UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
Ingenieros de Sistemas

TEMA:
ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA "GESTIÓN DE ACTIVOS DE TI" REQUERIDA POR LA EMPRESA GARANCHECK. CÍA. LTDA

AUTORES:
CARMEN AMELIA CÁCERES COTACACHI
CRISTIAN GEOVANNY CAZARES BALDEÓN

TUTOR:
ARÉVALO CAMPOS ALONSO RENÉ

Quito, marzo del 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Carmen Amelia Cáceres Cotacachi, con documento de identificación N° 171934094-3 y Cristian Geovanny Cazares Baldeón con documento de identificación N° 172289883-8, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA "GESTIÓN DE ACTIVOS DE TI" REQUERIDA POR LA EMPRESA GARANCHECK. CÍA. LTDA, mismo que ha sido desarrollado para optar por el título de: INGENIEROS DE SISTEMAS, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en digital a la Biblioteca de la Universidad Politécnica Salesiana.

CARMEN AMELIA
CÁCERES COTACACHI
C.I: 171934094-3

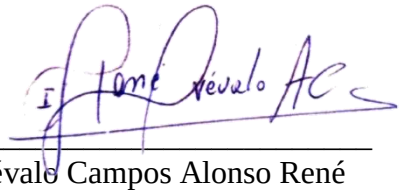
CRISTIAN GEOVANNY
CAZARES BALDEÓN
C.I: 172289883-8

Quito, marzo del 2021

DECLARATORIA DE COAUTORIA

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA "GESTIÓN DE ACTIVOS DE TI" REQUERIDA POR LA EMPRESA GARANCHECK. CÍA. LTDA, realizado por Cristian Geovanny Cazares Baldeón y Carmen Amelia Cáceres Cotacachi, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

Quito, marzo del 2021



Arévalo Campos Alonso René
C.I: 140016489-1

DEDICATORIA

El presente trabajo de Tesis está dedicada a mis Padres, quienes hicieron posible que llegue al culmino de mi carrera. Mi eterna gratitud y consagración a Ana Lucía mi madre, por todos los sacrificios que realizó por mí, esos gestos jamás se borrarán de mi mente ya que, si logré hoy finalizar con éxito mis estudios profesionales, llegar a tener una carrera y un futuro próspero, se lo debo a esta mujer incansable, que a ningún momento me dio la espalda, conté con su apoyo incondicional y estuvo siempre firme y cuando sentía miedo o decaimiento, me dio las fuerzas necesarias para salir adelante. Bayron padre mío, gracias por estar con mi hija cuidarla, mimarla y sobre todo ayudarme para que tenga siempre el tiempo que debía yo, dedicarles a mis estudios.

Luciana, a ti ya que eres la razón de mi existencia por quien lucho cada día y por haberte quitado el tiempo que necesitabas para realizarme profesionalmente.

A Felipe, mi hermano, por ser una muestra fehaciente de que uno con esfuerzo y dedicación logra llegar a donde se propone. Por ser esa persona que me brinda todos sus conocimientos y experiencia, y así, poder aprovecharlos en mi vida profesional y personal.

Y como no mencionar a mi amigo, Cristian, quién hizo que toda la travesía universitaria este llena de risas y momentos amenos, con quién en cada semestre nos dábamos la mano uno a al otro para seguir adelante y llegar al fin de esta etapa.

Carmen Amelia Cáceres.

Dedicado el presente trabajo a Dios, por haberme bendecido y guiado en toda mi carrera universitaria y por haberme dado la oportunidad de culminar mi añorado título universitario, a pesar de los buenos y malos momentos que tuve en el transcurso de mis estudios. Siempre sentí que fue mi guía en todo.

A mi Madrecita Martha Baldeón, quien, gracias a su esfuerzo, amor y su constante apoyo en todos los momentos de mi vida, y por ser el mayor impulso para lograr la culminación de mi carrera universitaria y de todos mis sueños.

A mi Padre Stalyn Cazares, quien, con sabiduría, esfuerzo guio mi vida para saber luchar por mis sueños, sin su ayuda no hubiese alcanzado mi título universitario.

A mi Hermana Belén, quien gracias a su admiración y amor me impulso a la culminación de esta nueva etapa de mi vida.

A mis amados hijos; Matthew y Kristen, quienes son el motor de mi vida y siempre me dieron la fuerza para no dar un paso atrás en mis estudios, son la razón de mi vida e inspiración para mi superación personal.

A mi futura Esposa Samantha Mosquera, quien es mi compañera de vida siempre dándome fuerzas en todo momento, haciéndome ver que la vida es mejor cuando tienes a una persona que te motiva y admira. Gracias a su amor, apoyo incondicional y constante preocupación me alentaron a terminar mi trabajo de tesis.

A mis familiares y amigos, quienes siempre me dieron una lección de vida en toda esta etapa, gracias por todos los gratos momentos que vivimos juntos durante mi carrera universitaria.

A mi amiga Amelia Cáceres, quien sin su valiosa amistad toda mi carrera universitaria no hubiese tenido esos lindos momentos y juntos pudimos realizar el trabajo de tesis.

Cristian Geovanny Cazares Baldeón

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto de titulación se enfoca en el desarrollo de un sistema informático web, para la gestión de activos de TI, en la empresa Garancheck. Este sistema busca agilizar los procesos de custodio, entrega o distribución y control del estado físico de todos los activos de TI que dispone la empresa. Además, toda la información relacionada con los cambios de estados que atraviesen los activos TI serán notificados vía correo electrónico al respectivo jefe de TI para conocimiento de dicho cambio y que sirva de respaldo al departamento encargado del custodio de dichos activos. El presente documento está organizado en capítulos. En el primer capítulo se describe el planeamiento del problema, objetivos y alcance de la propuesta del proyecto de tesis; además el marco teórico. El segundo capítulo puntualiza el análisis y el diseño de los requerimientos funcionales del sistema. El tercer capítulo detalla el proceso de construcción y pruebas del software desarrollado. Finalmente, se describen las conclusiones y recomendaciones.

ABSTRACT

This degree project focuses on the development of a computer system on the WEB for the management of IT assets, in the company Garancheck. This system seeks to streamline the custodian, delivery or distribution process and control of the physical state of all IT assets available to the company. In addition, all the information related to the changes of status that the IT assets go through will be notified via email to the respective Head of Area for knowledge of the same and serve as support to the department in charge of the custodian of said assets.

This document is organized into chapters. In the first chapter, the planning of the problem, objectives and scope of the thesis project proposal are described; also, the theoretical framework. The second chapter details the analysis and design of the functional requirements of the system. In the third chapter, the process of construction and testing of the developed software is described and finally the conclusions and recommendations are described.

