceso de recuperación segura.

Este diagrama asegura que todas las interacciones básicas con el sistema sean gestionadas de manera eficiente, permitiendo un acceso seguro y controlado a las funcionalidades del sistema.

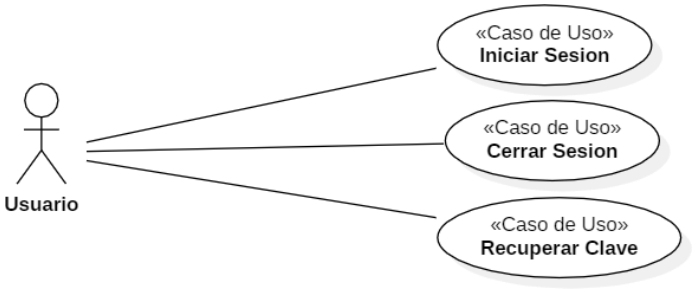


Ilustración 8 Diagrama de caso de uso del Usuario de sistema que permite ingresar, cerrar sesión o recuperar su clave

3.2.4. Priorización de casos de uso

El objetivo de priorizar estos casos de uso en la primera iteración es garantizar que los aspectos más críticos del sistema sean desarrollados y evaluados primero. Esto permite identificar y mitigar riesgos tempranamente, asegurar que los procesos fundamentales funcionen correctamente y proporcionar un valor inmediato a los usuarios finales. Al enfocar los esfuerzos en la gestión de requerimientos y el soporte de tickets, se busca:

Los criterios utilizados para priorizar son los siguientes

- Impacto en el Usuario Final: Priorizar aquellas funcionalidades que tienen el mayor impacto positivo en los usuarios finales, garantizando que sus necesidades más críticas sean atendidas primero.
- Riesgo: Identificar y desarrollar funcionalidades que presenten mayores riesgos técnicos o de negocio, permitiendo la mitigación temprana de estos riesgos y la validación de soluciones.
- Complejidad Técnica: Abordar inicialmente las funcionalidades de mayor complejidad técnica para evitar retrasos en fases posteriores del desarrollo debido a problemas imprevistos.
- Dependencias: Priorizar funcionalidades que son prerrequisitos para el desarrollo de otras partes del sistema, asegurando un flujo de trabajo más eficiente y sin interrupciones.
- Frecuencia de Uso: Enfocarse en funcionalidades que serán utilizadas con mayor frecuencia por los usuarios, mejorando su experiencia y satisfacción con el sistema desde el principio.
- Valor de Negocio: Priorizar aquellas funcionalidades que aportan mayor valor al negocio, ya sea a través de la generación de ingresos, ahorro de costos, o mejora de la eficiencia operativa.

- Feedback de Usuarios: Tener en cuenta la retroalimentación de usuarios y clientes potenciales para identificar las funcionalidades más solicitadas o problemáticas, asegurando que el sistema atienda sus necesidades más urgentes.

3.2.5. Casos de Uso Prioritarios

Los casos de uso priorizados incluyen:

Gestión de Requerimientos:

- Buscar Requerimientos
- Asignar Requerimientos: Facilita la asignación de tareas a los desarrolladores adecuados.
- Seguimiento de Requerimientos: Proporciona a los supervisores y clientes una herramienta para monitorear el estado y progreso de los requerimientos.

Soporte de Tickets:

- Apertura de Ticket de Soporte: Permite a los usuarios reportar incidentes y problemas técnicos.
- Asignación de Tickets: Coordina la asignación de tickets a los técnicos de soporte adecuados.
- Resolución de Tickets: Gestiona el proceso de resolución de problemas, asegurando que los incidentes sean abordados de manera eficiente y efectiva.

3.2.6. Especificación de Casos de Uso

En esta sección se presentan las especificaciones de los casos de uso presentados en la sección anterior. Cada caso de uso se describe en términos de sus actores, precondiciones y postcondiciones y flujos principales.

Especificación: Buscar Requerimiento

Descripción:

- Actor: Usuario
- Propósito: Permitir al usuario buscar requerimientos específicos en el sistema.
- Precondiciones: El usuario debe estar autenticado.

Postcondiciones:

El usuario visualiza los requerimientos encontrados y puede seleccionar uno para ver detalles adicionales.

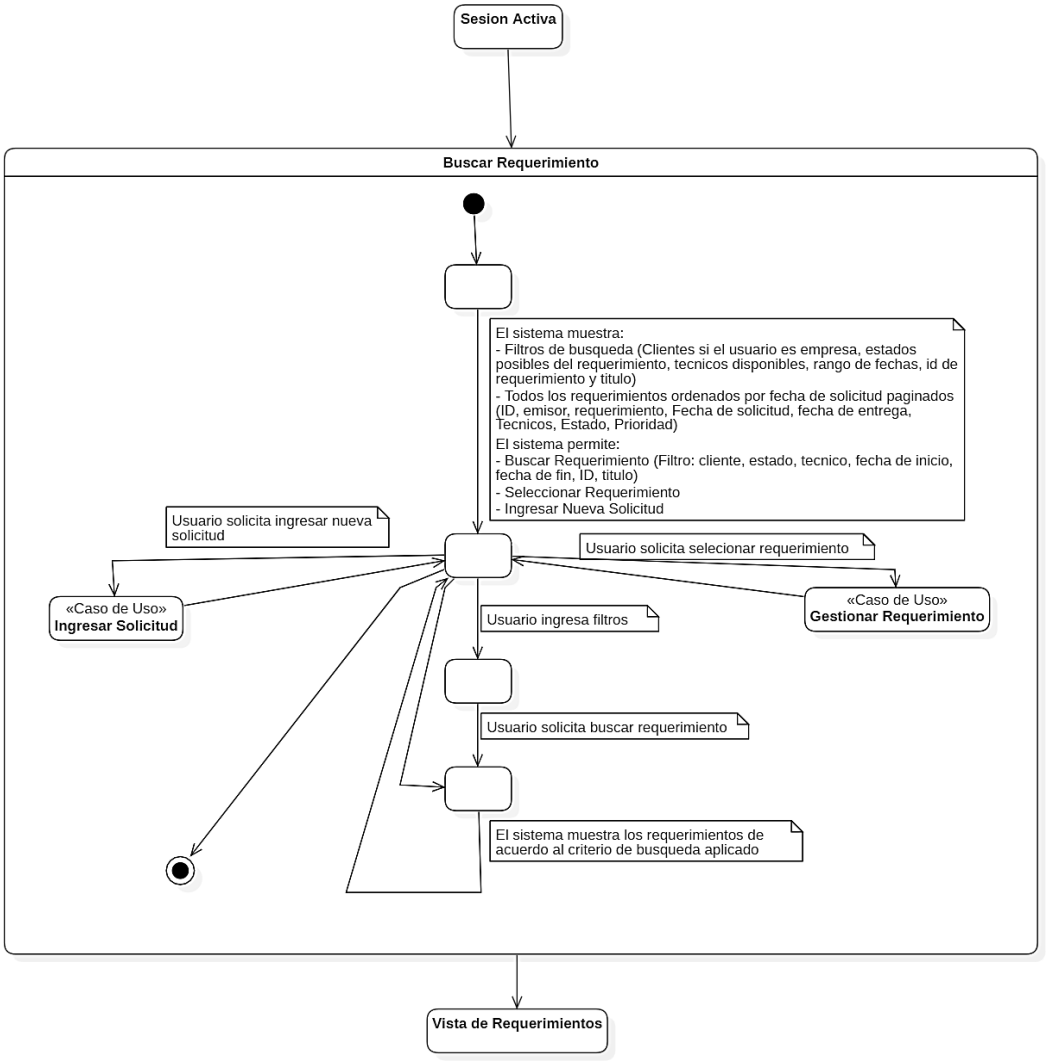


Ilustración 9 Especificación de caso de uso "Buscar Requerimiento" que permite al usuario encontrar todos los requerimientos bajo un criterio específico

Especificación: Apertura de Ticket de Soporte

Descripción:

- Actor: Cliente
- Propósito: Permitir al cliente abrir un nuevo ticket de soporte.
- Precondiciones: El cliente debe estar autenticado.

  Postcondiciones:

Un nuevo ticket de soporte es creado y registrado en el sistema.

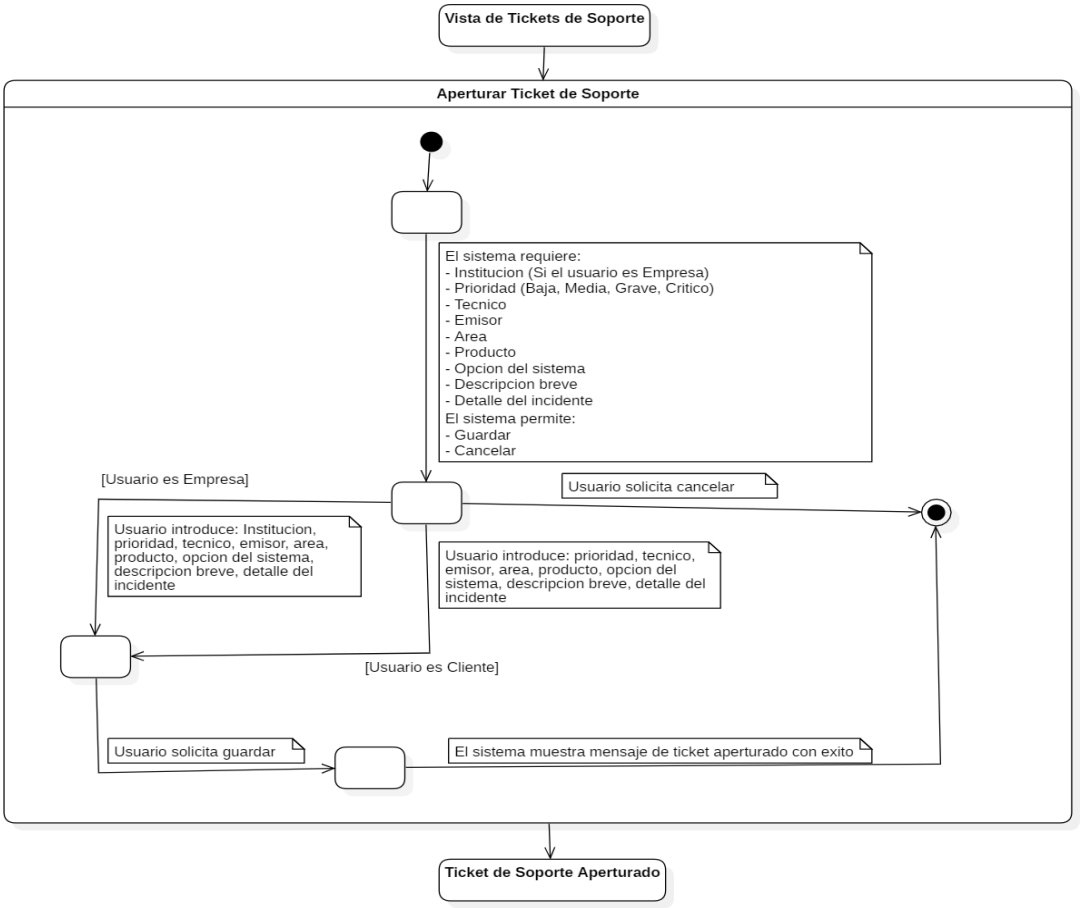


Ilustración 10 Especificación de caso de uso "Apertura Ticket de Soporte" que permite al usuario crear un ticket de soporte sobre una herramienta.

Especificación: Gestionar Requerimiento

Descripción:

- Actor: Usuario
- Propósito:
- Permitir al usuario gestionar el estado y la información de un requerimiento.

Precondiciones:

- El usuario debe estar autenticado y tener permisos para gestionar requerimientos.

Postcondiciones:

- Los cambios realizados en el requerimiento son guardados y reflejados en el sistema.

Acciones Alternativas:

- Agregar Responsable: El usuario puede solicitar agregar un responsable al requerimiento. El sistema muestra una lista de técnicos disponibles y permite al usuario seleccionar uno.

- Agregar Tarea: El usuario puede solicitar agregar una tarea al requerimiento. El sistema muestra una lista de tareas posibles y permite al usuario seleccionar una o más.
- Ver Documentación: El usuario puede solicitar ver la documentación adjunta al requerimiento. El sistema muestra la documentación disponible.
- Ver Proforma: Si el usuario es cliente, puede solicitar ver la proforma asociada al requerimiento.
- Enviar Comentario Interno: El usuario puede enviar un comentario interno relacionado con el requerimiento.
- Enviar Comentario con Cliente: El usuario puede enviar un comentario al cliente relacionado con el requerimiento.
- Solicitar Revisión: Si el usuario es cliente, puede solicitar una revisión del requerimiento.
- Guardar Cambios: El usuario guarda los cambios realizados en el requerimiento. El sistema valida los datos y guarda los cambios si son correctos.
- Cancelar: El usuario puede cancelar la operación en cualquier momento, y los cambios no serán guardados.

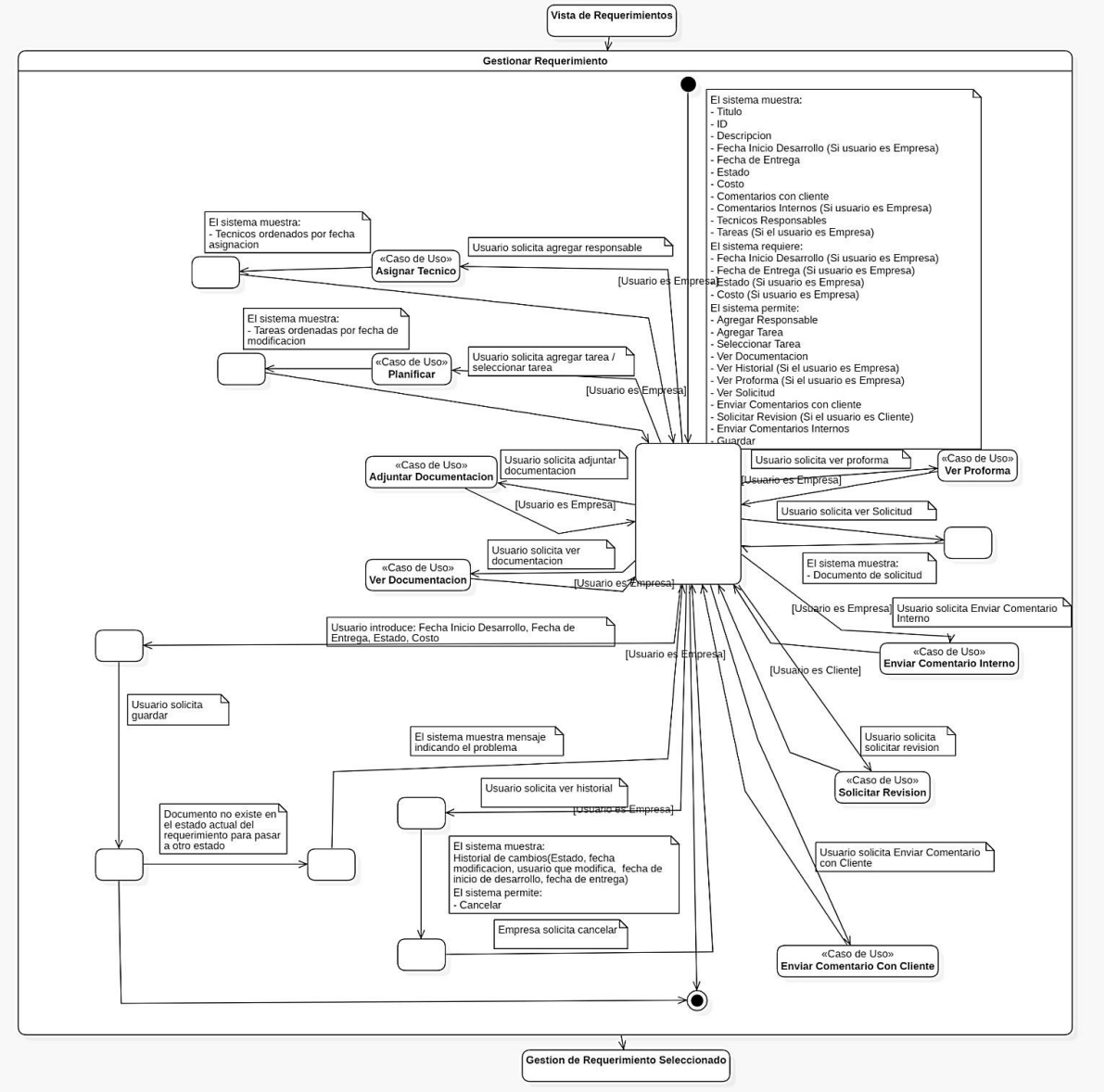


Ilustración 11 Especificación de caso de uso "Gestionar Requerimiento" que especifica todo el control sobre los requerimientos, los estados y procesos por los cuales este pasa

Los siguientes casos de uso adicionales se incluyen en los anexos para proporcionar una referencia completa del sistema. Aunque no se describen en detalle en el cuerpo principal de la tesis, estos casos de uso son fundamentales para una comprensión exhaustiva del sistema y sus funcionalidades:

- Cerrar Sesión
- Recuperar Clave
- Ingresar Solicitud
- Asignar Técnicos

- Buscar Técnico
- Planificar
- Ver Proforma
- Enviar Proforma
- Enviar Comentario Interno
- Enviar Comentario con Cliente
- Ver Documentación
- Adjuntar Documentación
- Solicitar Revisión

3.2.7. No Funcionales

Los siguientes son los requerimientos no funcionales identificados para el sistema de gestión de incidentes y administración de servicios:

Rendimiento:

- El sistema debe ser capaz de procesar hasta 50 solicitudes de soporte por día sin degradación del rendimiento.
- El sistema debe responder a cualquier operación en no más de 4 segundos bajo condiciones de carga normal.

Usabilidad:

- La interfaz de usuario debe ser intuitiva y sencilla de utilizar, permitiendo que los nuevos usuarios dominen el sistema en menos de 2 horas de formación.
- Se deben proporcionar mensajes de error claros y útiles para ayudar a los usuarios a resolver problemas comunes sin asistencia técnica.

Confiabilidad:

- El sistema debe tener una disponibilidad de 99.9% anual, asegurando que esté operativo casi todo el tiempo.

Seguridad:

- El acceso al sistema debe estar protegido mediante autenticación basada en roles, asegurando que solo los usuarios autorizados puedan realizar operaciones específicas.
- Todos los datos sensibles deben estar cifrados tanto durante su transmisión como cuando están almacenados para protegerlos de accesos no autorizados.

Escalabilidad:

- El sistema debe ser escalable, permitiendo la adición de nuevos usuarios y la gestión de un mayor número de solicitudes sin necesidad de reestructuración significativa.

Mantenibilidad:

- El sistema debe tener un diseño modular que permita la actualización y el mantenimiento de componentes individuales sin afectar al conjunto del sistema.
- Debe proporcionar documentación clara y completa para desarrolladores y administradores del sistema.

Compatibilidad:

- El sistema debe funcionar con los navegadores web más populares como: Chrome, Firefox, Safari, Edge.

3.2.8. Prototipos de interfaz de usuario

En esta sección se presentan los prototipos de interfaz de usuario desarrollados para el sistema. Estos prototipos son representaciones visuales que muestran la disposición y los elementos de las pantallas del sistema, proporcionando una visión clara de cómo el usuario va a interactuar con la aplicación. Los prototipos se utilizan para validar la experiencia del usuario y asegurar que el diseño cumple con los requisitos funcionales y de usabilidad. Cada prototipo se corresponde con las especificaciones de los casos de uso descritos en la sección 4.2.5, ilustrando cómo se implementan en la interfaz de usuario.

3.2.8.1. Prototipo de interfaz de usuario: Buscar Requerimiento

La pantalla de búsqueda de requerimientos permite al usuario encontrar requerimientos aplicando varios criterios de búsqueda, como cliente (si el usuario es un empleado de la empresa), estado del requerimiento, técnico asignado y rango de fechas de solicitud. En esta misma pantalla, el usuario puede acceder a la ventana para crear una nueva solicitud de requerimiento, así como a la ventana de gestión de requerimientos para administrar un requerimiento existente al seleccionarlo en el panel de resultados.