INTRODUCCIÓN

Antecedentes

Garancheck Cía. Ltda. es una empresa ecuatoriana en crecimiento desde el 2014, que se abre paso en el área de desarrollo de software y otros giros, en la actualidad lleva el proceso de manejo de inventario de activos de TI de manera manual, el proceso comienza con el ingreso al almacén del activo de TI y posteriormente pasará a las etapas de asignación de responsable, bodegaje, bajas, reasignaciones, préstamos y demás temas de mantenimiento que el activo requiere. El registro de datos de Excel, en donde se almacena la información, contiene alrededor de 300 registros, entre los cuales se encuentran registros duplicados, los mismos que están dispersos por ubicación y por áreas. Las personas que se encargan de este proceso deben actualizar, guardar y enviar la base de datos Excel. La complejidad de tener los datos en Excel es que no tenemos en tiempo real la información y como el Excel es manejado por diferentes personas de la empresa, la información sobre los activos de TI no es confiable, ya que la información se arriesga a estar desactualizada, otro problema de trabajar de esta forma es que no todas las áreas que necesitan esta información la tienen de inmediato y retrasa el proceso del área de trabajo como por ejemplo: si el departamento de finanzas necesita el total de equipos portátiles de la empresa para realizar una estimación de gastos debería esperar para obtener el registro de Excel final para realizar la estimación, por ende se necesita la automatización y centralización de este proceso de control de activos de TI.

Planteamiento del Problema

Garancheck Cía. Ltda. es una empresa ecuatoriana en crecimiento desde el 2014, que se abre paso en el negocio de desarrollo de software y otros giros, en la actualidad lleva el proceso de manejo de inventario de activos de TI de manera manual, lo que hace que este proceso empiece por el

ingreso al almacén y posteriormente pasará por etapas de, asignación de responsable, bodegaje, bajas, reasignaciones, préstamos y demás temas de mantenimiento que el activo requiere.

En la base de datos Excel en donde se mantiene esta información, contiene alrededor de 300 registros, entre los cuales se encuentran registros repetidos, los mismos que están dispersos por ubicación y por subárea.

Las personas que se encargan de este proceso deben actualizar, guardar y enviar la base de datos Excel. El problema de tener los datos en Excel es que no tenemos a tiempo real la información y como el Excel es manejado por diferentes personas de la empresa no se tiene la información verídica ya que se corre el riesgo de que se tenga una información antigua, otro problema de trabajar de esta forma es que no todas las áreas que necesitan esta información la tienen de inmediato y retrasa el proceso del área de trabajo como por ejemplo "si financiera necesitaría el total de licencias de la empresa para realizar una estimación de gastos debería esperar para obtener el Excel final para realizar la estimación" y por ende se necesita la automatización y centralización de este proceso de control de activos de TI.

Justificación

Por esa falta de cohesión y disponibilidad de la información, surge la necesidad de crear una aplicación web que automatice el proceso de gestión de activos de TI de la empresa Garancheck Cía. Ltda., se escogió una aplicación web por el hecho de que aquella tecnología es portátil y puede ser utilizado en cualquier plataforma con el fin de agilizar dicho proceso y dar trazabilidad a los movimientos de los equipos. A su vez, ofrecer un resultado óptimo al momento de generar reportes de estados, mantener el inventario actualizado y consolidado al acceder y editar información desde un solo lugar centralizado, y dar más facilidad en el proceso actual de gestión de activos.

La ventaja al desarrollar un sistema para la gestión de activos de TI es que se puede realizar el almacenado de data, procesamiento y registro de la información del software de una manera ágil y efectiva, de acuerdo a las necesidades solicitadas de la empresa ya que las herramientas gratuitas encontradas en la web no se adaptan a las necesidades de la empresa, o sobrepasan un gasto mensual asignado por la empresa para dicho proceso, además con el desarrollo del sistema se tendrá un mayor control sobre los futuros cambios que podría requerir el sistema y su mantenimiento (Gonzales, 2018).

Objetivos de la Investigación

Objetivo General

Analizar, diseñar y construir una aplicación web para la "Gestión de Activos de TI" requerida por la empresa Garancheck Cía. Ltda.

Objetivos Específicos

Desarrollar un sistema web, multiplataforma, para el seguimiento de los activos de TI de forma eficiente y segura.

Centralizar en una base de datos la información de los equipos tecnológicos de la empresa Garancheck Cía. Ltda., para tener un control y conexión hacia la data de forma segura y eficiente.

Dar a conocer información detallada y actualizada de asignación o préstamo de cada uno de los activos de la empresa y la baja de estos.

Controlar los estados asignados a cada equipo y fecha de cambio de los estados de cada una de ellas.

Metodología

La metodología que se ajusta a nuestro proyecto es XP por los siguientes criterios: tiempo del proyecto, la cantidad de integrantes y cambios constantes mientras se avanza en el proyecto, entre

otros, donde la metodología será una herramienta ágil, ya que el desarrollo del software necesita una constante interacción con el cliente porque es propenso a varios cambios en la marcha de la implementación donde se siguió las siguientes fases.

Fase de levantamiento de requerimientos funcionales y no funcionales: Se utilizarán historias de usuarios, para describir las demandas del cliente y de cómo se supone que el sistema resolverá estas demandas o como apoyará a un proceso del negocio. Cada historia de usuarios estará escrita en una carta de historias y se conservan a través de todo el proceso de planificación y desarrollo. Con esta información nos permitirá a los desarrolladores producir una estimación razonable del esfuerzo para implementar los requisitos.

Fase de Diseño: Transcripción de los requerimientos a documentos requeridos por la ingeniería de software: Diagramas UML, diagramas de procesos, y diagramas de clases, nos ayudarán a organizar, planificar y visualizar la aplicación a desarrollarse. Los diagramas deben ser entendidos por cualquier de los desarrolladores y el cliente para así ayudar a explicar las relaciones de la aplicación de manera directa.

Fase de Codificación: Para la codificación se utilizará la metodología UML-based Web Engineering (UWE) que nos ayudará junto a las actividades de modelado a realizar un correcto análisis y diseño del sistema.

Fase de Pruebas: La fase de pruebas nos va a ayudar al control de calidad del software desarrollado en este proyecto de titulación, en base a la elaboración de un plan de pruebas que nos posibilita demostrar que los requerimientos solicitados sean implementados apropiadamente.

Capítulo 1

Marco teórico

1. Conocimiento empresarial

1.1. Metodologías ágiles. Tras una reunión celebrada en Utah-EEUU en el año 2001, se creó La Agile Alliance, un grupo formado por pensadores independientes sobre el desarrollo de software, cuyo fin es promover los conceptos de la metodología de desarrollo de software a través de principios ágiles (ISSI, 2003), esto permitiría a los desarrolladores de software poder responder con facilidad a los cambios que se imponen a lo largo del desarrollo de un proyecto, además de permitir otras salidas a los procesos tradicionales para desarrollar software que se basan más en documentación excesiva en las actividades desarrolladas durante el proyecto.

La metodología de desarrollo de software que ha prevalecido es en cascada tradicional, debido a que se puede utilizar en diferentes tipos de proyectos, pero esta metodología no se ajusta a la maleabilidad, tiempo y gran cantidad de cambios que tienen como características principales los proyectos de hoy en día. Las metodologías ágiles buscan el funcionamiento correcto del software, iteraciones con el equipo de desarrollo, colaboración con el cliente y responder a los cambios bruscos que presenta el proyecto, todo esto con el objetivo de lograr que el proyecto funcione de manera correcta, evitar una extensa documentación y que el proyecto no funcione como el cliente lo desea.

1.2. Extreme Programming (XP). La metodología XP permite al desarrollador una iteración con su equipo de trabajo para lograr retroalimentaciones continuas de lo que están desarrollando, permite una comunicación con el cliente para determinar si existen cambios en el proyecto o conocer si se están cumpliendo los objetivos planteados al inicio del proyecto, tener un tiempo de

respuesta a los cambios que se realizan en el proyecto conforme avanza y lograr cumplir con cada uno de los hitos del proyecto en los tiempos establecidos al inicio del proyecto.

XP es una metodología ágil que alienta el compromiso del equipo para lograr un buen clima de trabajo entre los desarrolladores y el cliente. XP se centra en retroalimentación continua entre clientes y desarrolladores, es decir, tiene un canal libre de comunicación para brindar soluciones creativas y pequeños tiempos de respuestas a los cambios. XP está especialmente orientado a proyectos que son cambiantes, donde sus requisitos no están completamente definidos y su periodo de finalización es corto (ISSI, 2003).

Tabla 1. Comparativa entre metodologías tradicionales y ágiles

Metodologías Ágiles	**Metodologías Tradicionales (Cascada)**
Grupos pequeños (<10 integrantes).	Grupos mayores a 10 integrantes.
Orientado a aceptar los cambios durante el desarrollo.	No están abiertas a los cambios.
Secuencia mínimamente supervisada, con limitados principios.	Secuencia más supervisada, con un sinnúmero de normas y/o políticas.
Contrato tradicional inexistente o es muy flexible	Existencia de un contrato acordado.
Se toma en cuenta al cliente como miembro del equipo de desarrollo	Únicamente el cliente socializa con el equipo en la presentación de los avances de los módulos del software desarrollado.
Pocos entregables, pocos roles.	Alta demanda de la documentación, muchos roles.

Nota: Tabla de paridad entre la flexibilidad de las metodologías ágiles versus el sólido manejo de las metodologías tradicionales.

1.1.3. Historias de usuarios. Las historias de usuarios son técnicas utilizadas para describir los requisitos del software, el cliente detalla las cualidades que el sistema requiere tener ya sean requerimientos funcionales o no funcionales, es necesario que las historias de usuario deben ser claras para que pueda entenderlo el desarrollador (ISSI, 2003).

1.1.4. Roles XP. Los roles propuestos por Kent Beck, quien fue el padre de XP son los siguientes: programador, cliente, responsable de pruebas (facilita las pruebas funcionales al

cliente), encargado de seguimiento (brinda realimentación al equipo), entrenador (encargado del proceso global), consultor y gestor (quien es el vínculo entre cliente y programador) (ISSI, 2003).

1.3. Ingeniería Web.

1.3.1. UWE. Ingeniería Web Basada en UML es una metodología que nos ayuda en la construcción de una aplicación web que se enfoca en el diseño sistemático, personalización, y generación semiautomática de escenarios, para los requerimientos los separa en fase de captura, definición y validación, que hace que esta metodología nos permita tener una aplicación adaptable a todos los requerimientos (Schwabe, Rossi, Olsina, & Pastor, 2008).

UWE nos plantea una extensión de UML que tiene 4 pasos:

- **Análisis de requisitos:** En este paso para representar los requerimientos se utiliza el diagrama de caso de uso.
- **Diseño conceptual:** En este paso se construye el modelo conceptual tomando en cuenta el diagrama de casos de uso, donde tenemos como resultado el diagrama de clases de dominio.
- **Diseño de navegación:** En este paso se muestra la navegación que tiene la aplicación con ayuda de los modelos de estructura de navegación donde se obtiene el diagrama de clases.
- **Diseño de presentación:** En este paso con ayuda de diagramas de interacción UML representamos las vistas de interfaz de usuario.

1.4. Sistema de Inventario.

1.4.1. Activos de TI. Los activos de TI son recursos tecnológicos que poseen las empresas para realizar gestiones, estos pueden ser tangibles o intangibles. Los activos de TI se enfocan en hardware, software o cualquier herramienta física que posea la empresa, asegurando que la empresa tenga la capacidad de gestionar cada uno de estos activos durante todo su ciclo de vida.

1.4.2. Gestión de activos de TI. Prácticamente la gestión no es más que llevar un registro de cada uno de los activos de una empresa ya sea hardware o software con el objetivo de optimizar la eficacia de los servicios de TI, mejorar el conocimiento del coste asociado a los activos que posee la empresa, mejorar el proceso de entrega y retiro de los activos y conocer la ubicación exacta de cada activo (La Rosa, 2018).

Según Maldonado, el que una organización tenga inventariado los activos de TI, permitirá, en distintas áreas de la institución, la adecuada toma de decisiones, mejorando así la ejecución de tareas operacionales (Maldonado, 2019). Cuando se implementa con éxito un sistema de gestión de activos casi toda la empresa obtiene beneficios y a largo plazo la empresa logrará grandes resultados en cada uno de los objetivos planteados.

- Tener un registro que evidencie los objetivos de desempeño.
- La usabilidad potenciada de los activos.
- Disminuir recursos como tiempo y dinero en formación superfluas
- Promover los recursos.
- Avalar el desarrollo organizacional.
- Software actualizado para reducir amenazas.
- Respetar las normas y requisitos regulatorios.
- Alinear los costos de los activos a necesidades específicas.

1.4.3. Sistema de gestión de activos de TI. Este sistema es una herramienta de software que posibilita la recopilación y rastreo de activos de TI, además de proporcionar opciones que se adecuan a los requerimientos de dicha empresa. El sistema de gestión de activos es el único encargado de proporcionar información actualizada acerca de la ubicación o estado de un activo,

haciendo más simple la administración y control de estos al personal encargado de TI (Chisaguano & Ludeña, 2019).