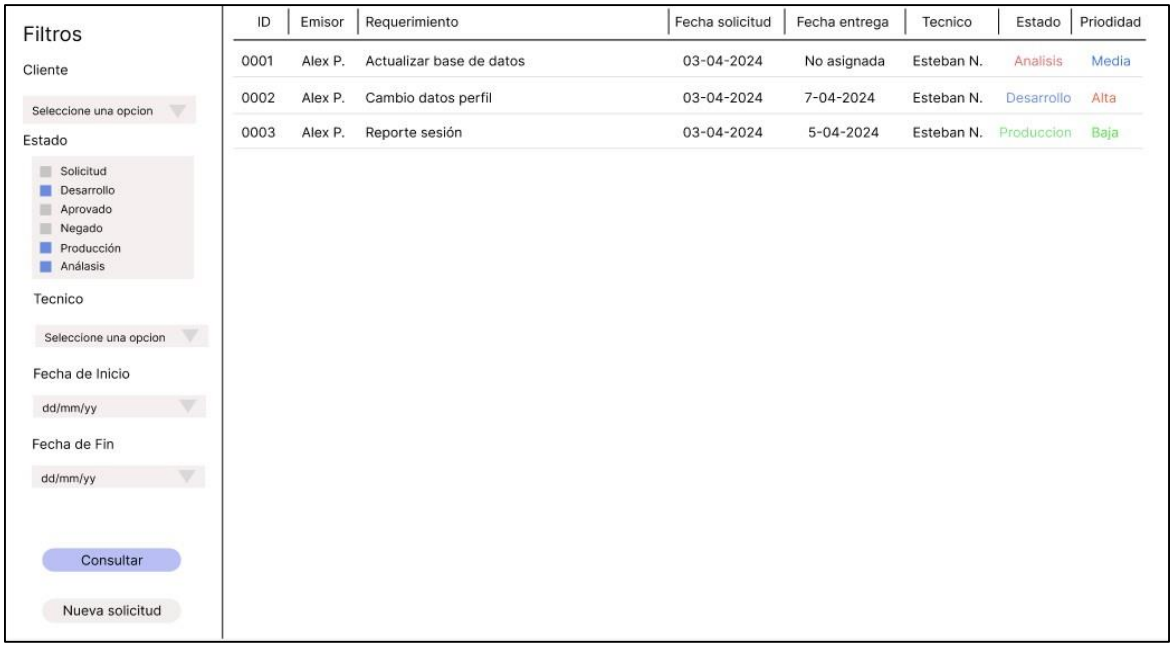


Ilustración 12 Prototipo de interfaz para la navegación y búsqueda de requerimientos

3.2.8.2. Prototipo de interfaz de usuario: Gestionar Requerimiento

La pantalla de gestión de requerimientos permite al usuario visualizar información detallada del requerimiento seleccionado desde la interfaz de búsqueda. Esta información incluye el título, código, descripción, fechas, estado del requerimiento y costo asociado.

Dentro de esta pantalla, los usuarios tipo empresa pueden gestionar los distintos estados del requerimiento, además de poder agregar técnicos, asignar tareas, adjuntar documentación, enviar comentarios y acceder al historial de cambios en los estados del requerimiento. También pueden gestionar proformas y documentos adjuntos asociados a la solicitud de requerimiento, si fueron cargados previamente.

Es importante destacar que cuando un cliente accede a esta opción, solo tiene permisos de lectura y puede visualizar información limitada como el título, código, descripción, fechas y estado actual del requerimiento. Además, puede enviar y leer comentarios intercambiados entre la empresa y él mismo.

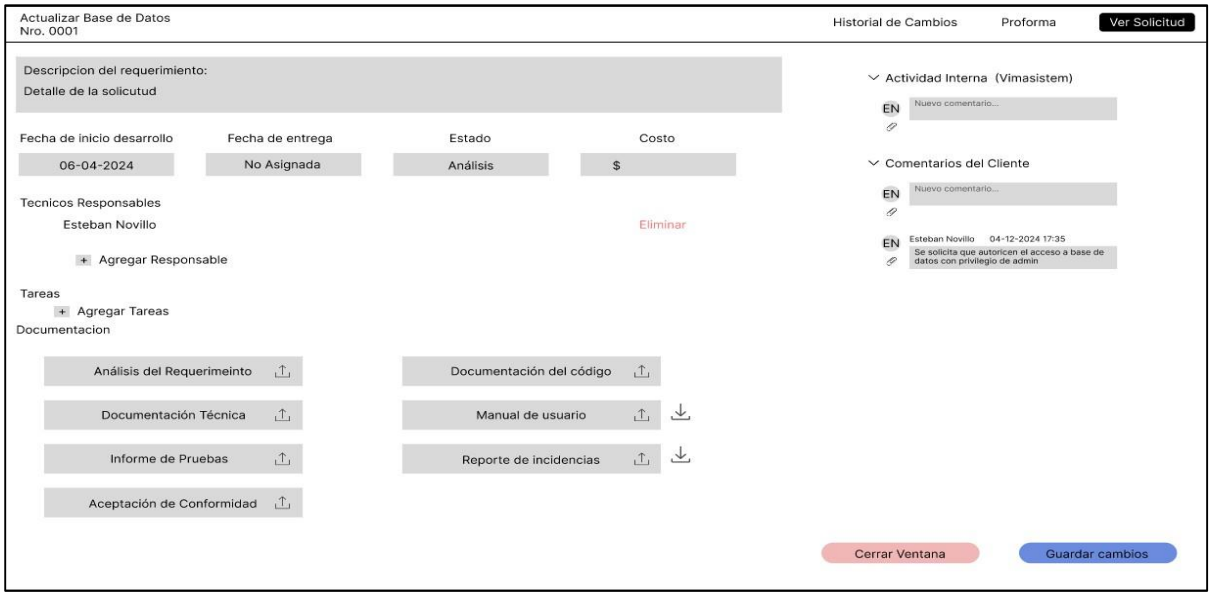


Ilustración 13 Prototipo de interfaz de gestionar requerimiento, permite que el usuario manipule los estados, comentarios y documentación sobre un requerimiento específico

3.2.8.3. Prototipo de interfaz de usuario: Aperturar Ticket de Soporte

En esta pantalla, el sistema permite abrir un ticket de soporte solicitando información como: empresa o institución desde la cual se cargará el ticket, la prioridad (alta, media, baja o crítica), el técnico que podría atender la petición, el emisor (usuario actualmente logueado en el sistema), el área (descripción del departamento al que pertenece el usuario cliente), producto (listado de productos disponibles para el cliente), la opción del sistema donde se presenta el inconveniente, una descripción breve y detalles adicionales.

Es importante destacar que cuando el usuario es un cliente, la pantalla no le solicita seleccionar la empresa, ya que esta información se obtiene automáticamente del usuario logueado.

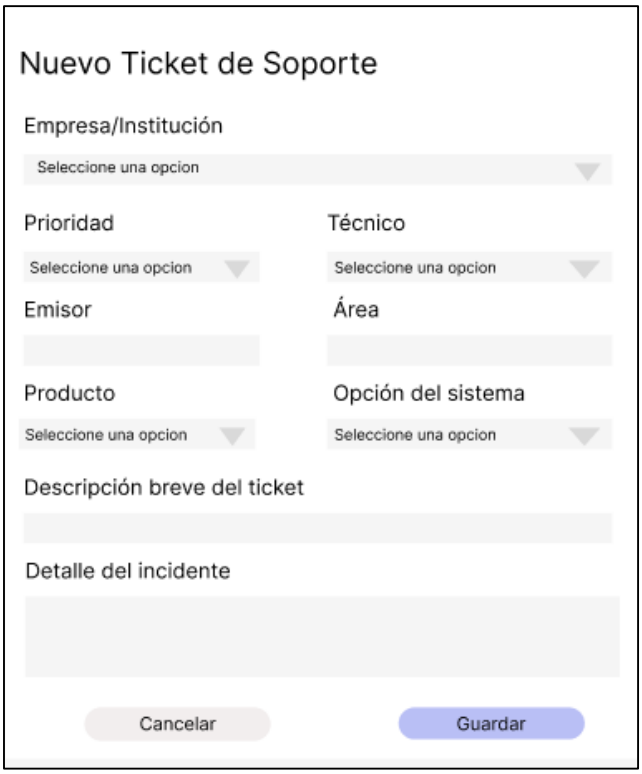


Ilustración 14 Prototipo de interfaz que permitirá el cargar un nuevo requerimiento con todas las especificaciones necesarias

4. Análisis

De acuerdo con la disciplina de análisis del Proceso Unificado de Rational (RUP), cuyo objetivo es identificar claramente qué debe hacer el sistema y definir una arquitectura lógica que permita cumplir con los requisitos, se ha realizado un estudio detallado que se presenta a continuación.

4.1. Análisis de Clases

Como resultado del análisis de clases, se han identificado las siguientes clases de software (ver imagen 15), que posteriormente se refinaron en la etapa de diseño.

Estas clases representan las dos principales capacidades del sistema desarrollado. Los atributos y métodos candidatos para el correcto modelado del proceso de negocio en la gestión de requerimientos se pueden visualizar en la Imagen 16.

En la Imagen 15, se muestran todas las clases junto con sus métodos y atributos que soportan la gestión de soporte técnico.

En ambos casos, existen clases comunes que interactúan con los dos procesos. Sin embargo, en esta etapa de análisis se ha preferido manejar una visión separada para comprender mejor estos procesos de manera individual.

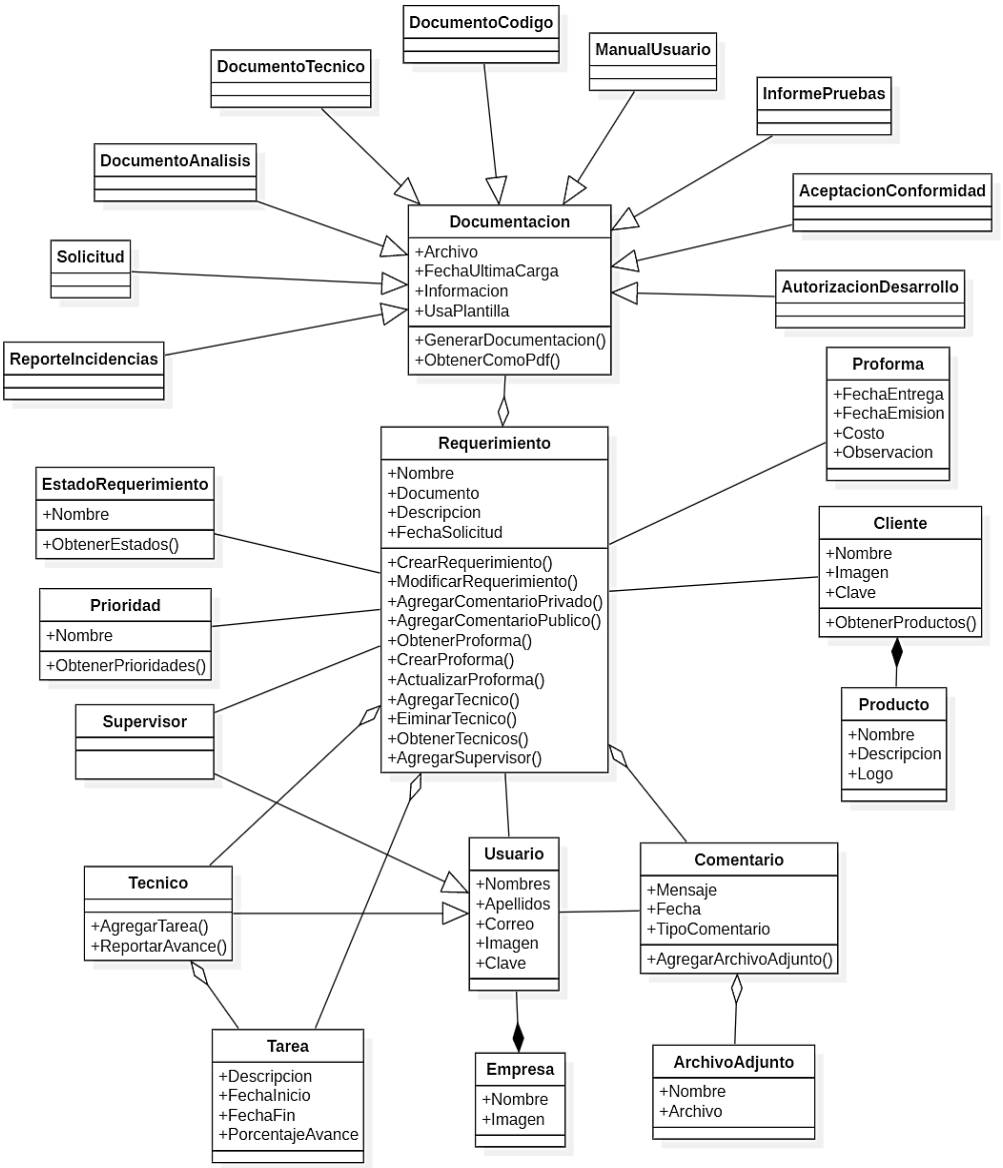


Ilustración 15 Diagrama de clases preliminar de la gestión de requerimientos

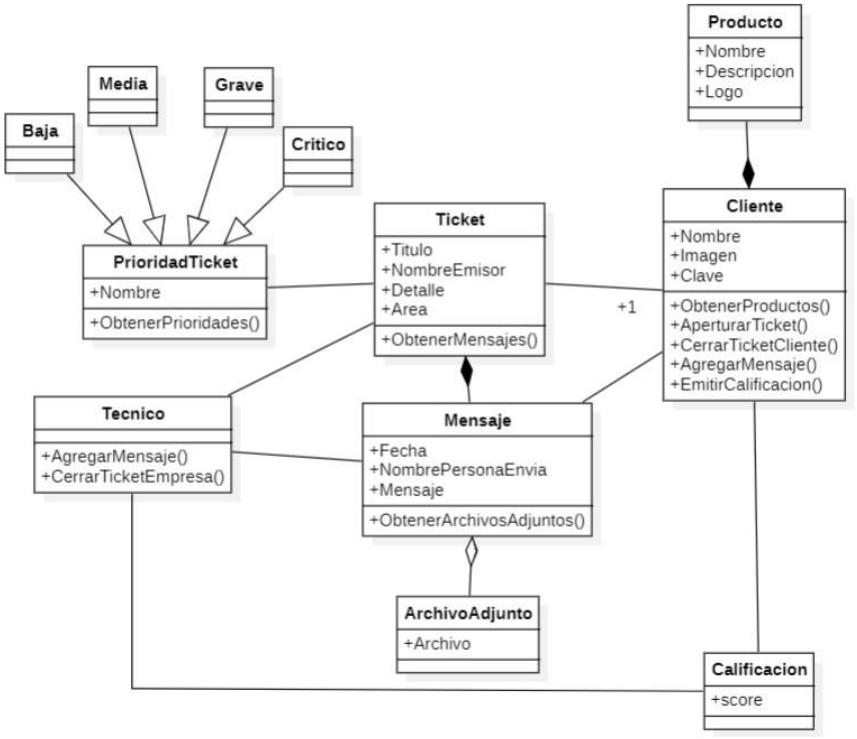


Ilustración 16 Diagrama de clases preliminar de la gestión de soporte técnico

4.2. Análisis de la arquitectura

Para el análisis de la arquitectura del sistema, se definió la siguiente estructura de paquetes como una versión preliminar. Esto permitió obtener una visión general de la arquitectura a utilizar en el proyecto, así como identificar posibles deficiencias y desafíos en su implementación.

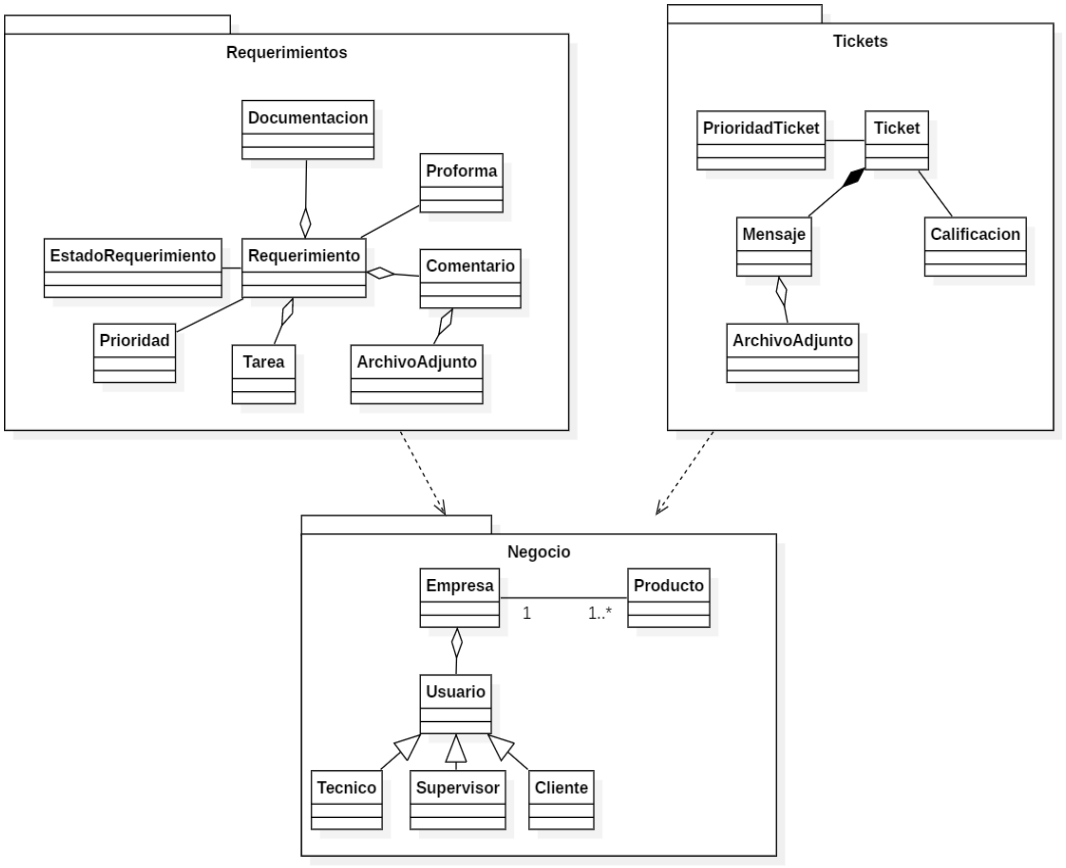


Ilustración 17 Diagrama de paquetes preliminar del sistema de gestión de incidentes y administración de servicios.

5. Diseño

El diseño es una de las disciplinas fundamentales del Proceso Unificado de Rational (RUP). En esta fase, se especifican las decisiones arquitectónicas y se perfecciona la arquitectura lógica identificada durante el análisis para garantizar que el sistema cumpla con todos los requisitos funcionales y no funcionales.

5.1. Diseño de la arquitectura

5.1.1. Capa de presentación

La capa de presentación está concebida para ofrecer una interfaz de usuario intuitiva y eficiente, facilitando una interacción fluida con el sistema. Esta capa fue desarrollada con Angular, respetando los principios de diseño definidos en los prototipos de interfaz de usuario creados durante la fase de especificación de casos de uso.

Los prototipos que se crearon inicialmente representaban las principales funcionalidades del sistema, incluyendo la búsqueda de requerimientos, la gestión de requerimientos y la apertura

de tickets de soporte. Estos prototipos sirvieron como guía para el desarrollo de las interfaces reales, asegurando que se mantuvieran las características clave y la usabilidad propuesta. A continuación, se presentan las interfaces de usuario implementadas, las cuales reflejan fielmente las características principales de los prototipos:

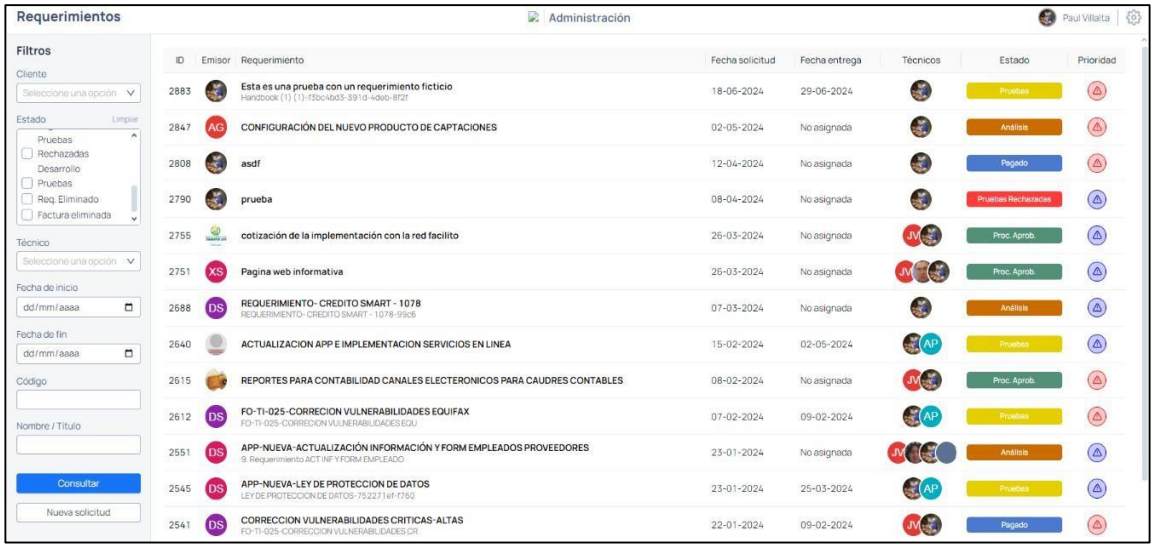


Ilustración 18 Interfaz de usuario para la búsqueda de requerimientos, diseñada para permitir a los usuarios filtrar y localizar rápidamente los requerimientos en el sistema, siguiendo las características del prototipo inicial

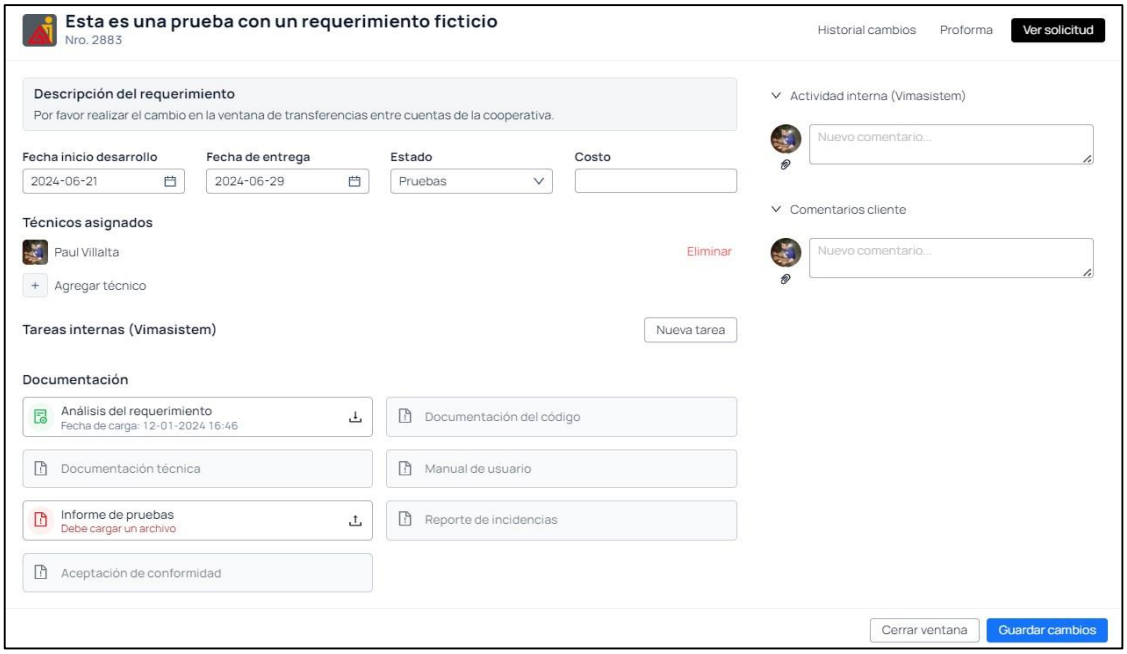


Ilustración 19 Interfaz de usuario para la gestión de requerimientos, permitiendo la creación, actualización y seguimiento de los requerimientos, construida de acuerdo con las especificaciones del prototipo.

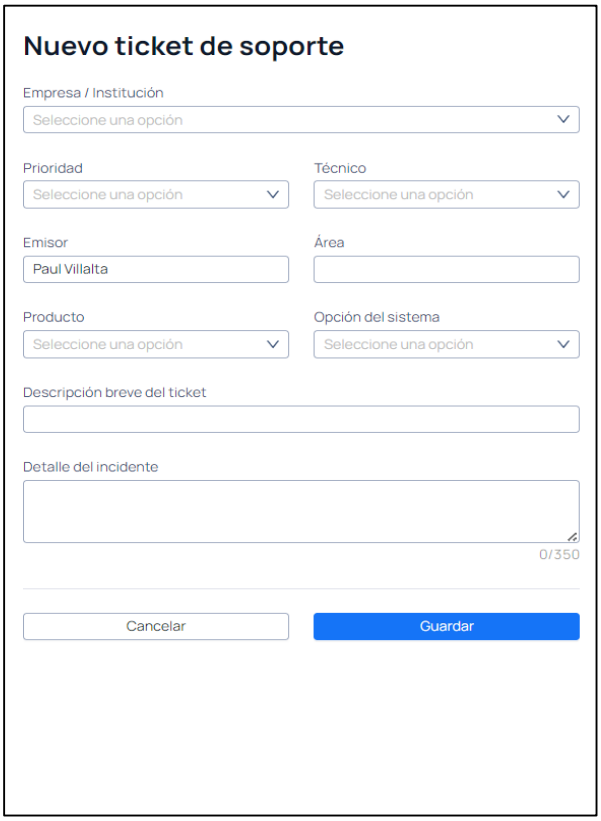


Ilustración 20 Interfaz de usuario para la apertura de tickets de soporte, facilitando a los usuarios reportar problemas y solicitar asistencia técnica, basada en las características definidas en el prototipo.

5.1.2. Capa de negocio

La capa de negocio implementa la lógica de negocio del sistema, gestionando las operaciones y reglas que rigen el funcionamiento del mismo. Las clases identificadas en la etapa de análisis se refinan y detallan para su implementación en esta capa. A continuación, en la imagen 21 se presenta el diagrama de clases correspondiente y, en la imagen 22 se muestran los enumerables que definen los diferentes estados y prioridades en la gestión de requerimientos y soporte técnico:

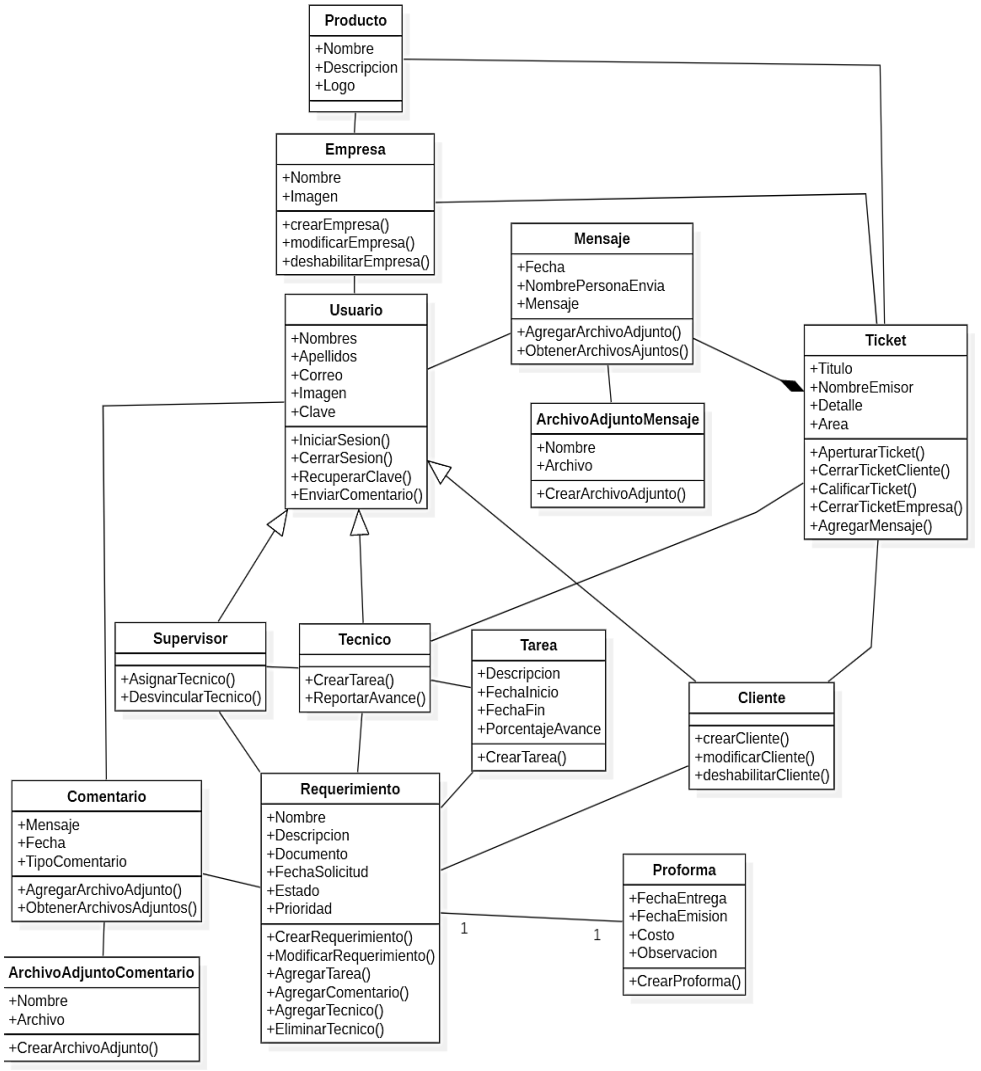


Ilustración 21 Diagrama de clases definitivo para la gestión de requerimientos y soporte técnico.

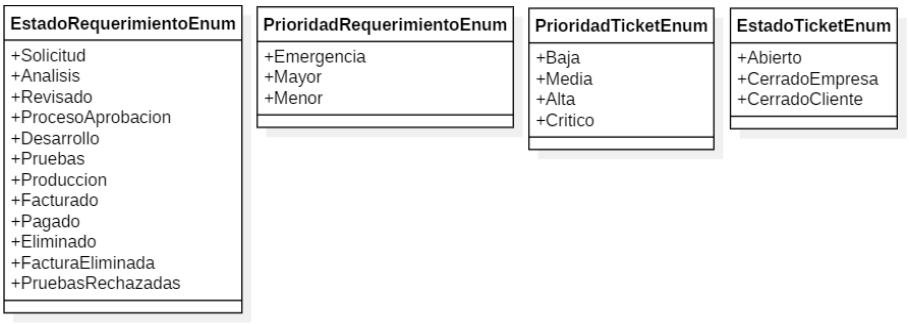


Ilustración 22 Enumerables que representan estados y prioridades para la gestión de requerimientos y soporte técnico: EstadoRequerimientoEnum y PrioridadRequerimientoEnum para requerimientos; PrioridadTicketEnum y EstadoTicketEnum para soporte técnico.

Durante esta etapa, se realizaron mejoras significativas en el diagrama de clases de análisis, principalmente enfocadas en la clarificación de las relaciones entre las clases y en la adición

de nuevos atributos y métodos que no fueron considerados inicialmente. Estas mejoras incluyen la identificación y separación de responsabilidades en diferentes clases, permitiendo una mejor modularización y reutilización del código. Además, se añadieron nuevas clases que representan entidades previamente no modeladas, mejorando así la representación del dominio del problema. Las actualizaciones realizadas facilitan una comprensión más clara del sistema y aseguran que todos los aspectos críticos están debidamente representados.

La estructura de los paquetes se diseñó siguiendo el esquema presentado en la sección de análisis de la arquitectura. En la imagen 23 se muestra la estructura de paquetes definitiva.

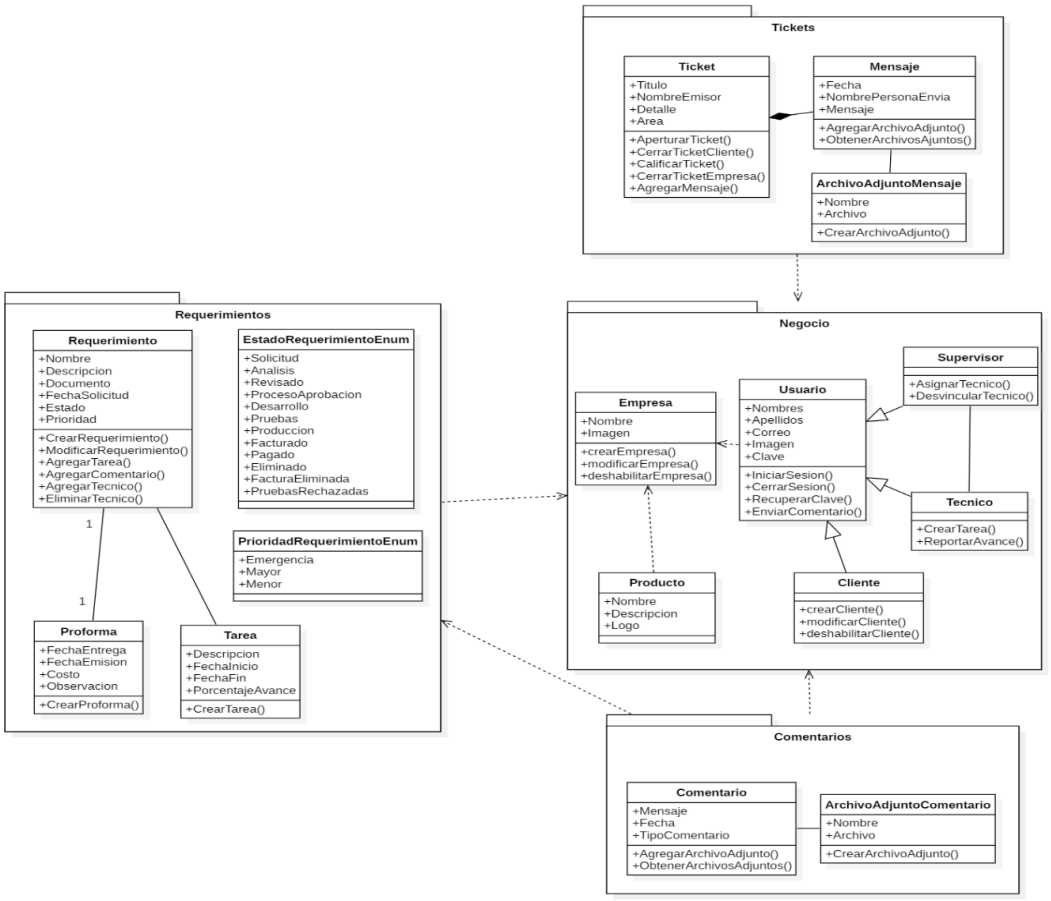


Ilustración 23 Diagrama de paquetes definitivo para la gestión de requerimientos y soporte técnico.

La estructura de paquetes también ha experimentado cambios importantes. Originalmente, la estructura de paquetes se diseñó de manera preliminar, enfocándose en una organización básica de las clases. En esta etapa, se ha refinado esta estructura para mejor reflejar la arquitectura lógica del sistema. Los paquetes se organizaron para agrupar de manera más coherente las clases relacionadas y para separar claramente las capas de presentación, negocio y datos. Esta nueva estructura permite una gestión más eficiente del código y facilita

el mantenimiento y la escalabilidad del sistema, asegurando que cada paquete tiene una responsabilidad claramente definida y que las dependencias entre ellos están minimizadas.

5.1.3. Capa de datos

La capa de datos de la herramienta desarrollada en esta tesis está diseñada para manejar la persistencia y recuperación de información crítica para la operación del sistema. Se ha definido un único esquema que agrupa todas las tablas necesarias para la gestión de requerimientos y soporte técnico. En la Imagen 24 se muestra el diagrama entidad-relación para la gestión de requerimientos, y en la Imagen 25, el diagrama entidad-relación para la gestión de soporte técnico.

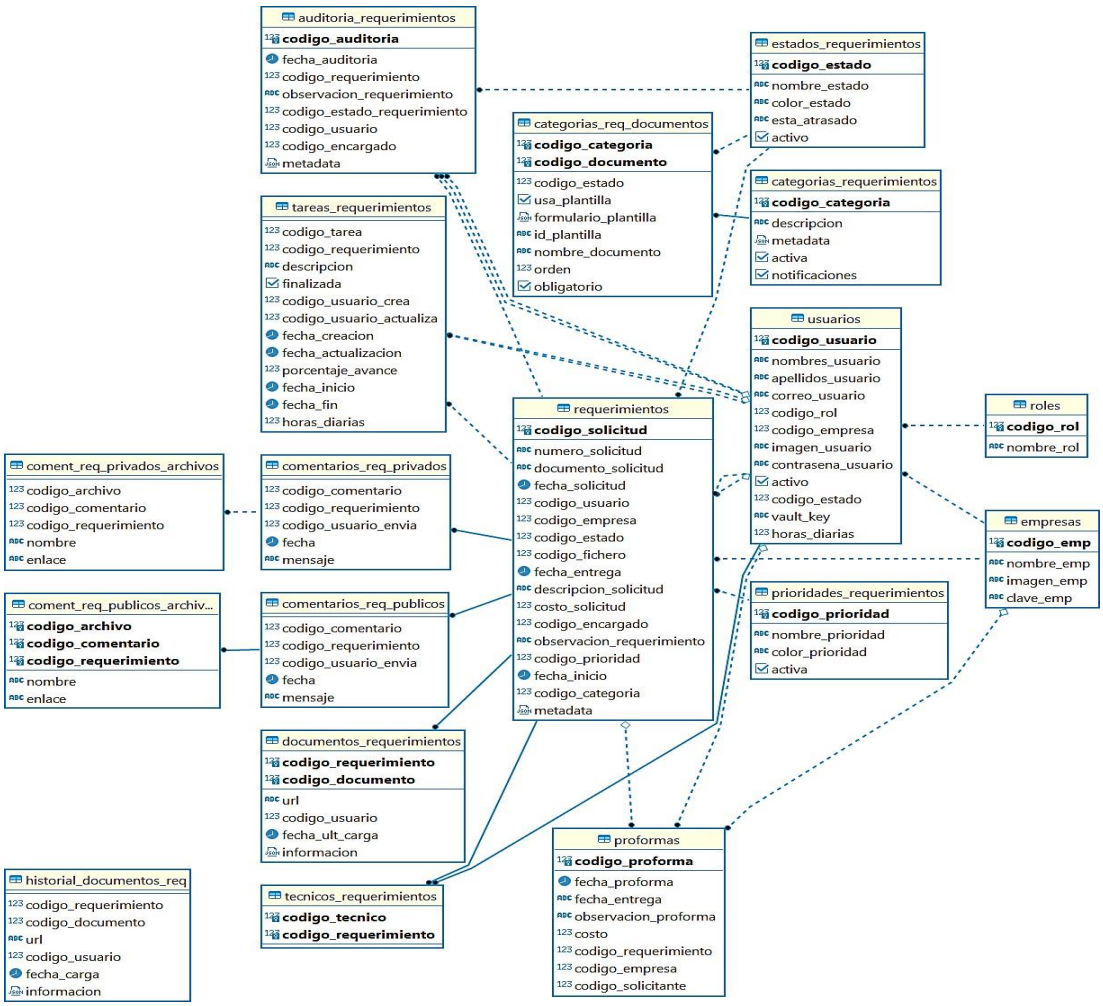


Ilustración 24 Diagrama entidad-relación de la gestión de requerimientos.