1.4.4. Información necesaria para la gestión de activos. La información registrada por el sistema de inventario debe ser clara para que el personal encargado de ingresar los datos lo pueda realizar de manera correcta. A continuación, se muestra un esquema de la aplicación requerida, en donde existen campos que deben ser llenados por los usuarios encargados de los registros de los activos.

- **Nro. de serie:** Código alfanumérico único de cada activo.
- **Tag:** Código alfanumérico único de cada activo.
- **Etiqueta:** Código del activo.
- **Tipo Equipo:** Que activo se va a gestionar.
- **Ubicación:** Piso 1, Piso 2, Bodega, etc.
- **MAC:** Código alfanumérico único del activo.
- **Estado Activo:** Conocer el estado del activo (Almacenado, donado, asignado, chatarrizado, préstamo, extraviado/robado, reparación).
- **Orden de Compra:** Número de la orden de compra.
- **Estado Funcional:** Operativo o no operativo.
- **Responsable:** Persona encargada del activo actualmente.
- **Adjuntos:** Carga de documentos de orden de compra o garantía del activo.
- **Área Responsable:** Que área se responsabiliza por el activo de la empresa.
- **Observaciones:** Alfanumérico 200 caracteres.

1.5. Base de Datos.

1.5.1. **Bases de datos relacionales.** Herramienta que se usa para almacenar y brindar el camino necesario a los datos vinculados entre sí. Las bases de datos relacionales consisten en el modelo relacional, que es una manera muy directa en la cual se presentan los datos mediante tablas. El modelo relacional brinda un modelo estándar de visualizar los datos, los desarrolladores vieron una gran ventaja en este modelo de base de datos y lo aprovecharon. El modelo relacional, garantiza que los datos sean precisos y accesibles, y lo hacen siguiendo ciertas reglas de integridad. Una de ellas es no permitir que existan filas duplicadas en la tabla, es decir que cada registro ingresado en la tabla se lo ingrese con una clave o código único con el objetivo de que no se ingrese información duplicada o errónea en la base de datos (Oracle, 2020).

1.6. Herramientas y lenguajes para el desarrollo del sistema web.

1.6.1. Microsoft Visual Studio. Herramienta de desarrollo que permite programar, depurar e implementar soluciones para cualquier plataforma, es así como se pueden crear aplicaciones o sistemas web para que se comuniquen entre estaciones de trabajo, dispositivos, consolas, etc. (Microsoft, 2020).

Algunas de las ventajas por las cuales se eligió ese lenguaje de programación son:

- Mayor aprovechamiento de autocompletado en archivos con codificación en C++.
- Obtiene un mejor desempeño en desarrollo local.
- Su explorador para soluciones es muy usado por los desarrolladores
- Compatibilidad GIT en IDE.
- Entendimiento Microsoft Azure.

1.6.2. Microsoft SQL Server. Es un sistema gestor de base de datos relacionales, que permite almacenar y recuperar información según las peticiones del usuario o de otras aplicaciones, dichas peticiones pueden ser: guardar, eliminar, sustituir, listar, etc. Además, SQL Server proporciona

seguridad, escalabilidad y soporte de transacciones lo que es vital en el sistema de gestión de activos de TI, debido a que se va a manejar información de casi toda la empresa (Microsoft, 2020).

1.6.3. Microsoft Visio. Permite crear diagramas de flujo, UML, diagramas organizacionales, planos de planta y diseños de ingeniería de manera fácil e intuitiva. (Microsoft, 2020) Visio permitirá crear diagramas UML para poder representar buenas prácticas para la construcción y documentación del sistema de gestión de activos. Los diagramas UML son normativas que describen la estructura y procedimiento de un sistema.

1.6.4. Microsoft .NET. Plataforma de desarrollo de software que permite fusionar sus herramientas de desarrollo y sistemas operativos, desde una perspectiva digital, el objetivo primordial de .NET es codificar aplicaciones, donde no importa el sistema operativo en el que se desplieguen. (Pellicer, 2017).

El desarrollo de aplicaciones usando .NET brinda grandes beneficios como:

- Disminuye el tiempo necesario para proyectos.
- Funcionalidades prediseñadas para facilitar a los desarrolladores en asuntos de exportación e importación
- Fácil mantenimiento.
- Aminorar costes.

1.6.5. C#. Es un lenguaje de programación simple y eficaz desarrollado por la empresa Microsoft, que permite la programación orientada a objetos. Ha sido diseñado con la función de compilar las aplicaciones que se ejecutan en el lenguaje de programación .NET Framework (Ortego, 2017).

1.6.6. HTML. Es un lenguaje que admite definir la información de una página web, lo realiza mediante etiquetas para definir texto, diseños, imágenes, videos, listas, estilos y otros componentes

que disponen la realización de un sitio web. HTML puede ser utilizado por personas que no sean especialmente programadores, la practicidad de HTML hace que un editor de texto sea lo único necesario para contener código HTML está con una extensión .HTML. Hoy en día no solo se pueden diseñar páginas web con HTML, si no también aplicaciones móviles o sistemas web (Alvarez, 2001).

1.6.7. jQuery. Es una biblioteca multiplataforma de software libre y código abierto de JavaScript, jQuery es una biblioteca pequeña que realiza un sinnúmero de funciones. Fue diseñada para que el desarrollador tenga la facilidad de navegar por el documento, mejora el manejo de eventos, mejora y facilita el diseño de animaciones y que el desarrollo de aplicaciones Ajax sean más simples para que funciones con una multitud de navegadores. También permite muchas más funcionalidades al ser de código abierto y tener funcionalidades propias para lograr buenos resultados en un corto periodo de tiempo (jQuery, 2020).

1.6.8. Bootstrap. Es un framework que combina css y JavaScript desarrollado por Twitter. Bootstrap es utilizado para caracterizar (cambiar estilos, colores y enlaces) los elementos de una página HTML, además, una de sus características que más ha sido aprovechada por los desarrolladores de aplicaciones web es que permite la construcción de sitios web resposive que se ejecutan en dispositivos móviles, desktop, tablets y smartphones (Guajardo, 2020).

1.6.9. Microsoft Azure. Es una herramienta de almacenado en la nube la cual nos permite tener acceso a infraestructura para poder almacenar y desarrollar prácticamente cualquier cosa en términos informáticos. En nuestro caso lo utilizaremos para almacenar nuestro sistema, compilarlo y poder administrarlo, con las credenciales necesarias y acceso a internet (Tecon, 2020).