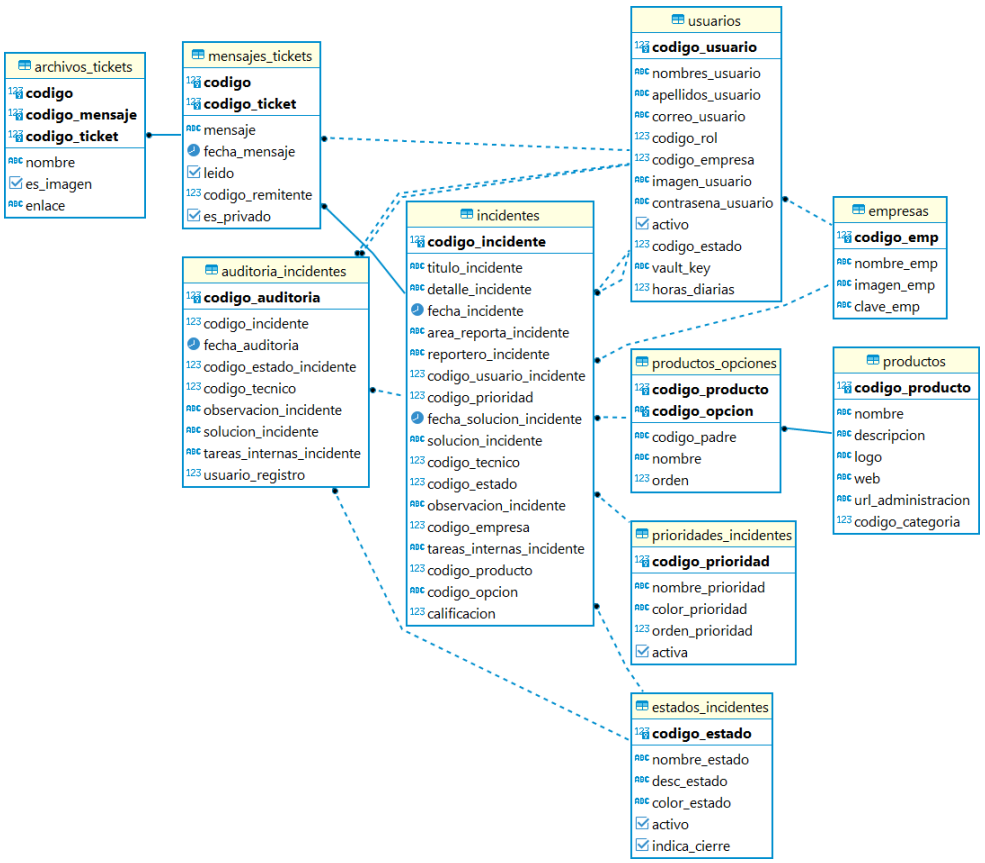


Ilustración 25 Diagrama entidad-relación para la gestión de soporte técnico

5.1.3.1. Gestión de Requerimientos

Como se mencionó en el capítulo 3.1, el objetivo principal del flujo de trabajo para la gestión de requerimientos se centra en la administración de solicitudes de desarrollo de software, abarcando desde su creación hasta su cierre. Las entidades clave que soportan este manejo de datos son:

Usuarios:

- Contiene información sobre los usuarios del sistema, incluyendo nombres, apellidos, correos electrónicos, roles, y estado de actividad.
- Relacionada con tablas como roles y empresas para una gestión integral de los usuarios.

Requerimientos:

- Almacena los detalles de cada requerimiento, como número de solicitud, fechas de creación y entrega, estado del requerimiento, categoría, y prioridad.
- Está relacionado con usuarios, empresas, y otros elementos como categorías y prioridades para proporcionar un contexto completo de cada requerimiento.

Tareas_requerimientos:

- Define las tareas específicas asociadas a cada requerimiento, permitiendo un seguimiento detallado del progreso y la asignación de responsables.
- Incluye información sobre el estado de finalización, fechas de inicio y fin, y porcentaje de avance.

Comentarios_req_privados y comentarios_req_publicos:

- Permiten a los usuarios añadir comentarios a los requerimientos, facilitando la comunicación interna y externa.
- Contienen campos para el contenido del comentario, el usuario que lo envía y la fecha de envío.

Documentos_requerimientos:

- Gestiona los documentos asociados a los requerimientos, incluyendo URLs de acceso, usuarios que los subieron y fechas de carga.

Proformas:

- Almacena las proformas generadas para los requerimientos, con detalles como fechas de creación y entrega, observaciones, y estado de aprobación.

Auditoria_requerimientos:

Registra las auditorías realizadas sobre los requerimientos, asegurando la trazabilidad y el control de cambios.

5.1.3.2. Gestión de Soporte Técnico

De igual manera, en el capítulo 3.1 también se habló del objetivo primordial de la gestión de soporte técnico, que se enfoca en la administración de incidentes reportados por los usuarios, proporcionando una solución estructurada para su resolución. Los elementos clave de este módulo son:

Usuarios:

- Esta tabla, que también se relaciona con la gestión de requerimientos, almacena información de los usuarios que interactúan con el sistema de soporte técnico.

Incidentes:

- Registra los incidentes, también denominados tickets en el modelo de dominio y clases de software definidas en los capítulos de análisis y diseño, reportados por los usuarios, desde su creación hasta su resolución.
- Incluye detalles como título, descripción, estado, prioridad, y fechas de reporte y solución.

Mensajes_tickets:

- Gestiona los mensajes intercambiados en el contexto de un ticket de soporte, facilitando la comunicación entre el técnico y el cliente.
- Contiene campos para el contenido del mensaje, el remitente, y la fecha de envío.

Archivos_tickets:

- Almacena los archivos adjuntos a los tickets de soporte, proporcionando una manera estructurada de gestionar la documentación relacionada con los incidentes.
- Incluye información sobre el nombre del archivo, si es una imagen y el enlace de acceso.

Auditoria_incidentes:

- Registra las auditorías realizadas sobre los incidentes, permitiendo un seguimiento detallado de las acciones tomadas y los cambios realizados.

Productos y productos_opciones:

- Almacenan información sobre los productos y sus opciones disponibles, que son gestionados a través del sistema de soporte técnico.

Prioridades_incidentes y estados_incidentes:

- Definen las prioridades y estados posibles para los incidentes, permitiendo una categorización y seguimiento adecuado de cada caso.

5.1.4. Diseño de la arquitectura vista de despliegue

Después de haber definido los prototipos de las interfaces gráficas para la gestión de requerimientos y soporte técnico, refinado las clases y paquetes obtenidos en la etapa de análisis, y habiendo establecido el modelo de persistencia de los datos en la etapa de diseño, se procedió a definir la vista de despliegue de la herramienta (ver imagen 26).

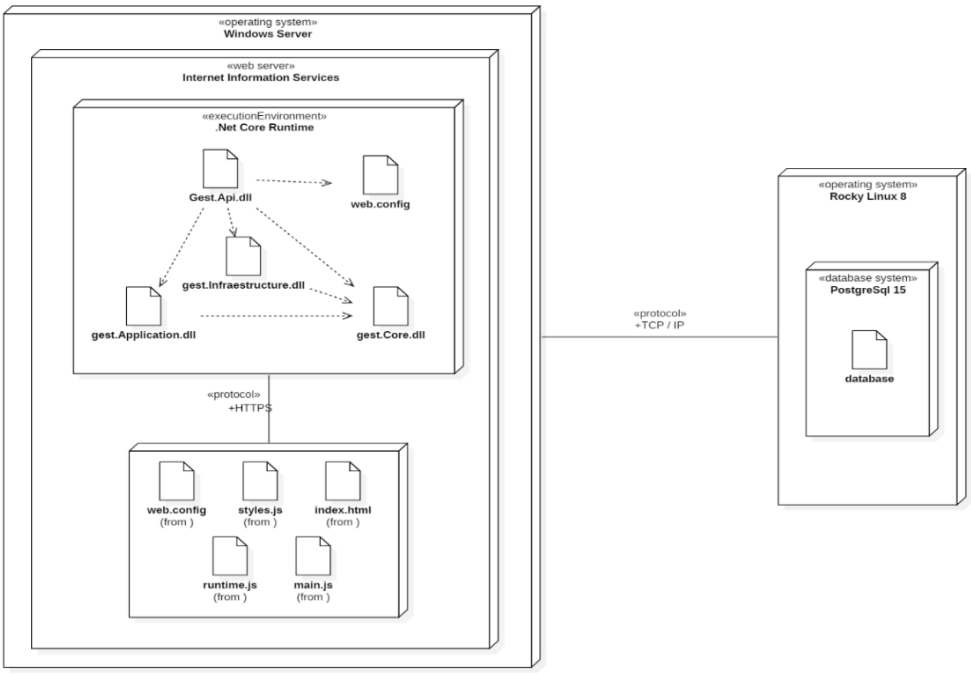


Ilustración 26 Diagrama de despliegue

El diagrama de despliegue presenta dos nodos principales: el servidor de aplicaciones y el servidor de base de datos, los cuales se comunican mediante el protocolo TCP/IP.

En el servidor de aplicaciones, el sistema operativo es Windows Server. Sobre Windows Server se encuentra el servidor web IIS, que alberga el entorno de ejecución .NET Core Runtime. En este entorno, se ejecuta el backend, que es una API REST. Los artefactos como los archivos DLL y el archivo web.config están representados en el diagrama.

Asimismo, sobre el servidor web IIS se encuentra la aplicación Angular, con sus archivos JS, HTML y CSS más importantes representados para ilustrar la estructura de la aplicación en el diagrama de despliegue.

6. Implementación

La sección de Implementación detalla el proceso práctico de desarrollo del sistema de Gestión de Incidentes y Administración de Servicios. Aquí se describe el entorno de desarrollo utilizado, la configuración del sistema, los pasos de desarrollo, la integración de componentes, y las pruebas realizadas con usuarios finales. El propósito es ofrecer una visión clara y detallada del desarrollo del proyecto, garantizando que el sistema cumpla con los requisitos y objetivos establecidos en las fases iniciales.

6.1. Entornos de desarrollo

La implementación del sistema de Gestión de Incidentes y Administración de Servicios se llevó a cabo utilizando herramientas y tecnologías específicas que fueron seleccionadas cuidadosamente para satisfacer tanto los requisitos técnicos del proyecto como las

preferencias de la empresa. En las siguientes líneas, se detallan los lenguajes de programación, frameworks y entornos de desarrollo integrados (IDE) empleados.

6.1.1. Lenguajes de programación

Para el desarrollo del sistema de Gestión de Incidentes y Administración de Servicios, se han elegido C# como lenguaje de programación para el backend y JavaScript para el frontend. C# fue seleccionado debido a su madurez, estabilidad, y alto rendimiento, lo cual es crucial para manejar grandes volúmenes de datos de manera eficiente y segura. Además, C# ofrece un amplio ecosistema de herramientas y bibliotecas, y su compatibilidad con .NET Core permite el desarrollo multiplataforma.

Por otro lado, JavaScript fue escogido para el frontend por su popularidad y adopción generalizada en el desarrollo web, lo que asegura una gran cantidad de recursos y soporte. JavaScript permite crear interfaces de usuario interactivas y dinámicas, mejorando la experiencia del usuario final. Además, su capacidad de ejecución en el cliente reduce la carga del servidor y mejora el rendimiento general de la aplicación. La elección de estos lenguajes también se alinea con las prácticas y preferencias tecnológicas de la empresa, facilitando la integración y mantenimiento del sistema dentro de su infraestructura existente.

6.1.2. Frameworks

Para el desarrollo de la herramienta propuesta en este proyecto de tesis, se han elegido los frameworks .NET Core para el backend y Angular para el frontend.

.NET Core se seleccionó por su capacidad multiplataforma, alto rendimiento y características avanzadas de seguridad, esenciales para aplicaciones empresariales que requieren escalabilidad y robustez. Su arquitectura modular permite usar únicamente los componentes necesarios, optimizando el rendimiento y la eficiencia del sistema.

Angular, por su parte, se escogió debido a su capacidad para crear aplicaciones web dinámicas y escalables mediante una arquitectura basada en componentes. Ofrece un ecosistema robusto con herramientas que facilitan el desarrollo y mantenimiento de aplicaciones complejas, y su soporte para aplicaciones de una sola página (SPA) mejora significativamente la experiencia del usuario final. La elección de estos frameworks también se alinea con las prácticas tecnológicas de la empresa, asegurando una implementación coherente y un mantenimiento eficiente del sistema.

6.1.3. IDEs

Los entornos de desarrollo integrado (IDE) seleccionados para el desarrollo del sistema fueron Visual Studio para el lado del backend y Visual Studio Code para el lado del frontend. Visual Studio fue seleccionado debido a su robustez y sus características avanzadas, que facilitan el desarrollo en .NET Core. Ofrece una depuración integrada, administración de proyectos eficiente y soporte para una amplia gama de extensiones que mejoran la

productividad del desarrollador. Estas características son esenciales para manejar la complejidad de la lógica de negocio y la persistencia de datos en el backend.

Por otro lado, Visual Studio Code se escogió como IDE para el frontend debido a su ligereza y flexibilidad. Es un editor de código fuente poderoso que admite una amplia gama de lenguajes y frameworks, incluyendo Angular. Su capacidad de integración con sistemas de control de versiones como Git y su vasta colección de extensiones permiten a los desarrolladores personalizar su entorno de trabajo según sus necesidades específicas. Además, Visual Studio Code proporciona una experiencia de desarrollo ágil y eficiente, lo cual es crucial para la creación de interfaces de usuario dinámicas y responsivas.

6.2. Configuración del entorno

Para la configuración del entorno, se realizaron las siguientes instalaciones y configuraciones:

6.2.1. Instalación de Visual Studio

Se descargó Visual Studio desde la página oficial y se instaló seleccionando la edición Community, que es gratuita y suficientemente robusta para el desarrollo en .NET Core. Durante la instalación, se seleccionaron las cargas de trabajo específicas para el desarrollo de .NET Core, lo que incluye herramientas y bibliotecas necesarias para trabajar con C#. Después de la instalación, se configuraron extensiones adicionales como herramientas de integración continua y opciones avanzadas de depuración, lo que facilitó un entorno de desarrollo optimizado para el backend.

6.2.2. Instalación de Visual Studio Code

Para el frontend, se descargó Visual Studio Code desde la página oficial e instaló en el sistema. Se configuró con extensiones esenciales como Angular Language Service y Prettier para formateo de código. Estas extensiones mejoraron la experiencia de desarrollo, proporcionando un entorno ligero y flexible ideal para trabajar con Angular y JavaScript.

6.2.3. Instalación de .Net Core

El SDK de .NET Core se descargó desde la página oficial de .NET y se instaló siguiendo las instrucciones específicas para el sistema operativo utilizado. La instalación se verificó mediante la ejecución del comando dotnet --version en la terminal, asegurando que .NET Core estuviera correctamente instalado y listo para el desarrollo del backend.

6.2.4. Instalación de Node.js y Angular CLI

Node.js se descargó e instaló desde su página oficial. Posteriormente, se instaló Angular CLI mediante el comando npm install -g @angular/cli, lo que permitió la creación y gestión de proyectos Angular. Se verificó la instalación ejecutando el comando ng --version para asegurar que Angular CLI estuviera correctamente configurado.

6.2.5. Configuración de proyectos en C# y Javascript

En Visual Studio, se creó un nuevo proyecto de .NET Core utilizando C#. Se configuró la estructura de carpetas y se agregaron las dependencias necesarias para el desarrollo del sistema, tales como Entity Framework Core para el acceso a datos, AutoMapper para el mapeo de objetos, y Newtonsoft.Json para la manipulación de JSON.

En Visual Studio Code, se creó un nuevo proyecto Angular utilizando Angular CLI, asegurando que todas las dependencias estuvieran correctamente instaladas, incluyendo NG ZORRO para componentes de la interfaz de usuario y RxJS para programación reactiva. Se configuraron las rutas y componentes necesarios para la aplicación frontend.

6.2.6. Configuración de base de datos Postgresql

PostgreSQL se descargó desde la página oficial y se instaló en el servidor con sistema operativo Rocky Linux. Se creó una nueva base de datos y se configuró un único esquema para almacenar toda la información relacionada con la gestión de requerimientos y soporte técnico. Los parámetros de conexión, el nombre de la base de datos, el nombre de usuario y la contraseña se configuraron en el backend para garantizar una comunicación eficiente con la base de datos.

6.3. Desarrollo del sistema

Luego de completar las disciplinas de análisis y diseño mencionadas en RUP, se procedió con la implementación, en donde el objetivo fue convertir el diseño del sistema en un código ejecutable y bien estructurado.

6.3.1. Estructura del proyecto

El proyecto se organizó en dos partes principales: el backend y el frontend. Cada parte se estructuró en módulos y componentes para facilitar el desarrollo y el mantenimiento.

6.3.1.1. Backend

El backend del proyecto se encuentra dividido en cuatro paquetes o directorios: Gest.Api, Gest.Core, Gest.Application y Gest.Infraestructure.

Gest.Api: Contiene todos los controladores que gestionan las solicitudes HTTP y definen las rutas de la API. Por ejemplo, el RequerimientosController maneja las solicitudes relacionadas con la creación, actualización y eliminación de requerimientos.

Algunos de los controladores se pueden observar en la imagen 27:

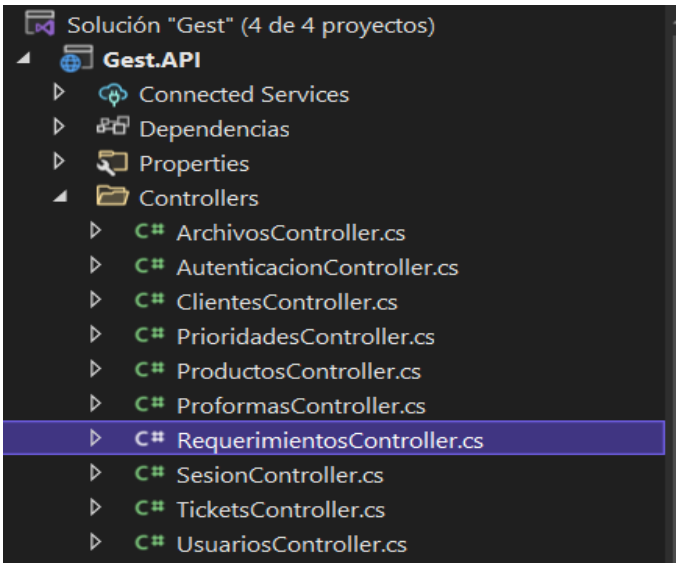


Ilustración 27 Controladores que reciben las peticiones HTTP

Gest.Application: Incluye clases de servicio que gestionan lo recibido por los controladores y orquestan el trabajo con el dominio, el cual se encuentra en Gest.Core, y la persistencia de datos, que se encuentra en Gest.Infraestructure. En la imagen 28 se observa el directorio con las clases de servicio implementadas.

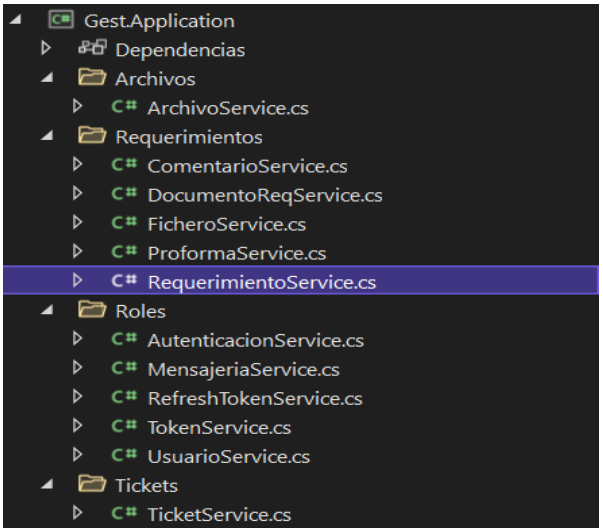


Ilustración 28 Clases de servicio que orquestan la interacción entre los controladores, la lógica de dominio y la persistencia de datos.

Gest.Core: Contiene toda la lógica de negocio del sistema. La imagen 29 muestra las distintas clases que contienen la lógica de negocio del sistema.

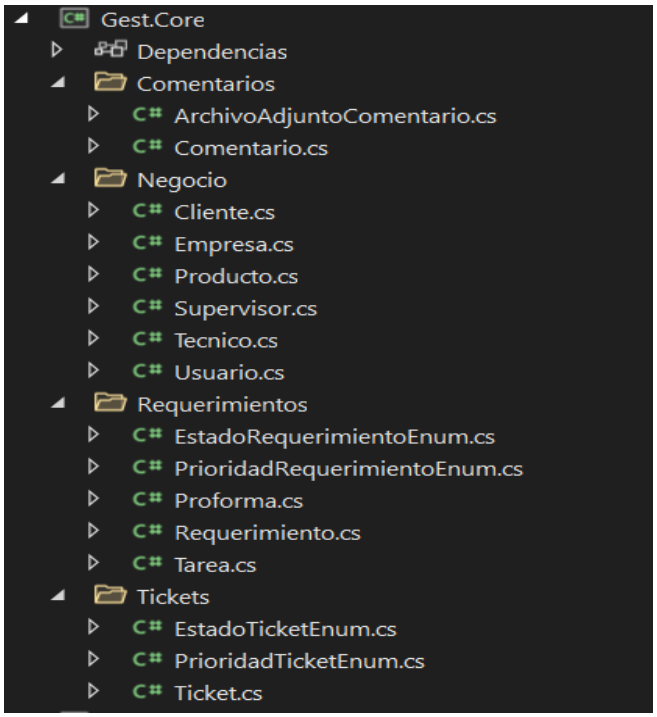


Ilustración 29 Clases de dominio esenciales que manejan la lógica de negocio del sistema.

Gest.Infraestructure: Maneja la persistencia de datos del sistema. Para esto se ha utilizado Entity Framework Core (EF Core) para la persistencia de datos. EF Core es un mapeador objeto-relacional (ORM) para .NET Core que facilita la interacción con la base de datos mediante objetos .NET, reduciendo la necesidad de escribir manualmente la mayor parte del código de acceso a datos. Esto simplifica la interacción con la base de datos y garantiza una integración eficiente y segura de los datos en la aplicación.

En la imagen 30 se pueden ver las clases que manejan la persistencia y las entidades correspondientes a la base de datos implementada.

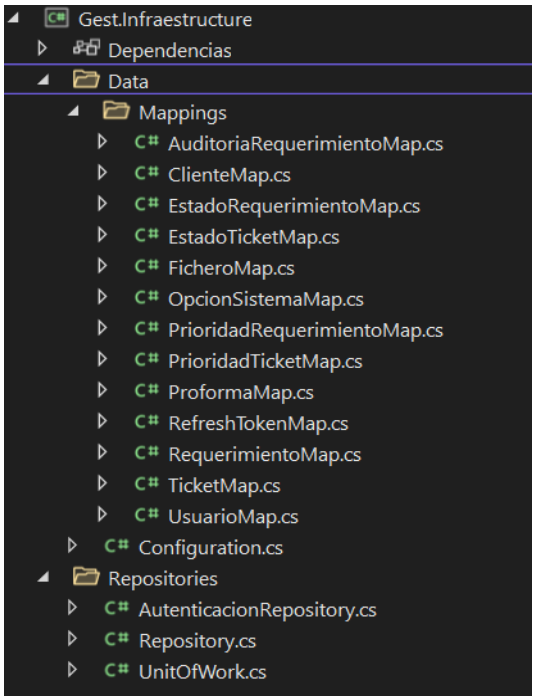


Ilustración 30 Clases que manejan la persistencia de datos usando EF Core. Incluye también clases que se encargan de mapear las peticiones con las entidades correspondientes.

6.3.1.2. Frontend

Componentes: Los componentes se estructuraron en módulos según las funcionalidades. Por ejemplo, el RequerimientosModule incluye componentes como ListaRequerimientosComponent y DetalleRequerimientoComponent. En la imagen 31 se puede observar los archivos que componen este módulo.

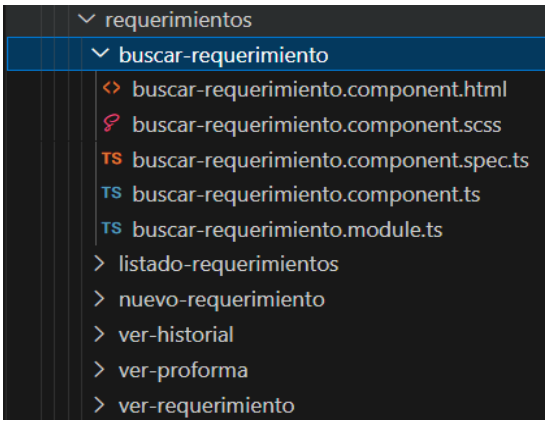


Ilustración 31 Módulo de requerimientos que incluye las vistas para la búsqueda y creación de requerimientos.

Servicios: Los servicios en el frontend gestionan la comunicación con el backend utilizando HTTPClient. Por ejemplo, RequerimientosService realiza solicitudes HTTP para interactuar con las API del backend.

En la imagen 32 se puede observar dichos servicios.

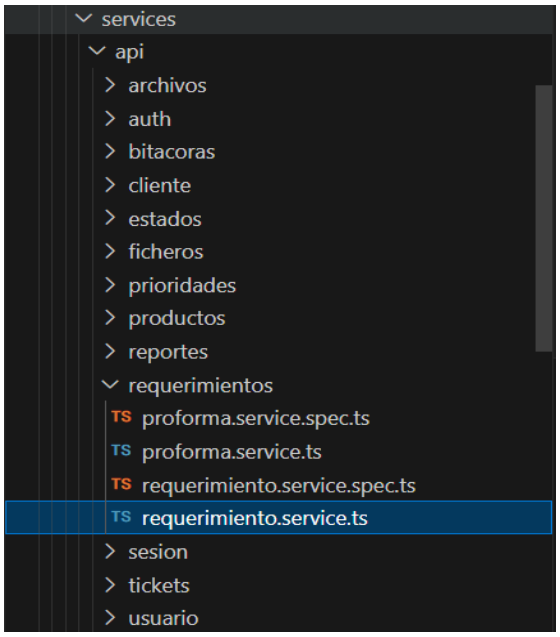


Ilustración 32 Servicios encargados de la comunicación con el backend utilizando el módulo HttpClient de Angular.

Rutas: Se configuraron rutas en app-routing.module.ts para la navegación entre las distintas vistas de la aplicación. Por ejemplo, la ruta /requerimientos carga el componente ListaRequerimientosComponent. Esto se puede observar en la imagen 33.

```
routes: Routes = [

  path: '', component: MainLayoutComponent, children: [
      { path: 'inicio', loadChildren: () => import('../../pages/inicio/inicio/inicio.module').then(m => m.InicioModu
      { path: 'soporte', loadChildren: () => import('../../pages/soporte/listado-soporte/listado-soporte.module').th
      { path: 'soporte/:codigo', loadChildren: () => import('../../pages/soporte/ver-soporte/ver-soporte.module').th
      { path: 'requerimientos', loadChildren: () => import('../../pages/requerimientos/listado-requerimientos/listad

      { path: 'documentos', loadChildren: () => import('../../pages/documentos/listado-documentos/listado-documentos

      { path: 'productos', loadChildren: () => import('../../pages/productos/listado-productos/listado-productos.mod
      { path: 'productos/oracle/autonomous/:codigoCliente', loadChildren: () => import('../../pages/productos/admini


      { path: '*', loadChildren: () => import('../../pages/inicio/inicio/inicio.module').then(m => m.InicioModule) }
  ]
```

Ilustración 33 Contenido del archivo que maneja las rutas de las vistas principales del sistema.

6.4. Integración

La integración del sistema implicó la conexión del frontend con el backend y la base de datos para asegurar un flujo de datos coherente y eficiente.

6.4.1. Comunicación entre Frontend y Backend

Uso de Servicios HTTP: Se implementaron servicios HTTP en el frontend para enviar solicitudes al backend y recibir respuestas. Esto incluyó el uso de Angular HTTPClient para realizar las distintas operaciones a través de la API RESTful del backend.

6.4.2. Manejo de Errores

Implementación de Manejo de Errores: Se desarrollaron mecanismos para manejar errores tanto en el frontend como en el backend. En el backend, se utilizaron middleware de ASP.NET Core para capturar y registrar excepciones (ver imagen 34). En el frontend, se implementaron interceptores HTTP para gestionar errores y mostrar mensajes de error amigables al usuario (ver imagen 35).

```
4 referencias
public class GlobalExceptionFilter : IExceptionFilter
{
    private readonly GeneralOption _options;
    private readonly ILogger<GlobalExceptionFilter> _logger;

    0 referencias
    public GlobalExceptionFilter(IOptions<GeneralOption> options, ILogger<GlobalExceptionFilter> logger)
    {
        _options = options.Value;
        _logger = logger;
    }

    0 referencias
    public void OnException(ExceptionContext context)
    {
        //var logger = new ErrorLogger(_options);
        //logger.Log($"[Excepcion desconocida]: {context.Exception}");

        if (context.Exception.GetType() == typeof(BusinessException))
        {
            var exception = (BusinessException)context.Exception;
            var validation = new
            {
                Status = 400,
                Title = "Bad Request",
                Detail = $"{exception.Message}"
            };

            var json = new
            {
                errors = new[] { validation }
            };
```

Ilustración 34 Middleware de ASP.NET Core para la captura y registro de excepciones.

```
export class AuthInterceptorsService implements HttpInterceptor {

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
      let request = req;
      let token = this.storage.getInformation()?.Token;
      const prefix = environment.apiUrl;

      if (!req.headers.has('Content-Type')) {
          if(!req.headers.has('Auto-header')) {
              req = req.clone({
                  headers: req.headers.set('Content-Type', 'application/json')
              });
          }
      }
      return next.handle(request).pipe(
          tap((data: any) => {
              if(data.body) {
                  if(data.body.Data && !environment.production) {
                      console.log('RESPONSE:', data.body.Data)
                  }
                  if(data.body.Meta && !environment.production) {
                      console.log('METADATA:', data.body.Meta)
                  }
              }
          }),
          shareReplay(),
          catchError((err: any) => {
              if (err instanceof HttpErrorResponse && err.status === 401) {
                  return this.handle401Error(request, next)
              } else {
                  console.log(err)
                  console.log('here')
                  return throwError(err['error']['errors']);
              }
          })
      );
```

Ilustración 35 Interceptor HTTP para la gestión de errores en angular.

6.4.3. Autenticación y Autorización

Uso de JWT (JSON Web Tokens): Se implementó un sistema de autenticación y autorización basado en JWT. Los usuarios se autentican mediante el envío de credenciales, y el servidor responde con un token JWT. Este token se incluye en las solicitudes subsecuentes para acceder a recursos protegidos.

6.5. Pruebas y validación

Se realizaron diversas pruebas con usuarios finales para asegurar que el sistema funcionara correctamente y cumpliera con los requisitos de los usuarios.

6.5.1. Pruebas de Usuario Final

- Objetivo: Asegurar que los flujos de trabajo funcionen correctamente y que los usuarios puedan realizar sus tareas sin problemas.
- Método: Se llevaron a cabo pruebas con supervisores y técnicos para validar diferentes aspectos del sistema.
- Resultados:
  - Supervisores: Verificaron que podían recibir y asignar solicitudes de requerimientos correctamente.
  - Técnicos: Confirmaron que podían analizar requerimientos, desarrollarlos y gestionar tickets de soporte.

6.6. Conclusión

El sistema desarrollado cumple con los objetivos planteados al inicio, proporcionando una plataforma robusta para la gestión de requerimientos y soporte técnico.

Durante el proceso de implementación se aprendieron valiosas lecciones sobre la integración de componentes y la gestión de datos. Para futuras mejoras, se considera la implementación de pruebas unitarias e integraciones adicionales para mejorar la calidad del código y la detección temprana de errores.

7. Planificación del experimento

7.1. Objetivo del experimento

Evaluar la facilidad de uso y la efectividad de una nueva herramienta de gestión de incidentes y administración de servicios desarrollada para una empresa de software, utilizando evaluadores expertos con experiencia en procesos de desarrollo de software.

7.2. Definición del contexto

7.2.1. Selección de sujetos

Para este experimento, se seleccionaron evaluadores expertos, ya que tienen la capacidad de detectar más problemas que los evaluadores novatos [48]. La aplicación fue desarrollada para una empresa de desarrollo de software, y los participantes del experimento son profesionales con experiencia en procesos de desarrollo de software. El perfil de estos participantes incluye Ingenieros en sistemas con al menos 3 años de experiencia en la empresa, con amplio conocimiento y experiencia en los procesos de desarrollo de software. Por ello, se seleccionaron 9 sujetos que cumplen con dichas características, asegurando así que el experimento sea evaluado por personas con el conocimiento y la experiencia necesarios para proporcionar un análisis profundo y detallado del sistema. La participación de estos evaluadores expertos es crucial para obtener resultados precisos y relevantes en la evaluación de la aplicación, gracias a su alto nivel de competencia en el campo del desarrollo de software.

El experimento se realizó en un entorno en línea dentro de una empresa. Los participantes fueron empleados de la empresa, todos con experiencia previa en la gestión de incidentes y administración de servicios. Dada su experiencia en el dominio, estos evaluadores expertos están mejor capacitados para identificar problemas y proporcionar una evaluación más precisa y detallada en comparación con evaluadores menos experimentados.

7.3. Preparación y ejecución del experimento

El experimento se dividió en dos sesiones. En el primer día, se realizó una sesión de entrenamiento de 120 minutos antes de la sesión experimental. El propósito de esta sesión fue familiarizar a los participantes con las diferentes opciones de la herramienta y ofrecer dos

escenarios guiados que abarcaban todo el proceso relacionado con el flujo de gestión de requerimientos y soporte técnico, respectivamente.

El escenario planteado para la gestión de requerimientos abarcaba aspectos relevantes como la búsqueda de requerimientos, el proceso de análisis, la asignación de técnicos, la generación de documentación, y más, hasta la fase de producción. Las fases posteriores (facturación y pago) no se tomaron en cuenta ya que estas simplemente cambian el estado del requerimiento y son validadas externamente.

El escenario dispuesto para la gestión de soporte técnico incluía todas las actividades relacionadas con la gestión de soporte técnico, como la búsqueda de tickets de soporte, la interacción con el cliente a través del chat proporcionado por la interfaz, y la finalización del flujo mediante el cierre de los tickets.

En la segunda sesión se plantearon los mismos escenarios, con la finalidad de que los participantes pudieran completar las actividades planteadas en cada uno de ellos. El objetivo fue evaluar cómo el sistema facilita el registro de dichas actividades.

7.3.1. Escenario de la gestión de requerimientos

Para la gestión de requerimientos, se proporcionaron usuarios con roles de supervisores y técnicos, así como una serie de requerimientos en estado de solicitud para repasar todo el flujo normal que siguen los requerimientos.

Tarea	Actividad Realizada	Finalidad
Acceso y Búsqueda de Requerimientos	Con el usuario supervisor, los participantes debían ingresar al sistema y buscar algún requerimiento en estado de solicitud utilizando los filtros de búsqueda.	Identificar y gestionar los requerimientos pendientes para asegurar una atención oportuna y eficiente.
Asignación de Técnico	Seleccionar el requerimiento y asignar un usuario técnico al mismo.	Asignar responsabilidades y garantizar que cada requerimiento tenga un técnico encargado para su resolución.
Cambio de Estado a Análisis	Establecer el requerimiento en estado de análisis.	Indicar el inicio del proceso de análisis del requerimiento, asegurando que se evalúe

		adecuadamente antes de proceder.
Acceso con Usuario Técnico	Ingresar al sistema con el usuario técnico asignado al requerimiento.	Permitir al técnico acceder y gestionar los requerimientos que le han sido asignados.
Búsqueda por Código	Buscar el requerimiento utilizando el código proporcionado.	Facilitar la localización precisa y rápida de un requerimiento específico mediante su código identificador.
Generación de Documento de Análisis y Cambio de Estado a Revisado	Generar el documento de análisis y establecer el requerimiento en estado "revisado".	Documentar el análisis realizado y actualizar el estado del requerimiento para reflejar su progreso.
Logueo con Usuario Supervisor y Establecimiento de Fechas	Volver a realizar un inicio de sesión con el usuario supervisor y seleccionar el requerimiento en proceso. Establecer una fecha de cuándo iniciará el desarrollo y la posible fecha de entrega.	Planificar el desarrollo y establecer plazos claros para el inicio y la finalización del requerimiento.
Generación y Envío de Proforma	Generar y enviar la proforma correspondiente al requerimiento.	Formalizar el costo y las condiciones del requerimiento, y enviarlas al cliente para su aprobación.
Cambio de Estado a Proceso de Aprobación	Cambiar el estado del requerimiento a "proceso de aprobación". Explicación: En este paso, se simula que el cliente acepta la proforma (esta función no la realiza el sistema, solo para efectos del experimento).	Simular la aceptación del cliente de la proforma, lo que permite avanzar al siguiente estado del proceso.

Cambio de Estado a Desarrollo	Cambiar el estado del requerimiento a "desarrollo".	Indicar el inicio del trabajo de desarrollo sobre el requerimiento, reflejando el avance en el proceso.
Cambio de Estado a Pruebas con Usuario Técnico	Iniciar sesión nuevamente con el usuario técnico, seleccionar el requerimiento y cambiar el estado a "pruebas".	Marcar la transición del requerimiento a la fase de pruebas, donde se verificará su funcionalidad y calidad.
Carga de Documento de Pruebas y Cambio de Estado a Producción	Cargar el documento o informe de pruebas y establecer el estado del requerimiento en "producción". Explicación: Para efectos del experimento, se simula que el cliente acepta el desarrollo realizado.	Documentar los resultados de las pruebas y actualizar el estado a "producción", simulando la aceptación del cliente y la finalización del desarrollo.

Tabla 1 Tala del Experimento de Gestión de Requerimientos

Con este flujo detallado de actividades, los participantes pudieron familiarizarse y evaluar cómo el sistema facilita el registro y gestión de los requerimientos, asegurando que todas las fases críticas del proceso se cubran adecuadamente.

7.3.2. Escenario de la gestión de soporte técnico

Para la gestión de tickets de soporte, se proporcionaron usuarios con rol de técnico y tickets abiertos asignados a dichos técnicos, permitiendo que los participantes realicen todo el flujo normal de atención de un ticket hasta la fase en la que el ticket se encuentra cerrado por la empresa (es decir, cuando el técnico ya ha atendido la solicitud).

Tarea	Actividad Realizada	Finalidad
Acceso a la Opción de Soporte Técnico	Como primer paso, los participantes debían iniciar sesión con el usuario técnico y dirigirse a la opción de soporte técnico en el sistema.	Permitir al técnico acceder a la sección de soporte técnico para gestionar tickets de soporte.
Búsqueda de Tickets de Soporte	Una vez en la opción de soporte técnico, se solicitó que buscarán tickets de soporte aplicando los filtros disponibles en la	Identificar los tickets de soporte pendientes para su atención.

	herramienta para refinar la búsqueda.	
Selección de Tickets en Estado Abierto	Se pidió que los participantes buscaran y seleccionaran específicamente tickets que se encontraban en estado abierto.	Priorizar la atención de tickets que requieren intervención inmediata.
Envío de Mensajes y Cambio de Estado a Cerrado Empresa	Se solicitó que enviaran mensajes en la ventana de chat de la pantalla de gestión de tickets de soporte, interactuando con el cliente según el procedimiento habitual. Posteriormente, se les pidió que cambiaran el estado del ticket a "cerrado empresa".	Comunicarse con el cliente para resolver el ticket y actualizar el estado a "cerrado empresa" una vez solucionado.
Verificación del Estado del Ticket	Finalmente, se solicitó que buscaran nuevamente el ticket atendido para verificar que ya no se encontraba en estado abierto, asegurándose de que el cambio de estado se haya registrado correctamente.	Asegurar que el ticket ha sido cerrado correctamente y que los cambios de estado se reflejan en el sistema.

Tabla 2 Experimento Gestión de Tickets

Este flujo de actividades permitió que los participantes evaluaran la facilidad de uso y efectividad del sistema en la gestión de soporte técnico, asegurando que todos los pasos críticos del proceso, desde la búsqueda y selección de tickets hasta la interacción con el cliente y el cierre del ticket, fueran cubiertos adecuadamente.

8. Resultados

En este capítulo se presentan los resultados obtenidos de las evaluaciones de usabilidad y experiencia de usuario realizadas para el Sistema de Gestión de Incidentes y Administración de Servicios. Los encuestados, que son los mismos 9 usuarios seleccionados para el experimento, proporcionaron sus opiniones y percepciones mediante dos cuestionarios distintos: el Cuestionario de Usabilidad (USU) y la Encuesta de Experiencia de Usuario (UEQ). A continuación, se detallan los resultados y análisis obtenidos a partir de estos instrumentos de evaluación.

8.1. Evaluación mediante Cuestionario de Usabilidad (USU)

En esta sección se presentan los resultados obtenidos del cuestionario de Usabilidad USU aplicado a los usuarios del Sistema de Gestión de Incidentes y Administración de Servicios. El cuestionario USU permite capturar las opiniones de los usuarios sobre diversos aspectos clave del sistema, incluyendo la complejidad percibida, la integración de funciones y la curva de aprendizaje. Las respuestas fueron estructuradas en una escala de 1 a 7, donde 1 indica una fuerte discrepancia y 7 un fuerte acuerdo, evaluando así diferentes dimensiones de usabilidad.

A continuación, se presentan los resultados detallados:

Pregunta	Media	Desviación Estándar
1 - "Creo que me gustaría usar este sistema con frecuencia"	6.57	0.57
2 - "Encontré el sistema innecesariamente complejo"	1.71	0.69
3 - "Pienso que el sistema es fácil de usar"	6.71	0.49
4 - "Creo que necesitaría el apoyo de una persona técnica para poder utilizar este sistema"	1.14	0.38
5 - "Encontré que las diversas funciones de este sistema estaban bien integradas"	6.00	0.82
6 - "Pienso que había demasiada inconsistencia en este sistema"	1.14	0.38
7 - "Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente"	6.14	0.69
8 - "El sistema me pareció muy complicado de usar"	1.14	0.38
9 - "Me sentí muy seguro usando el sistema"	6.29	0.49
10 - "Necesité aprender muchas cosas antes de poder empezar a usar el sistema"	1.43	0.53

Tabla 3 Resultados de Encuestas 2

Los resultados indican que el sistema desarrollado ha recibido altas puntuaciones en facilidad de uso y seguridad percibida, con una baja percepción de complejidad por parte de los usuarios. Sin embargo, se observa la necesidad de mejorar la integración de funciones y asegurar la consistencia para todos los usuarios en trabajos futuros. Estos hallazgos no solo resaltan las fortalezas del sistema, sino que también señalan áreas específicas que podrían optimizarse para mejorar aún más la experiencia del usuario y cumplir con las expectativas operativas de la empresa.