1.7. Marco empresarial

1.7.1. Garancheck. La empresa de desarrollo Garancheck se encuentra ubicada en la ciudad de Quito, parroquia Pomasqui específicamente en la dirección: Pasaje Oe1C N2-61 y Manuela Sáenz. Garancheck es una empresa con un amplio margen de crecimiento y cuenta el talento humano con amplia experiencia que ofrecen servicios personalizados y asistencia técnica de calidad. Además, promueven la innovación, creatividad en cada uno de sus proyectos y la calidad que es la marca que los distingue, para lograr que cada uno de sus clientes mejoren sus procesos de negocios y crezcan (Garancheck Cía. Ltda, 2020).

Garancheck Cía. Ltda

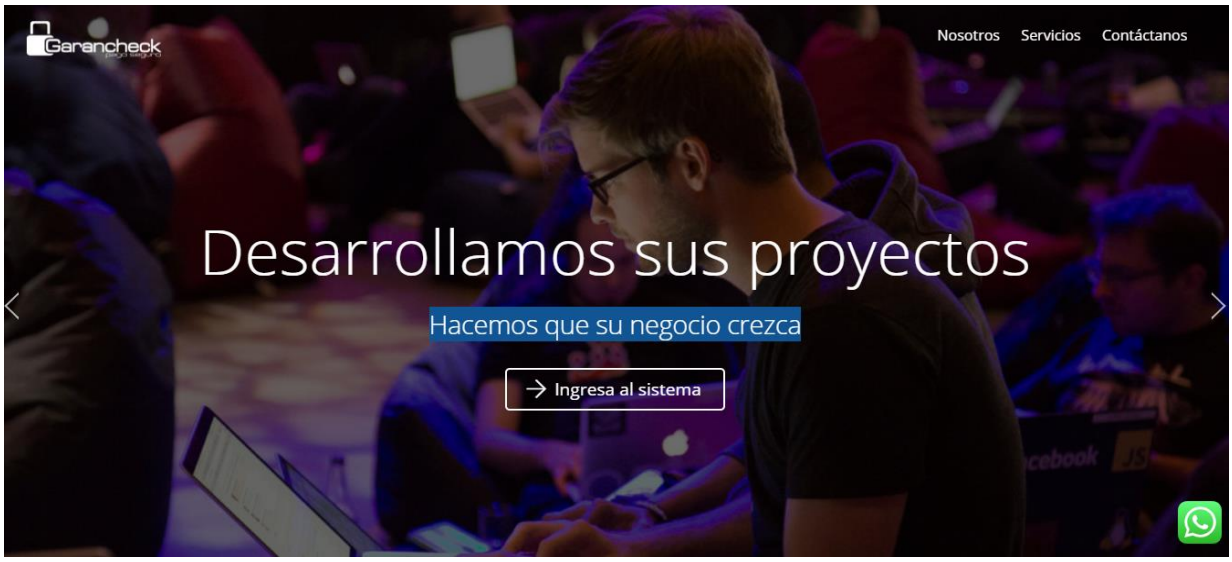


Figura 1. Empresa Garancheck Cía. Ltda.
Fuente: (Garancheck Cía. Ltda, 2020)

1.7.2. Servicios. Ofrece servicios en el área tecnológica a grandes, pequeñas y medianas empresas para que tengan un crecimiento exponencial en un pequeño periodo de tiempo. Los servicios que brinda son los siguientes:

- Desarrollo de software.
- Validación de cheques.
- Facturación electrónica.
- Sistema contable.

- Diseño web.
- Mantenimiento y reparación de Pcs.
- Outsourcing.

1.7.3. Misión y visón Garancheck.

1.7.3.1. Misión. “Proporcionar a nuestros clientes las mejores alternativas de productos y soluciones de información, con el objetivo de elevar su competitividad en el mercado, todo esto con valores como Responsabilidad, Lealtad, Respeto y Honradez, considerando el beneficio para nuestros clientes”. (Garancheck Cía. Ltda, 2020).

1.7.3.2. Visión. “En el 2022 mantener el liderazgo en el mercado ecuatoriano con tecnología e información eficiente, a través de una gestión transparente y creativa que genere rentabilidad, brindando el mayor beneficio a nuestros accionistas, colaboradores y clientes”. (Garancheck Cía. Ltda, 2020).

1.7.4. Organigrama estructural de Garancheck Cía. Ltda. Garancheck, cuenta con un capacitado cuerpo de trabajo, el cual está dividido en varias áreas, para conocer algunas de las funciones de cada área de trabajo y la persona encargada, cada área de la empresa se dividió en 3 niveles jerárquicos.

En el nivel 3 se encuentra el gerente general Felipe Cáceres y se encarga de la programación de las acciones de la empresa, organización, fijación de objetivos, etc. La contabilidad de la empresa es manejada de manera externa y se encarga de realizar balances, auditorías externas e internas y declaraciones. En el nivel 2 tenemos tres áreas: Administración / Operaciones, Comercial, Sistemas / Soporte Técnico.

1.7.4.1. Administración/Operaciones. Realizan inventarios de la empresa, compras, facturación, retenciones, inspección de cuentas por cobrar y cuentas por pagar (Tanya Cedeño).

1.7.4.2. Comercial. Esta área se encarga de las ventas, acercamiento con los clientes, seguimiento a clientes y asesoría del sistema (Freddy Báez).

1.7.4.3. Sistemas/Soporte técnico. Área encargada de soporte técnico y desarrollo de software (Enner Pérez).

En el nivel 1 se encuentra el personal de trabajo de cada área de nivel 2, en el área de administración/operaciones se encuentran tres personas las cuales son (Adamaris Prado, David Cotacachi y Jennifer Tasiguano) todos con el cargo de asistente administrativo/operador. En el área comercial se encuentran tres personas las cuales son (Daniel Báez, Miguel Rosas, Merly Molina) todos con el cargo de asesor, y finalmente en el área de sistemas/soporte técnico se encuentran cuatro personas las cuales son (Joel Ludeña, Amelia Cáceres, Alex Chacan y Francisco Aguirre) todos con el cargo de desarrollador.

Organigrama Garancheck Cía. Ltda.