8.2. Evaluación mediante Cuestionario UEQ

En esta subsección se presentan los resultados obtenidos a partir de la encuesta de Experiencia de Usuario (UEQ) aplicada a los usuarios del Sistema de Gestión de Incidentes y Administración de Servicios. La encuesta UEQ permite medir la percepción de los usuarios en relación a diferentes aspectos del sistema, tales como la atracción, claridad, eficiencia, precisión, estimulación y novedad [38].

Las preguntas de la encuesta UEQ se agrupan en diferentes dimensiones para proporcionar una evaluación exhaustiva de la experiencia del usuario. Estas dimensiones incluyen atracción, claridad, eficiencia, precisión, estimulación y originalidad. La Tabla 3 muestra cómo se distribuyeron las preguntas en cada escala:

Escala	Preguntas (Números)
Atracción	1, 12, 14, 24
Claridad	2, 8, 13, 21
Eficiencia	9, 20, 22, 23
Precisión	6, 11, 19
Estimulación	5, 18, 26
Originalidad	3, 7, 10, 15, 16, 17, 25

Tabla 4 Agrupación de Preguntas para UEQ

8.2.1. Recopilación de Respuestas:

Las respuestas se obtuvieron mediante el Cuestionario de Experiencia de Usuario (UEQ) aplicado a los usuarios del Sistema de Gestión de Incidentes y Administración de Servicios.

8.2.2. Inversión de Puntuaciones para Ítems en Escala Invertida:

Algunas preguntas del cuestionario usan una escala invertida (donde 1 es positivo y 7 es negativo). Estas preguntas son: 3, 6, 17.

Para estas preguntas, se invirtieron las puntuaciones utilizando la fórmula:

$$\text{puntuación invertida} = 8 - \text{puntuación original}$$

8.2.3. Cálculo de Promedios:

Para cada dimensión, se calculó el promedio de las respuestas ajustadas (incluyendo las invertidas) de los ítems correspondientes.

Conversión a la Escala de -3 a +3:

Las puntuaciones promedio obtenidas en la escala original (1 a 7) se convirtieron a la escala de -3 a +3 usando la fórmula:

$$\text{puntuación ajustada} = \frac{\text{puntuación promedio} - 4}{3}$$

Con este proceso, finalmente obtenemos las puntuaciones para cada dimensión, como se muestra en la Tabla 5.

Dimensión	Puntuación Promedio Ajustada
Atractividad	0.42
Claridad	0.53
Eficiencia	0.68
Precisión	0.53
Estimulación	0.60
Dependabilidad	0.33

Tabla 5 Puntuación para cada dimensión

Interpretación de la Escala de -3 a +3

La escala de -3 a +3 permite interpretar la percepción del usuario de manera más intuitiva:

- +1 a +3: La dimensión se percibe positivamente.
  - Ligeramente positivo: 0 a 1
  - Positivo: 1 a 2
  - Muy positivo: 2 a 3
- -1 a +1: La dimensión es neutral.
  - Ligeramente negativo: -1 a 0
  - Neutral: -0.5 a 0.5

- Ligeramente positivo: 0 a 1

- -3 a -1: La dimensión se percibe negativamente.
  - Muy negativo: -3 a -2
  - Negativo: -2 a -1
  - Ligeramente negativo: -1 a 0

Atractividad (0.42): Los usuarios encuentran el sistema ligeramente atractivo. Está por encima de la neutralidad, pero hay espacio para mejoras para que sea percibido como altamente atractivo.

Claridad (0.53): Los usuarios consideran que el sistema es bastante claro y entendible. Esto es un punto fuerte, aunque todavía se puede trabajar en mejorar la claridad para alcanzar una percepción muy positiva.

Eficiencia (0.68): La eficiencia del sistema es bien valorada por los usuarios, indicando que pueden realizar tareas con un esfuerzo razonable y de manera efectiva. Este es uno de los puntos más fuertes del sistema.

Precisión (0.53): Los usuarios perciben que el sistema es preciso y que responde adecuadamente a sus acciones. Al igual que con la claridad, esto es positivo, pero hay margen para mejorar.

Estimulación (0.60): El sistema es percibido como estimulante y motivador para los usuarios, lo cual es crucial para mantener el interés y la satisfacción a largo plazo.

Dependabilidad (0.33): Aunque los usuarios encuentran que el sistema es confiable, este es el área con la menor puntuación entre las dimensiones. Mejorar la confiabilidad podría tener un impacto significativo en la percepción general del sistema.

La tabla 8 proporciona un resumen detallado de los resultados implementados, destacando las principales métricas y hallazgos obtenidos durante el proceso de evaluación.

Dimensión	**Puntuación Promedio Ajustada**	**Interpretación**
Atractividad	0.42	Ligeramente atractivo
Claridad	0.53	Bastante claro y entendible
Eficiencia	0.68	Bien valorada en términos de eficiencia
Precisión	0.53	Percepción positiva de precisión
Estimulación	0.60	Percibido como estimulante y motivador

Dependabilidad	0.33	Considerado confiable, pero con margen de mejora

Tabla 6 Resumen de la interpretación UEQ

Los resultados obtenidos de las evaluaciones indican una percepción positiva del sistema por parte de los usuarios, pese al corto tiempo de interacción de aproximadamente dos semanas. Aunque los usuarios tuvieron un tiempo limitado para familiarizarse completamente con el sistema, las respuestas reflejan una tendencia inicial muy favorable. Es importante señalar que una evaluación más prolongada podría proporcionar resultados aún más específicos y detallados, ya que los usuarios tendrían la oportunidad de explorar y utilizar el sistema en mayor profundidad. Sin embargo, estos hallazgos preliminares sugieren que la herramienta ha sido bien recibida y cumple con las expectativas de los usuarios en esta primera instancia.

9. Conclusiones y Trabajos futuros

En esta tesis se desarrolló un Sistema de Gestión de Incidentes y Administración de Servicios para empresas de desarrollo de software, utilizando la metodología Rational Unified Process (RUP) como base. Este trabajo tuvo como objetivo resolver los procesos comunes en la gestión de requerimientos y soporte técnico, incrementando la eficiencia operativa y la satisfacción del cliente. A continuación, se presentan las conclusiones alcanzadas en relación con los objetivos planteados al inicio del proyecto.

Con respecto al objetivo específico “Analizar los requerimientos y procesos empresariales”, el aporte más importante de este trabajo de titulación es la definición de un proceso de negocio para la administración de los flujos de trabajo de requerimientos y soporte. Para alcanzar este objetivo, así como los demás objetivos, se contó con la participación activa de profesionales en el área de desarrollo de software que trabajan en la empresa Vimasistem. Este proceso no solo puede ser aplicado a la empresa Vimasistem sino a cualquier empresa que requiera usarlo como guía para sus actividades. Adicionalmente, en base a esta definición, este trabajo de titulación generó una especificación de requerimientos fundamental para el diseño adecuado de la solución. Esta especificación puede ser utilizada por cualquier empresa, equipo de desarrollo o grupo interesado como base para la construcción de sistemas de gestión de requisitos.

En cuanto al objetivo específico “Diseñar la Arquitectura del Sistema”, se diseñó una arquitectura modular y escalable, utilizando una estructura de paquetes refinada para organizar las clases y responsabilidades de manera coherente. Esta arquitectura soporta eficientemente las operaciones del sistema y facilita su mantenimiento y expansión futura, satisfaciendo los requisitos no funcionales que fueron identificados.

Para cumplir con el objetivo específico “Implementar el Sistema”, se implementaron las funcionalidades principales en base a la priorización llevada a cabo durante la especificación de requisitos. Se desarrollaron y probaron todas las funcionalidades clave, incluyendo la gestión de requerimientos y tickets de soporte, asegurando que el sistema cumpliera con los requisitos especificados y los estándares de calidad esperados. Es importante mencionar que durante las pruebas de aceptación participaron usuarios expertos en el desarrollo de software, así como los usuarios finales de la aplicación.

En relación con el objetivo específico de "Probar y validar empíricamente el sistema en un entorno de usuario final", se llevó a cabo una validación empírica del sistema mediante un experimento con evaluadores expertos, utilizando los cuestionarios de usabilidad (SUS) y experiencia de usuario (UEQ). Los resultados mostraron un alto nivel de satisfacción en cuanto a usabilidad, funcionalidad y eficiencia del sistema, confirmando que el enfoque integral basado en RUP es adecuado para la gestión de incidentes y servicios en el contexto empresarial del desarrollo de software.

Finalmente, en cuanto al cumplimiento del objetivo general “Analizar, diseñar, desarrollar e implementar un sistema de gestión de incidentes y administración de servicios para empresas de desarrollo de software aplicando la metodología de Rational Unified Process (RUP)”, se logró desarrollar e implementar un sistema funcional y robusto que integra de manera coherente todas las fases del ciclo de vida del software, desde la solicitud inicial hasta la resolución de incidentes, siguiendo las mejores prácticas de RUP. El sistema no solo cumplió con las expectativas iniciales, sino que también demostró ser una herramienta eficaz para mejorar la gestión de requerimientos y el soporte técnico en empresas de desarrollo de software.

Impacto Positivo en la Operación y Satisfacción del Cliente: El sistema desarrollado ha demostrado un notable incremento en la eficiencia operativa y la satisfacción del cliente, al ofrecer una herramienta centralizada y coherente para la gestión de incidentes y servicios.

Mitigación de Riesgos: La adopción del desarrollo iterativo permitió identificar y mitigar riesgos tempranamente, mejorando la adaptabilidad y la calidad del sistema.

Escalabilidad y Mantenibilidad: La arquitectura modular del sistema asegura su escalabilidad y facilidad de mantenimiento, permitiendo futuras ampliaciones y adaptaciones sin complicaciones significativas.

9.1. Trabajos futuros

Integración de Funcionalidades Adicionales:

Futuras iteraciones en el proceso RUP integrarán los procesos que no han sido cubiertos en esta primera iteración del proceso. Este trabajo de titulación ha aplicado un enfoque para el desarrollo que ha mostrado ser eficiente, sentando las bases para otras aplicaciones dentro

de la empresa y estableciendo un fundamento sólido para futuros desarrollos en empresas de software. Además, podrían explorarse la integración de funcionalidades adicionales, como la automatización de procesos y la integración con otras herramientas de gestión de proyectos.

Formación y Documentación:

La creación de manuales de usuario más detallados y la implementación de sesiones de formación adicionales podrían mejorar aún más la usabilidad del sistema y reducir la curva de aprendizaje para nuevos usuarios. Estas acciones asegurarán que todos los usuarios puedan aprovechar al máximo las capacidades del sistema desde el inicio, aumentando la eficiencia y la satisfacción general.

En conclusión, el trabajo realizado no solo cumple con los objetivos establecidos, sino que también proporciona una base sólida para futuras mejoras y ampliaciones, contribuyendo significativamente a la calidad y eficiencia de la gestión de incidentes y servicios en empresas de desarrollo de software.

Referencias

Angular—Introducción a la Documentación de Angular. (2024, iunius 17). https://docs.angular.lat/docs

Apache NetBeans Wiki. (2024, iunius 17). https://netbeans.apache.org/wiki/main/wiki/

Atlassian. (2024, iunius 18). Bienvenido a Jira. Atlassian. https://www.atlassian.com/es/software/jira/guides/getting-started/introduction

Brooke, john. (1996). SUS: A 'Quick and Dirty' Usability Scale. In Usability Evaluation In Industry. CRC Press.

Build an app with Asana. (2024, iunius 18). Asana Docs. https://developers.asana.com/

C++ documentation—DevDocs. (s.d.). Recuperatus 9 september 2024, ab https://devdocs.io/cpp/

Django documentation | Django documentation. (2024, iunius 18). Django Project. https://docs.djangoproject.com/en/5.0/

Documentation for Visual Studio Code. (2024, iunius 18). https://code.visualstudio.com/docs

dotnet-bot. (2024, iunius 17). .NET Framework documentation. https://learn.microsoft.com/en-us/dotnet/framework/

Eclipse Documentation | The Eclipse Foundation. (2024, iunius 17). Eclipse Foundation. https://www.eclipse.org/documentation/

Ferreira, J. M., & Acuña, S. T. (s.d.). Evaluación Empírica de los Mecanismos de Usabilidad: Efectos sobre la Eficiencia y la Eficacia del Usuario.

Freshworks Developer Docs | Freshworks app ecosystem. (2024, iunius 18). https://developers.freshworks.com/docs/app-sdk/v2.3/freshdesk/freshworks-app-ecosystem/

Getting started | IntelliJ IDEA Documentation. (2024, iunius 17). https://www.jetbrains.com/help/idea/getting-started.html

Getting started | PyCharm. (2024, iunius 17). PyCharm Help. https://www.jetbrains.com/help/pycharm/getting-started.html

Getting Started – React. (2024, iunius 18). https://legacy.reactjs.org/docs/getting-started.html

Getting started with WebStorm | WebStorm Documentation. (2024, iunius 17). https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html

ghogen. (2024, iunius 17). Visual Studio documentation. https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022

HTML: HyperText Markup Language | MDN. (2024, aprilis 25). https://developer.mozilla.org/en-US/docs/Web/HTML

Inicio | Vimasistem. (2024, iunius 20). https://www.vimasistem.com/

Introduction to YouTrack | YouTrack Server. (s.d.). YouTrack Server Help. Recuperatus 9 september 2024, ab https://www.jetbrains.com/help/youtrack/server/introduction-to-youtrack-server.html

Java Documentation. (2024, iunius 18). Oracle Help Center. https://docs.oracle.com/en/java/

JavaScript | MDN. (2024, martius 5). https://developer.mozilla.org/en-US/docs/Web/JavaScript

Kallmuenzer, A., Mikhaylov, A., Chelaru, M., & Czakon, W. (2024). Adoption and performance outcome of digitalization in small and medium-sized enterprises. Review of Managerial Science. https://doi.org/10.1007/s11846-024-00744-2

Kotlin Docs | Kotlin. (2024, iunius 18). Kotlin Help. https://kotlinlang.org/docs/home.html

Kruchten, P. (2004). The Rational Unified Process: An Introduction. Addison-Wesley Professional.

Lagerburg, R. A. (2023, augustus 5). Overcoming Challenges in Software Development Projects: An Integrated Approach for Small and Medium Enterprises (SMEs) [Info:eu-repo/semantics/masterThesis]. University of Twente. https://essay.utwente.nl/96674/

Learn Scala. (2024, iunius 18). Scala Documentation. https://docs.scala-lang.org/

LogMeIn Rescue API User Guide – Overview of the LogMeIn Rescue API. (s.d.). [Concept]. Recuperatus 9 september 2024, ab

https://secure.logmeinrescue.com/welcome/webhelp/en/rescueapi/API/API_Rescue_Overview.html

Managing Information Technology | SpringerLink. (2024, iunius 18). https://link.springer.com/book/10.1007/978-3-031-39016-6

Nunes, M. P., & Russo, A. P. (2019). Analysis of business models innovation – a multiple case study. Innovation & Management Review, 16(1), Article 1. https://doi.org/10.1108/INMR-11-2018-0085

Olson, D., & Kesharwani, S. (2010). Enterprise Information System Trends. 73, 3–14. https://doi.org/10.1007/978-3-642-19802-1_1

Osterwalder, A., Pigneur, Y., & Clark, T. (2010). Business model generation: A handbook for visionaries, game changers, and challengers. Wiley.

PHP: Documentation. (2024, iunius 20). https://www.php.net/docs.php

Product Documentation | ServiceNow. (s.d.). Recuperatus 9 september 2024, ab https://docs.servicenow.com/

Rêgo, B. S., Jayantilal, S., Ferreira, J. J., & Carayannis, E. G. (2022). Digital Transformation and Strategic Management: A Systematic Review of the Literature. Journal of the Knowledge Economy, 13(4), Article 4. https://doi.org/10.1007/s13132-021-00853-3

Ries, E. (2017). The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Currency.

Sharma, R., Madireddy, V., Jain, V., & Apoorva, S. R. (2008). Best Practices for Communication between Client and Vendor in IT Outsourcing Projects (SSRN Scholarly Paper No. 1258636; Numerus 1258636). https://papers.ssrn.com/abstract=1258636

Silvestro, A. R., Althof, R. B., & Varvakis, G. J. (s.d.). CHALLENGES IN IMPLEMENTING ORGANIZATIONAL KNOWLEDGE MANAGEMENT: AN INTEGRATIVE REVIEW.

Slack platform overview | Slack. (2024, iunius 18). https://api.slack.com/docs

Soveizi, N., Turkmen, F., & Karastoyanova, D. (2023). Security and privacy concerns in cloud-based scientific and business workflows: A systematic review. Future Generation Computer Systems, 148, 184–200. https://doi.org/10.1016/j.future.2023.05.015

Trello Guides: Help Getting Started With Trello | Trello. (2024, iunius 18). https://trello.com/guide

Trenkle, J. (2020). Digital Transformation in Small and Medium-Sized Enterprises: Strategy, Management Control, and Network Involvement. Nomos Verlagsgesellschaft mbH & Co. KG. https://doi.org/10.5771/9783748922131

User Exper*ience Questionnaire (UEQ)*. (2024, iunius 17). https://www.ueq-online.org/

Vue.js. (2024, iunius 20). https://vuejs.org/

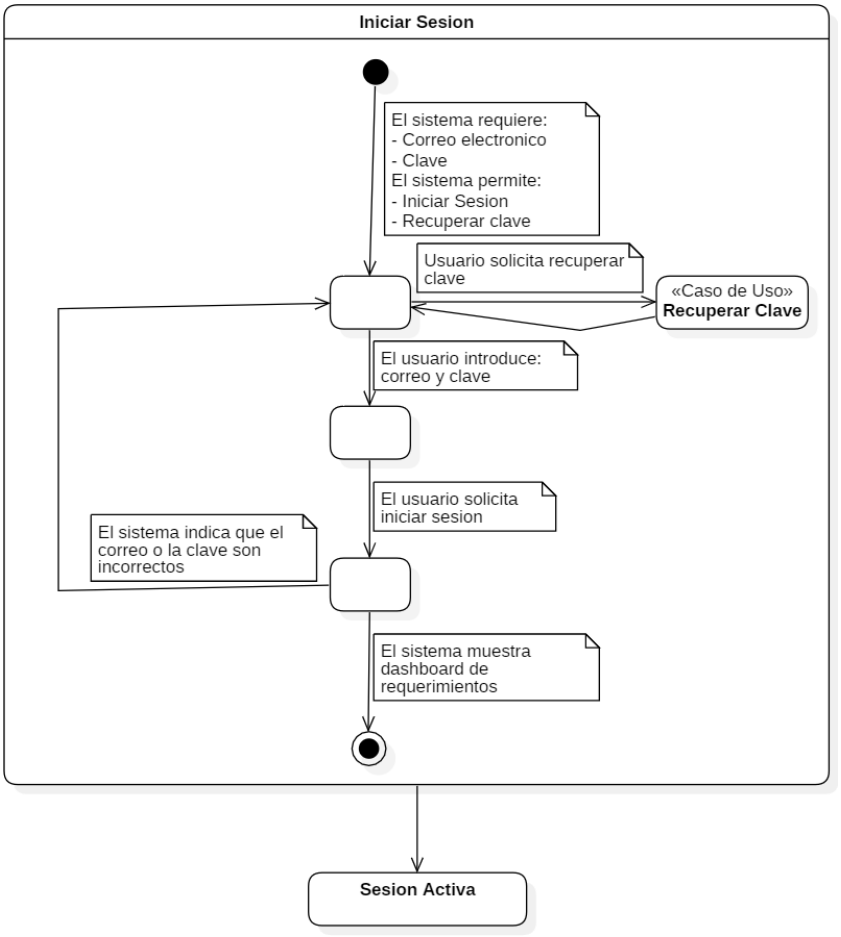
Welcome to Flask—Flask Documentation (3.0.x). (2024, iunius 18). https://flask.palletsprojects.com/en/3.0.x/