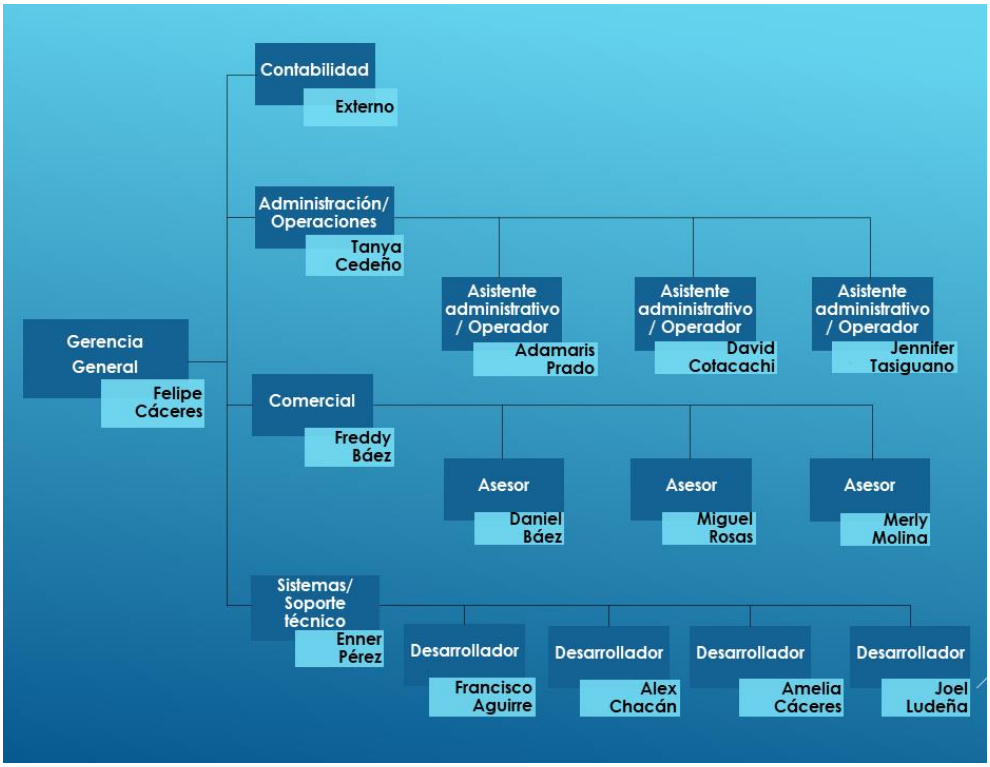


Figura 2. Organigrama estructural de Garancheck Cía. Ltda.
Elaborado por: Amelia Cáceres y Cristian Cazares

Capítulo 2

Análisis de escenarios

2. Fase de análisis y diseño

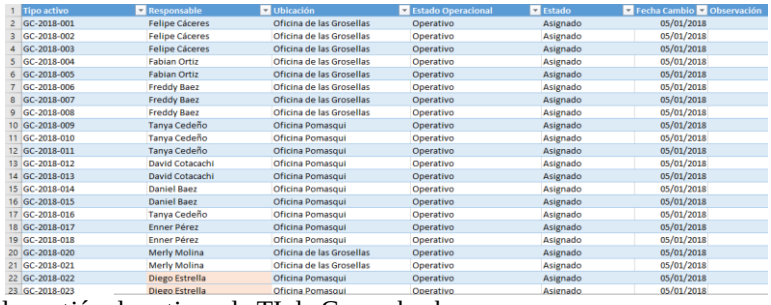
2.1. Proceso actual. La dirección de los activos permite contar con un registro inventariado de cada uno de los activos de la empresa, en algunos casos se conoce a detalle la ubicación y el responsable de dicho activo de la empresa. La gestión de activos permite a la empresa emplear cada uno de sus activos de manera más eficaz y esto evita que se realicen compras duplicadas o innecesarias del mismo activo. Además, se puede disminuir riesgos de costos, pérdidas o daños del activo.

La sistematización juega un rol fundamental en la empresa ya que ayuda a captar, catalogar, gestionar, analizar y reportar datos de los activos. Hoy en día, en gran cantidad de empresas, las herramientas para la gestión de activos son fundamentales ya que permiten el escalamiento de procesos operacionales y la toma de decisiones (Freshservice, 2020).

La empresa Garancheck Cía. Ltda., realiza la gestión de activos mediante un archivo de Excel donde se trata de mostrar a detalle la gestión de sus activos, en el archivo se muestra: el activo, estado del activo, tipo de activo, ubicación y usuarios. Enseguida, se mostrará el formato de gestión de activos de la empresa Garancheck Cía. Ltda.

2.2. Gestión de activos de TI.

Gestión de activos de TI



	Tipo activo	Responsable	Ubicación	Estado Operacional	Estado	Fecha Cambio	Observación
2	GC-2018-001	Felipe Cáceres	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
3	GC-2018-002	Felipe Cáceres	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
4	GC-2018-003	Felipe Cáceres	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
5	GC-2018-004	Fabian Ortiz	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
6	GC-2018-005	Fabian Ortiz	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
7	GC-2018-006	Freddy Baez	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
8	GC-2018-007	Freddy Baez	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
9	GC-2018-008	Freddy Baez	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
10	GC-2018-009	Tanya Cedeño	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
11	GC-2018-010	Tanya Cedeño	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
12	GC-2018-011	Tanya Cedeño	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
13	GC-2018-012	David Cotacachi	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
14	GC-2018-013	David Cotacachi	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
15	GC-2018-014	Daniel Baez	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
16	GC-2018-015	Daniel Baez	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
17	GC-2018-016	Tanya Cedeño	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
18	GC-2018-017	Enner Pérez	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
19	GC-2018-018	Enner Pérez	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
20	GC-2018-020	Merly Molina	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
21	GC-2018-021	Merly Molina	Oficina de las Grosellas	Operativo	Asignado	05/01/2018	
22	GC-2018-022	Diego Estrella	Oficina Pomasqui	Operativo	Asignado	05/01/2018	
23	GC-2018-023	Diego Estrella	Oficina Pomasqui	Operativo	Asignado	05/01/2018	

Figura 3. Reporte de la gestión de activos de TI de Garancheck.
Elaborado por: Amelia Cáceres y Cristian Cazares

En la Figura 3., se muestra la gestión actual de activos de TI en la empresa Garancheck, la información que se refleja en dicho archivo es:

- **Tipo de activo:** Código único del activo de TI de la empresa.
- **Responsable:** Persona responsable del activo de la empresa.
- **Ubicación:** Área u oficina de la persona encargada del activo.
- **Estado operacional:** Si el activo se encuentra operativo o no operativo.
- **Estado del activo:** Almacenado, asignado, chatarrizado, préstamo, extraviado y reparación.
- **Fecha:** Fecha en la cual el activo es almacenado, asignado, chatarrizado, prestado, extraviado o reparado.
- **Observaciones:** Se detalla alguna observación del activo después de su devolución.

2.3. Usuarios.

Gestión de Usuarios

	Empleado	Departamento
2	Felipe Cáceres	Gerencia
3	Tanya Cedeño	Administración
4	Freddy Baez	Ventas
5	Tadeo Pullas	Administración
6	Enner Pérez	Sistemas
7	Daniel Baez	Ventas
8	Francis Espin	Ventas
9	Diego Estrella	Ventas
10	Ana Merino	Ventas
11	Pablo Zapata	Ventas
12	Juan Pombosa	Ventas
13	Amelia Caceres	Sistemas
14	Francisco Aguirre	Sistemas
15	Alex Chacan	Sistemas
16	Joel Ludeña	Sistemas
17	Merly Molina	Ventas
18	Fabian Ortiz	Administración
19	Miguel Rosas	Ventas
20	David Cotacachi	Administración
21	Adamaris Prado	Administración
22	Jennifer Tasiguano	Administración

Figura 4. Gestión de usuarios.
Elaborado por: Amelia Cáceres y Cristian Cazares

En la Figura 4., se muestra la gestión de usuarios donde se muestran los campos: empleado, donde se detalla el nombre y apellido de los empleados que han realizado préstamos de activos, y el campo departamento, que es el departamento o área donde labora el empleado descrito en el anterior campo.

2.4. Tipo Activo.

Tipo de activos

	Tipo	Tag	Marca	Serial
2	Laptop	GC-2018-001	Acer	NXGHGAA0056471D13D7600
3	Cargador	GC-2018-002	Acer	KP065030176450CA8BPE04
4	IMPRESORA L360	GC-2018-003	Epson	O015T34W1
5	Laptop	GC-2018-004	HP	5CD7338X24
6	Cargador	GC-2018-005	HP	WFTLC0A3U89FIJ
7	Laptop	GC-2018-006	HP	5CD733BVCF
8	Cargador	GC-2018-007	HP	WFTKU0EDHADNQ8
9	Telefono	GC-2018-008	Panasonic	4cbkd203462
10	Laptop	GC-2018-009	HP	5CD811975Q
11	Cargador	GC-2018-010	HP	WFTKU0ENJAJD4S
12	TV Sony	GC-2018-011	Sony	8502435
13	Laptop	GC-2018-012	HP	5CD733BV94
14	Cargador	GC-2018-013	HP	WFTLC0A4L89J2Q
15	Laptop	GC-2018-014	Acer	LXTCJ06013633030FD2500
16	Cargador	GC-2018-015	Acer	AP0900300962607FFAVL01
17	Decodificador	GC-2019-016	DirectTV	T10XAT323TS17R
18	Laptop	GC-2018-017	HP	5CD80971NY
19	Cargador	GC-2018-018	HP	WFTLC0A4L89J3L
20	Laptop	GC-2018-020	HP	5CG6353QVL
21	Cargador	GC-2018-021	HP	WDUUT0BCX9UNBN
22	Laptop	GC-2018-022	HP	5CD8185K2F
23	Cargador	GC-2018-023	HP	WFTLC0A4LAQHR6

Figura 5. Tipo de activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

En la Figura 5., se describe los tipos de activos que posee la empresa, se muestran los campos: tipo, tag, marca y serial.

- **Tipo:** Modelo de activo de la empresa.
- **Tag:** Etiqueta del activo de TI.
- **Marca:** Identificación comercial del activo de TI.

- **Serial:** Código único de los activos de TI de la empresa.

2.5. Diagrama de flujo de activos. En el presente diagrama se explica el proceso que debe seguir el administrador del sistema para realizar la administración de activos del negocio.

- Nuevo Registro: se registrará los datos completos del nuevo activo de TI, posteriormente se procede a guardar.
- Estado de activos: un activo puede tener un solo estado (extraviado, asignado, préstamo, chatarrizado, reparación, almacenado), además el sistema permitirá conocer a que personal de la empresa se le asigna dicho activo.
- Cambio estados: solo el administrador puede regresar el activo a un estado anterior.

Diagrama de flujo de activos

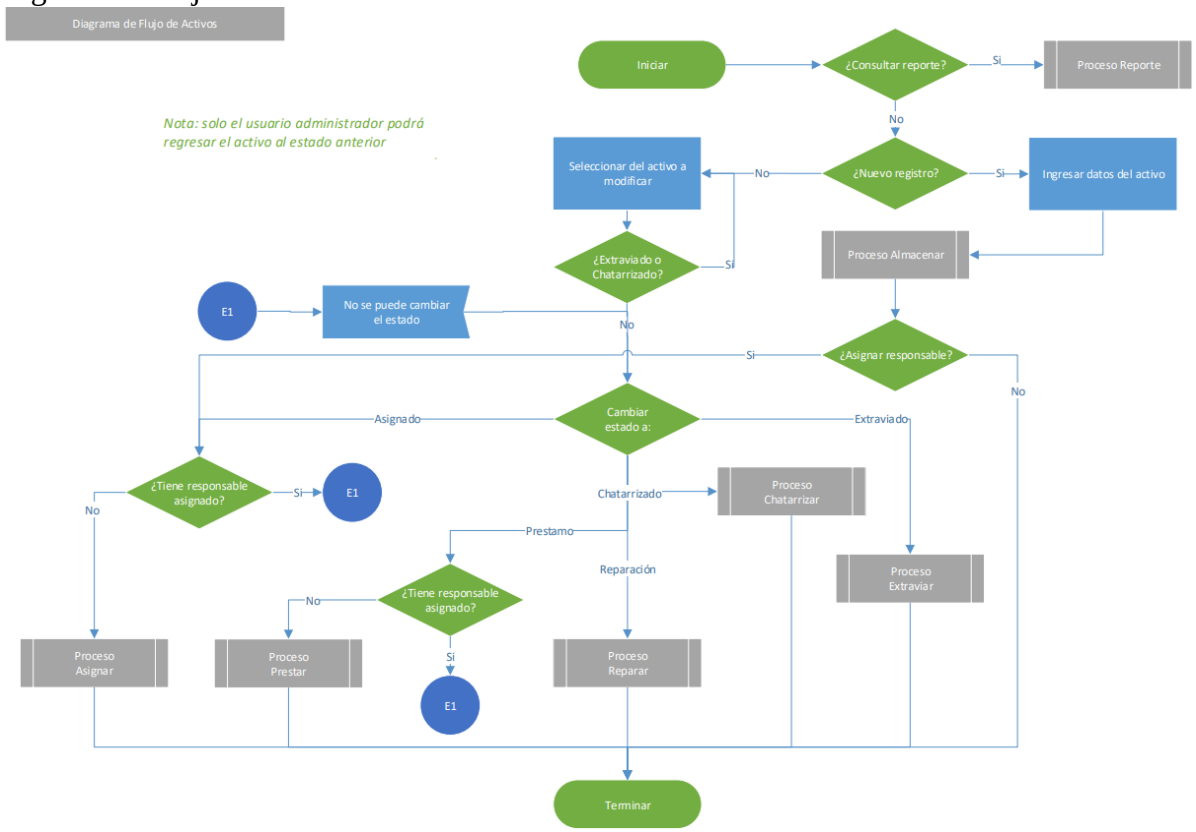


Figura 6. Gestión de activos de TI - Diagrama de flujo.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.6. Especificación de requisitos por parte de la empresa Garancheck Cía. Ltda. Garancheck Cía. Ltda., actualmente llevan el proceso de inventario/manejo de activos de manera manual. Este proceso empieza con la creación del activo y pasando por etapas de asignación de responsable, bodegaje, bajas, reasignaciones, préstamos y demás temas de mantenimiento que el activo requiere.