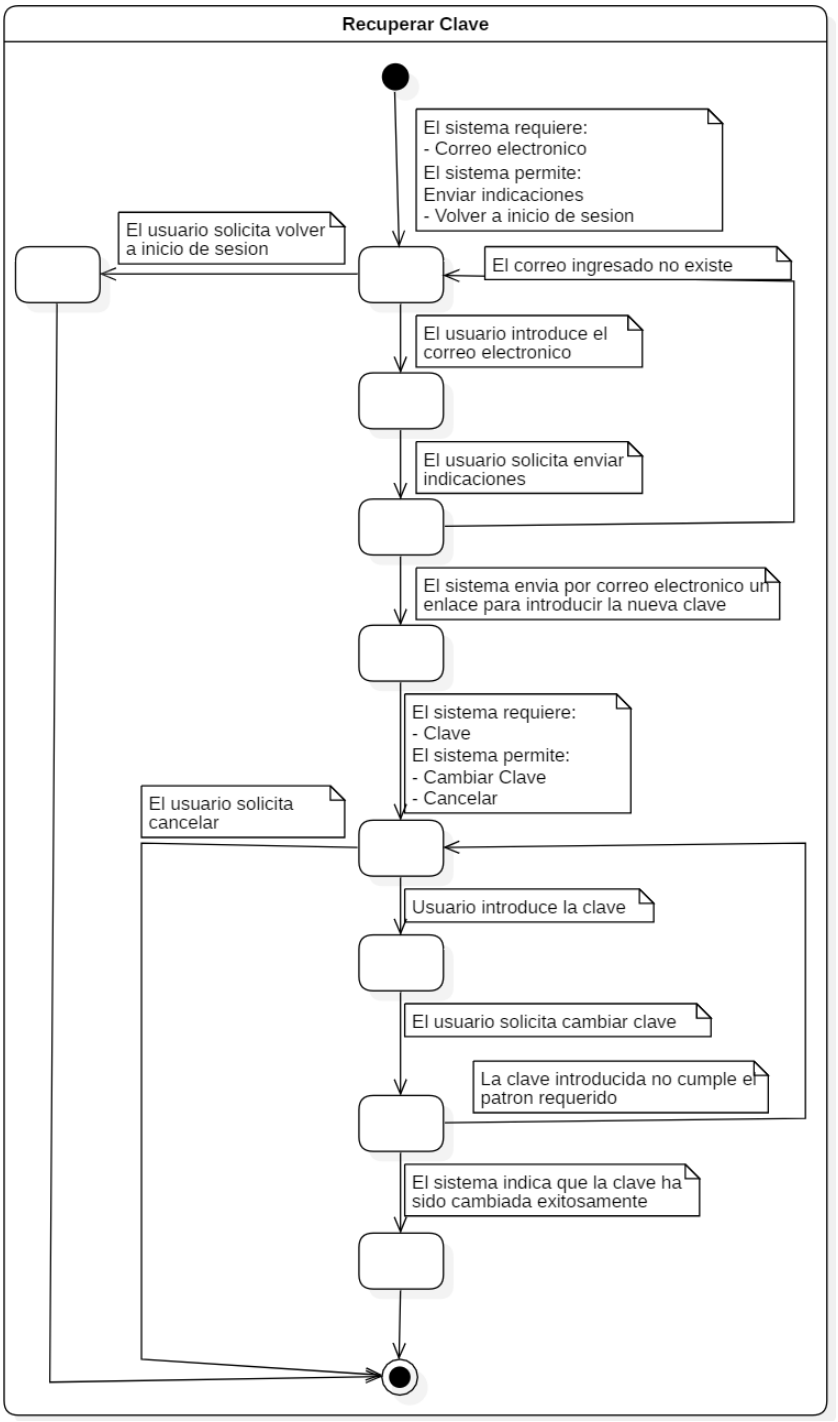
Welcome to Python.org. (2024, september 7). Python.Org. https://www.python.org/

What is a REST API? (2024, iunius 18). https://www.redhat.com/en/topics/api/what-is-a-rest-api

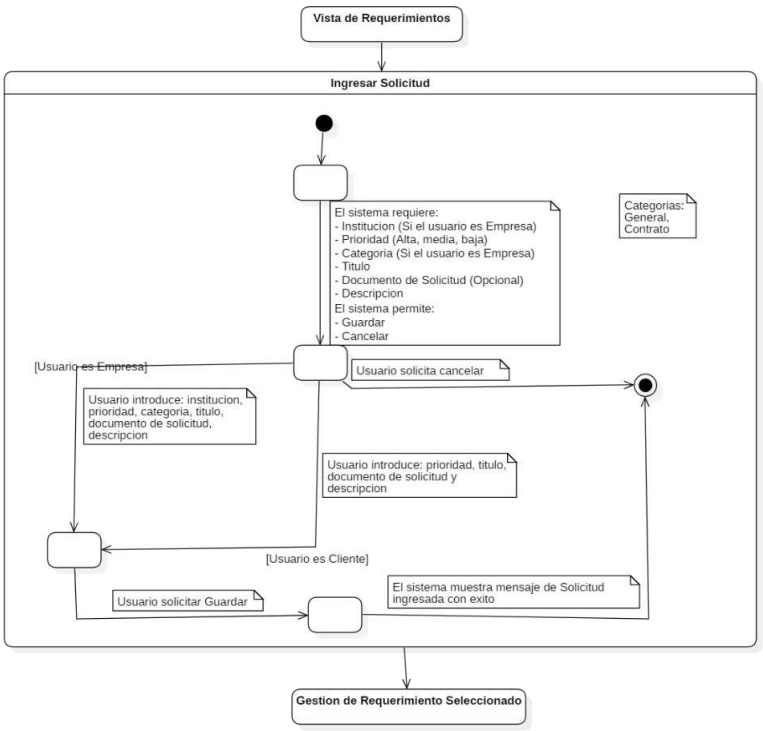
Anexos

Anexo A

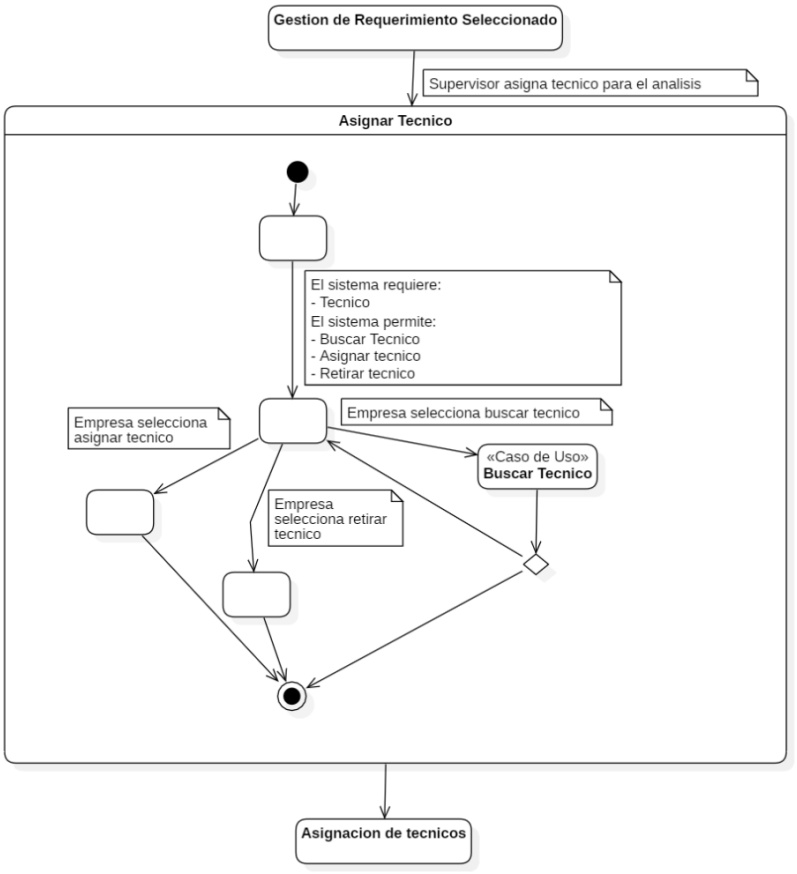



Recuperar Clave
El sistema requiere:
- Correo electronico
El sistema permite:
Enviar indicaciones
- Volver a inicio de sesion
El usuario solicita volver a inicio de sesion
El correo ingresado no existe
El usuario introduce el correo electronico
El usuario solicita enviar indicaciones
El sistema envia por correo electronico un enlace para introducir la nueva clave
El sistema requiere:
- Clave
El sistema permite:
- Cambiar Clave
- Cancelar
El usuario solicita cancelar
Usuario introduce la clave
El usuario solicita cambiar clave
La clave introducida no cumple el patron requerido
El sistema indica que la clave ha sido cambiada exitosamente

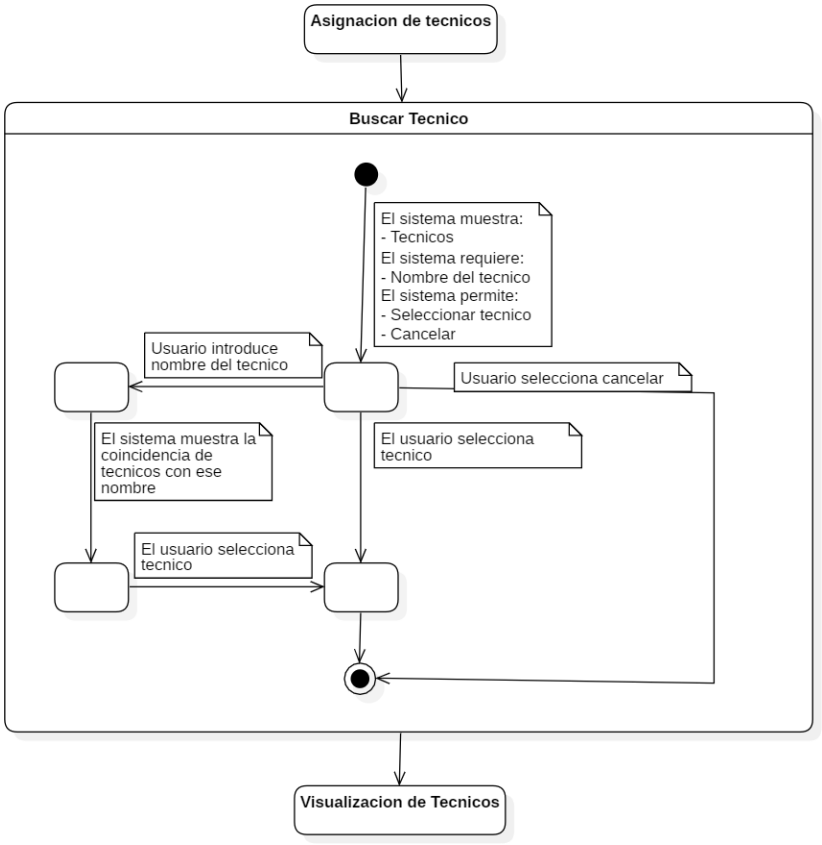
Anexo B



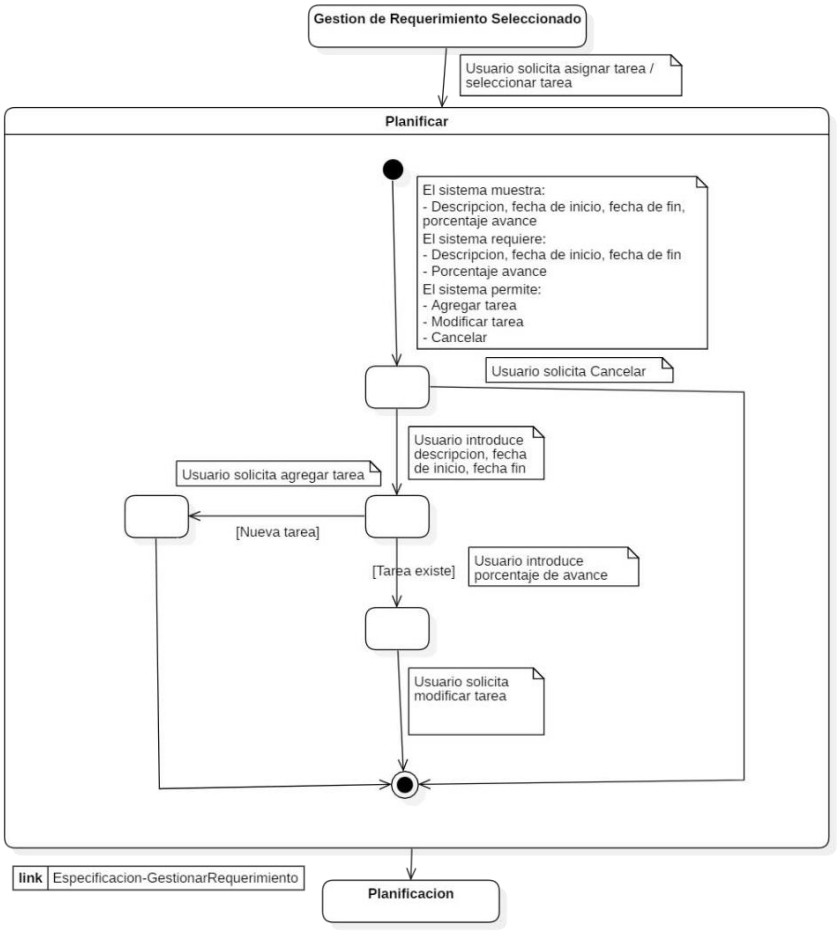
Anexo C



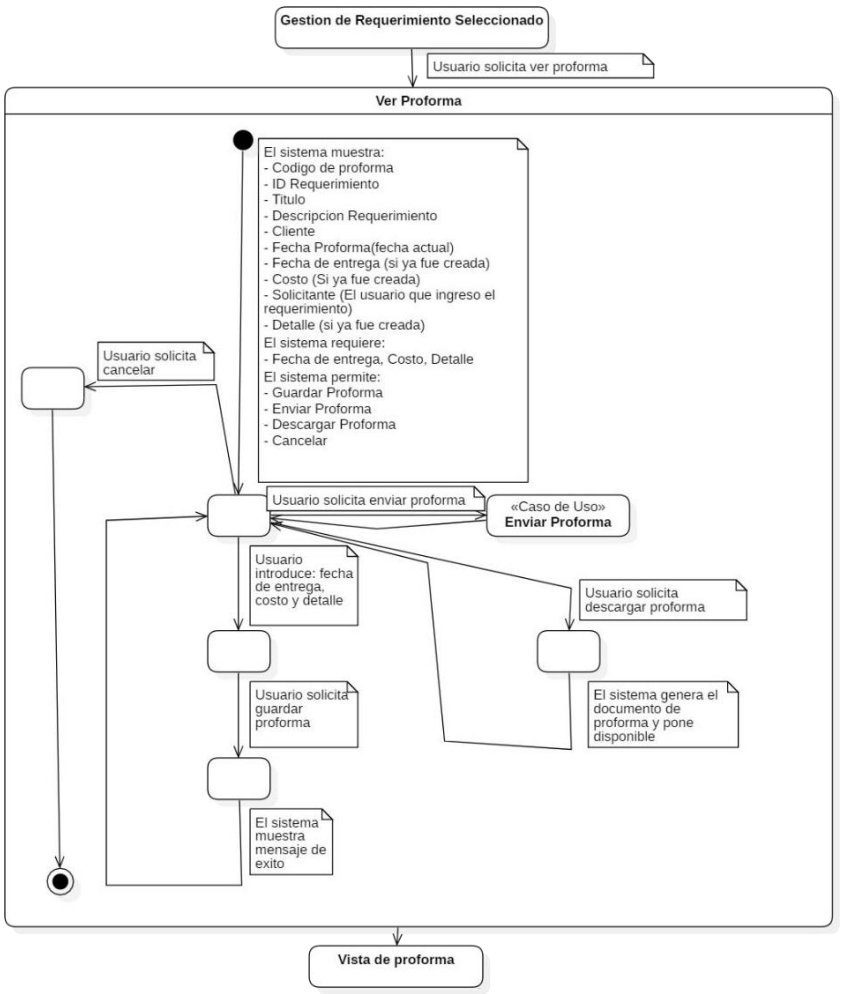
Anexo D


Asignacion de tecnicos
Buscar Tecnico
El sistema muestra:
- Tecnicos
El sistema requiere:
- Nombre del tecnico
El sistema permite:
- Seleccionar tecnico
- Cancelar
Usuario introduce nombre del tecnico
Usuario selecciona cancelar
El sistema muestra la coincidencia de tecnicos con ese nombre
El usuario selecciona tecnico
El usuario selecciona tecnico
Visualizacion de Tecnicos

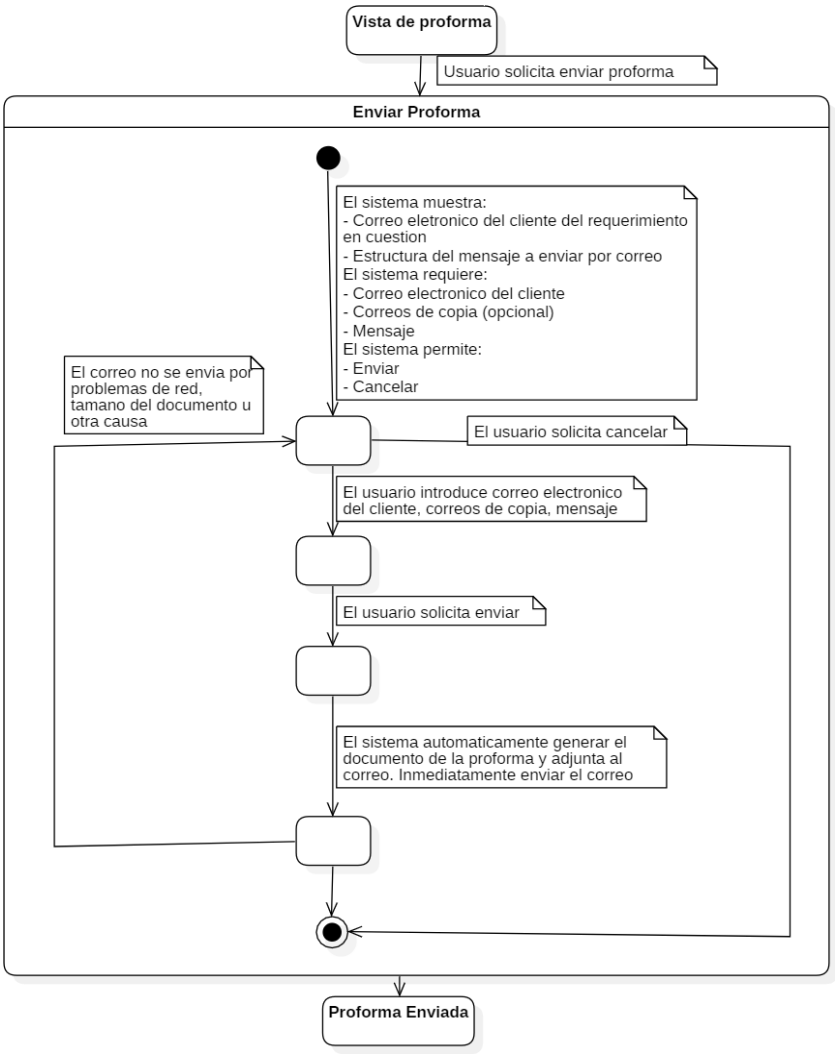
Anexo E


Gestion de Requerimiento Seleccionado
Usuario solicita asignar tarea / seleccionar tarea
Planificar
El sistema muestra:
- Descripcion, fecha de inicio, fecha de fin, porcentaje avance
El sistema requiere:
- Descripcion, fecha de inicio, fecha de fin
- Porcentaje avance
El sistema permite:
- Agregar tarea
- Modificar tarea
- Cancelar
Usuario solicita Cancelar
Usuario introduce descripcion, fecha de inicio, fecha fin
Usuario solicita agregar tarea
[Nueva tarea]
[Tarea existe]
Usuario introduce porcentaje de avance
Usuario solicita modificar tarea
link
Especificacion-GestionarRequerimiento
Planificacion

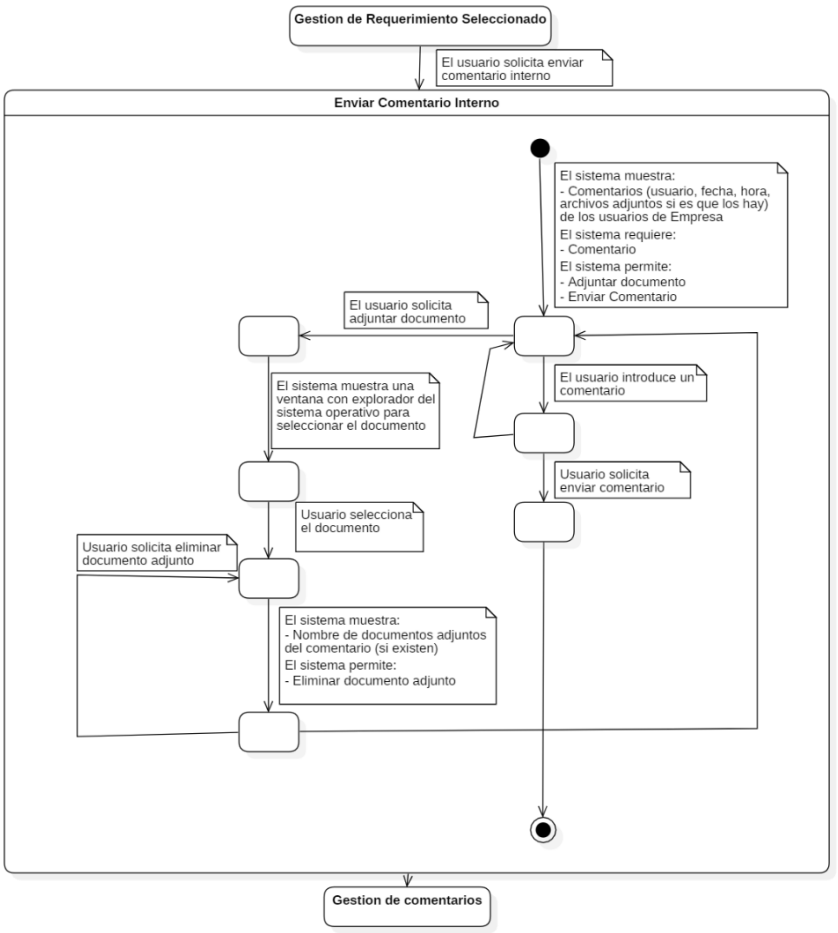
Anexo F


Gestion de Requerimiento Seleccionado
Usuario solicita ver proforma
Ver Proforma
El sistema muestra:
- Codigo de proforma
- ID Requerimiento
- Titulo
- Descripcion Requerimiento
- Cliente
- Fecha Proforma(fecha actual)
- Fecha de entrega (si ya fue creada)
- Costo (Si ya fue creada)
- Solicitante (El usuario que ingreso el requerimiento)
- Detalle (si ya fue creada)
El sistema requiere:
- Fecha de entrega, Costo, Detalle
El sistema permite:
- Guardar Proforma
- Enviar Proforma
- Descargar Proforma
- Cancelar
Usuario solicita cancelar
Usuario solicita enviar proforma
«Caso de Uso»
Enviar Proforma
Usuario introduce: fecha de entrega, costo y detalle
Usuario solicita descargar proforma
Usuario solicita guardar proforma
El sistema genera el documento de proforma y pone disponible
El sistema muestra mensaje de exito
Vista de proforma

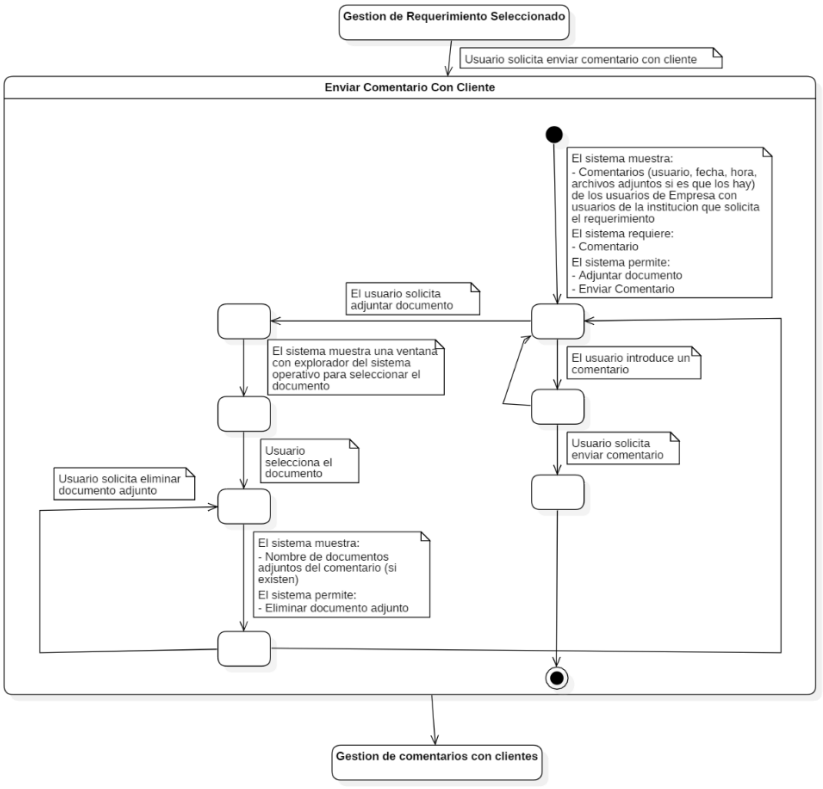
Anexo G


Vista de proforma
Usuario solicita enviar proforma
Enviar Proforma
El sistema muestra:
- Correo eletronico del cliente del requerimiento en cuestion
- Estructura del mensaje a enviar por correo
El sistema requiere:
- Correo electronico del cliente
- Correos de copia (opcional)
- Mensaje
El sistema permite:
- Enviar
- Cancelar
El correo no se envia por problemas de red, tamano del documento u otra causa
El usuario solicita cancelar
El usuario introduce correo electronico del cliente, correos de copia, mensaje
El usuario solicita enviar
El sistema automaticamente generar el documento de la proforma y adjunta al correo. Inmediatamente enviar el correo
Proforma Enviada

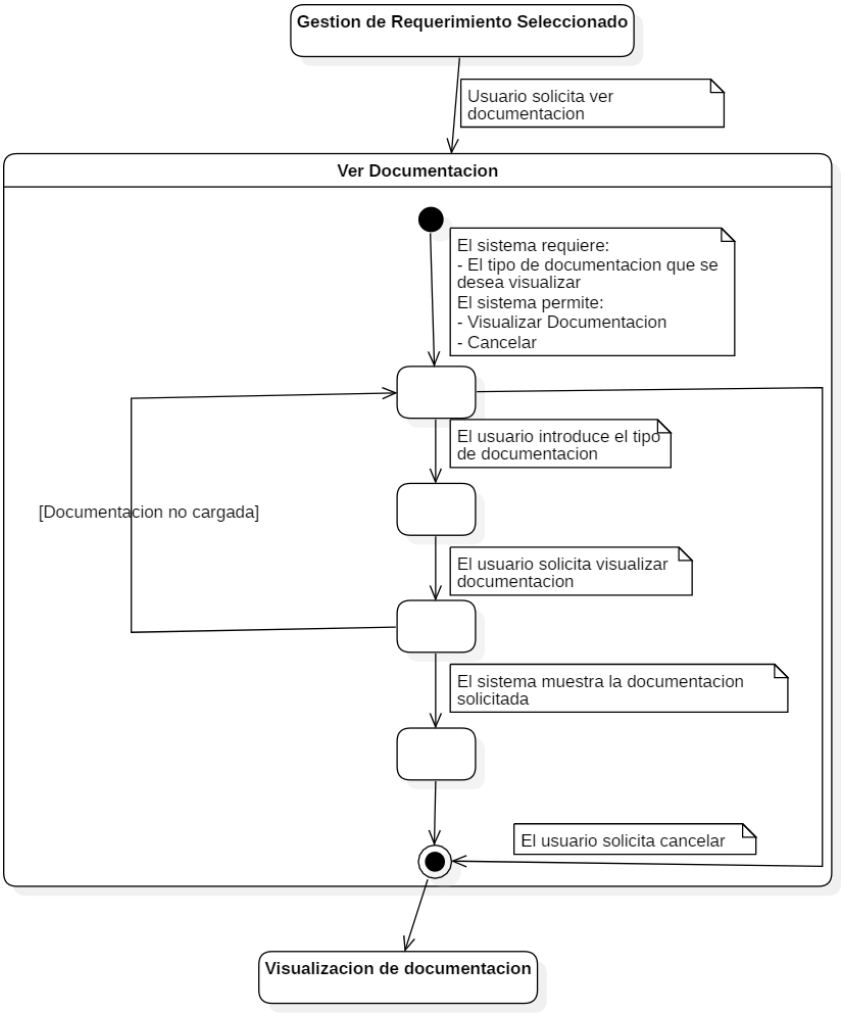
Anexo H


Gestion de Requerimiento Seleccionado
El usuario solicita enviar comentario interno
Enviar Comentario Interno
El sistema muestra:
- Comentarios (usuario, fecha, hora, archivos adjuntos si es que los hay) de los usuarios de Empresa
El sistema requiere:
- Comentario
El sistema permite:
- Adjuntar documento
- Enviar Comentario
El usuario solicita adjuntar documento
El sistema muestra una ventana con explorador del sistema operativo para seleccionar el documento
El usuario introduce un comentario
Usuario solicita enviar comentario
Usuario selecciona el documento
Usuario solicita eliminar documento adjunto
El sistema muestra:
- Nombre de documentos adjuntos del comentario (si existen)
El sistema permite:
- Eliminar documento adjunto
Gestion de comentarios

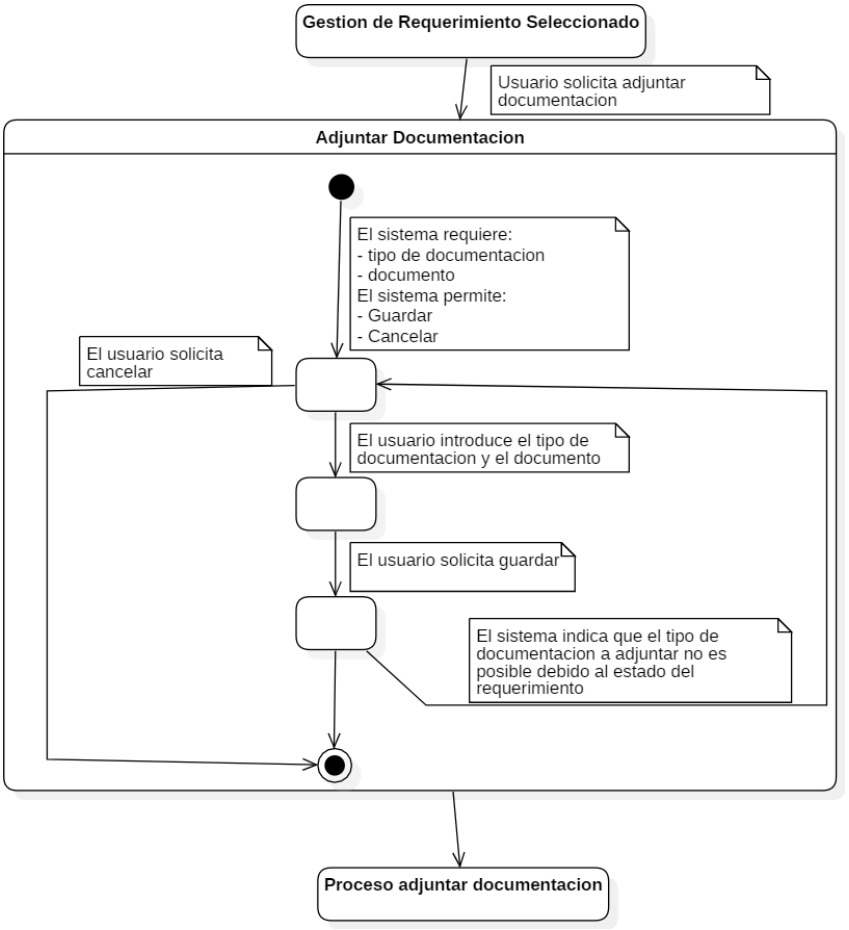
Anexo I


Gestion de Requerimiento Seleccionado
Usuario solicita enviar comentario con cliente
Enviar Comentario Con Cliente
El sistema muestra:
- Comentarios (usuario, fecha, hora, archivos adjuntos si es que los hay) de los usuarios de Empresa con usuarios de la institucion que solicita el requerimiento
El sistema requiere:
- Comentario
El sistema permite:
- Adjuntar documento
- Enviar Comentario
El usuario solicita adjuntar documento
El sistema muestra una ventana con explorador del sistema operativo para seleccionar el documento
El usuario introduce un comentario
Usuario solicita enviar comentario
Usuario selecciona el documento
Usuario solicita eliminar documento adjunto
El sistema muestra:
- Nombre de documentos adjuntos del comentario (si existen)
El sistema permite:
- Eliminar documento adjunto
Gestion de comentarios con clientes

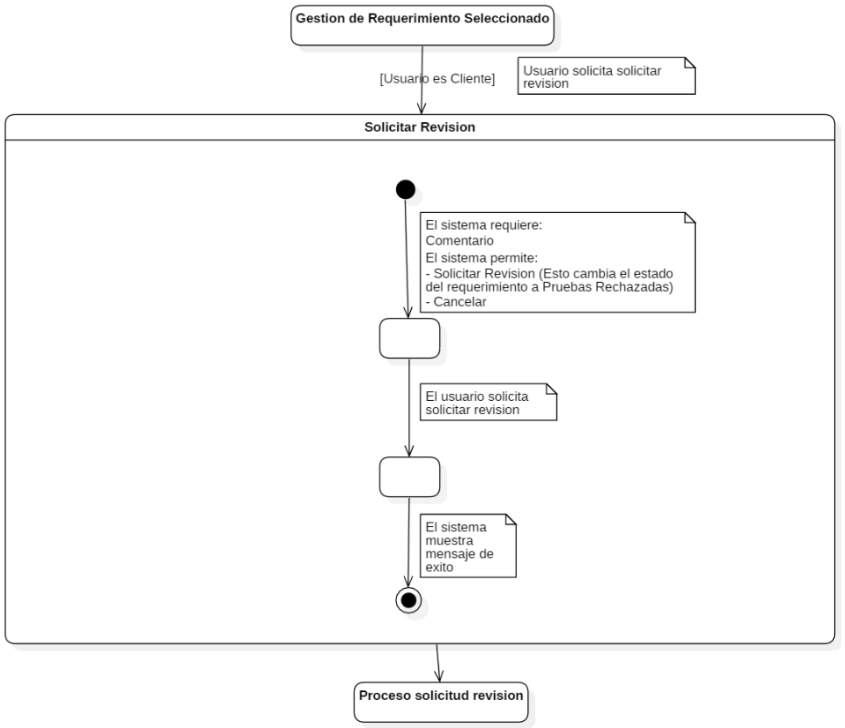
Anexo J



Anexo K



Anexo L



Anexo M

Encuesta de usabilidad

Proyecto de grado

Sistema de Gestión de Incidentes y Administración de Servicios

En una escala del 1 (Totalmente en desacuerdo) al 7 (Totalmente de acuerdo), indique su nivel de satisfacción en base a los siguientes enunciados:

1. Creo que me gustaría usar este sistema con frecuencia

Totalmente en desacuerdo								Totalmente de acuerdo

2. Encontré el sistema innecesariamente complejo

Totalmente en desacuerdo								Totalmente de acuerdo

3. Pienso que el sistema es fácil de usar

Totalmente en desacuerdo								Totalmente de acuerdo

4. Creo que necesitaría el apoyo de una persona técnica para poder utilizar este sistema

Totalmente en desacuerdo								Totalmente de acuerdo

5. Encontré que las diversas funciones de este sistema estaban bien integradas

Totalmente en desacuerdo								Totalmente de acuerdo

6. Pienso que había demasiada inconsistencia en este sistema

Totalmente en desacuerdo								Totalmente de acuerdo

7. Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente

Totalmente en desacuerdo								Totalmente de acuerdo

8. El sistema me pareció muy complicado de usar

Totalmente en desacuerdo								Totalmente de acuerdo

9. Me sentí muy seguro usando el sistema

Totalmente en desacuerdo								Totalmente de acuerdo

10. Necesite aprender muchas cosas antes de poder empezar a usar el sistema

Totalmente en desacuerdo								Totalmente de acuerdo

Anexo N

Cuestionario de Experiencia de Usuario

Proyecto de grado

Sistema de Gestión de Incidentes y Administración de Servicios

Por favor rellene el siguiente cuestionario con el fin de evaluar el sistema. El cuestionario consta de pares de propiedades opuestas que podrían aplicar al sistema. Para expresar su conformidad con una propiedad usted puede seleccionar una de las 7 casillas que mejor refleje su opinión en cada pregunta.

Ejemplo:

24	Atractivo		X					No atractivo

Esta respuesta significa que usted califica a la aplicación como más atractiva que no atractiva.

Por favor, decida espontáneamente. No piense demasiado su opinión y asegúrese que exprese su impresión inicial.

Por favor, marque una casilla para cada par de propiedades, aunque piense que no son aplicables o que hay propiedades parecidas o prácticamente iguales. Su opinión personal cuenta.

Recuerde: ¡no hay respuesta correcta o incorrecta!

Nro		1	2	3	4	5	6	7	
1	Desagradable								Agradable
2	No entendible								Entendible
3	Creativo								Sin imaginación
4	Fácil de aprender								Difícil de aprender
5	Valioso								De poco valor
6	Aburrido								Emocionante
7	No interesante								Interesante
8	Impredecible								Predecible
9	Rápido								Lento
10	Creativo								Convencional

11	Obstructivo								Impulsor de apoyo
12	Bueno								Malo
13	Complicado								Fácil
14	Repeler								Atraer
15	Habitual								Vanguardista
16	Incómodo								Cómodo
17	Seguro								Inseguro
18	Motivante								Desalentador
19	Cumple las expectativas								No cumple las expectativas
20	Ineficiente								Eficiente
21	Claro								Confuso
22	Impráctico								Práctico
23	Ordenado								Sobrecargado
24	Atractivo								No atractivo
25	Intuitivo								Difícil de usar
26	Conservador								Innovador

Anexo O

Resultados de aplicación del cuestionario de experiencia de usuario (UEQ) del Sistema de Gestión de Incidentes y Administración de Servicios									
Preguntas	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5	Sujeto 6	Sujeto 7	Sujeto 8	Sujeto 9
1	6	7	7	7	7	7	7	6	1
2	6	7	7	7	7	7	7	6	7
3	6	1	1	2	1	1	1	2	6
4	6	1	1	1	1	1	1	2	7
5	7	1	1	1	1	1	1	1	7
6	5	6	6	6	4	7	4	5	4
7	6	7	7	6	5	7	7	6	3
8	6	6	7	6	7	7	7	2	7
9	6	1	1	1	1	1	1	1	2
10	6	2	1	1	1	1	1	3	4

11	6	6	7	6	6	7	7	6	6
12	6	1	1	1	1	1	1	2	1
13	6	7	7	7	6	7	7	7	7
14	6	6	7	6	6	7	4	6	4
15	6	6	1	6	6	7	6	3	5
16	6	6	7	6	7	7	6	6	5
17	6	6	1	1	1	1	1	2	2
18	6	1	2	1	4	1	1	2	4
19	6	2	1	1	1	1	1	3	1
20	6	6	7	6	7	7	7	6	7
21	6	1	1	1	1	1	3	2	1
22	6	7	7	7	7	7	7	6	7
23	6	1	1	1	1	1	1	2	2
24	6	1	1	1	1	1	1	2	2
25	1	1	1	1	1	1	1	2	1
26	6	7	7	7	7	7	6	5	6

Anexo P

Resultados de aplicación de la encuesta de usabilidad (USU) del Sistema de Gestión de Incidentes y Administración de Servicios									
	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5	Sujeto 6	Sujeto 7	Sujeto 8	Sujeto 9
1. Creo que me gustaría usar este sistema con frecuencia	7	7	7	7	6	5	7	6	7

2. Encontré el sistema innecesariamente complejo	1	1	1	1	6	2	2	1	1
3. Pienso que el sistema es fácil de usar	7	7	7	7	7	7	5	6	7
4. Creo que necesitaría el apoyo de una persona técnica para poder utilizar este sistema	2	1	1	2	7	1	1	1	1
5. Encontré que las diversas funciones de este sistema estaban bien integradas	7	7	7	7	6	5	5	6	6
6. Pienso que había demasiada	1	1	1	1	1	2	2	1	2

inconsistencia en este sistema									
7. Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente	7	7	6	7	6	7	6	6	7
8. El sistema me pareció muy complicado de usar	1	1	1	1	1	1	1	1	1
9. Me sentí muy seguro usando el sistema	7	7	7	7	6	7	5	6	6
10. Necesite aprender muchas cosas antes de poder empezar a	2	1	1	2	2	1	1	1	1

usar el sistema									