La necesidad de agilizar el proceso de gestión de activos nace por dar trazabilidad a los movimientos de los equipos, generar reportes de estados, mantener el inventario actualizado - consolidado y dar más facilidad en el proceso actual de gestión de activos.

2.7. Roles y responsabilidades de usuarios. Los usuarios del software serán los miembros de TI, con roles asignados según su responsabilidad.

Tabla 2. Roles y responsabilidades de usuarios

Rol	Responsabilidad
Responsable del manejo de inventarios de activos de TI.	✓ Creación, modificación y eliminación de activos. ✓ Asignación y des asignación de responsables. ✓ Mantenimiento de equipos. ✓ Préstamo de equipos. ✓ Ubicaciones de activos. ✓ Reportes.
Jefe de área.	✓ Reportes.
Administrador.	✓ Mantenimiento de la aplicación. • Usuarios. • Catálogos. • Reportes. ✓ Creación, modificación y eliminación de activos. ✓ Asignación y des asignación de responsables. ✓ Mantenimiento de equipos. ✓ Préstamo de equipos. ✓ Ubicaciones de activos. ✓ Reportes.
Usuario final.	✓ Recibe notificaciones.

Nota: Tabla de roles y responsabilidades de usuarios.

2.8. Diagrama de caso de uso – Gestión de activos. El sistema de gestión de activos de TI tiene que cumplir con el caso de uso a continuación.

Gestión de activos

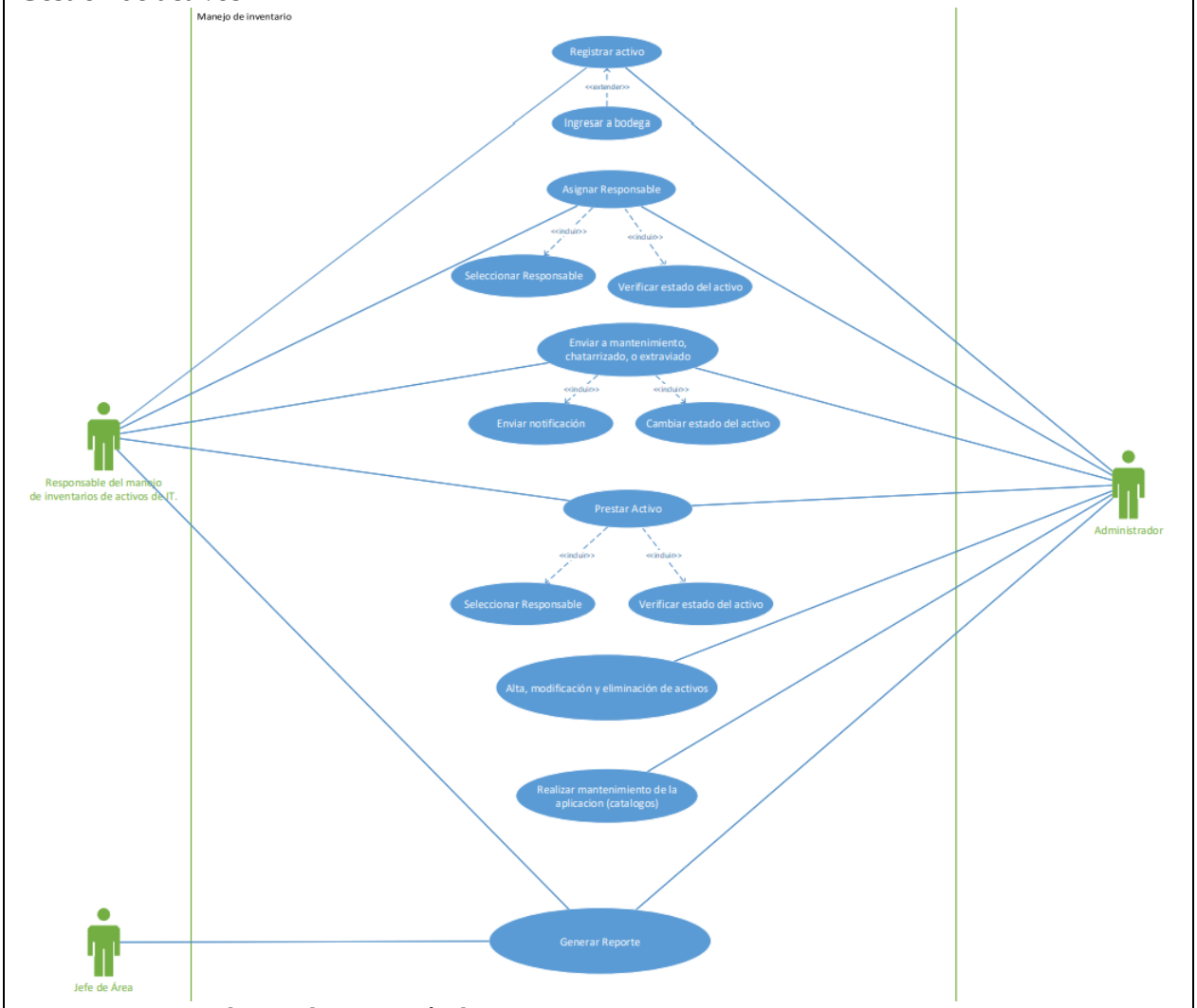


Figura 7. Diagrama de caso de uso gestión de activos.
Elaborado por: Amelia Cáceres y Cristian Cazares

A continuación se describirá que acciones realizarán los usuarios del sistema.

2.8.1. Responsable del manejo de inventario de activos de TI.

2.8.1.1. Registro de un activo.

- El sistema solicita al responssable los datos completos del activo a registrar.
- El responsable procede a guardar los datos ingresados y automaticamente el sistema registra en bodega el nuevo activo.

2.8.1.2. Asignar responsable.

- El sistema solicita que se asigne un responsable al activo.
- El usuario asigna un responsable para el activo.
- El responsable verifica el estado del activo para poderlo asignarlo.
- Si el activo se encuentra disponible el responsable procede a realizar la asignación.

2.8.1.3. Mantenimiento, extraviado o chatarrizado.

- El usuario solicita al sistema el envío a mantenimiento o chatarrizado del activo.
- El sistema verifica si el activo pertenece a la empresa.
- El sistema envía una notificación al usuario y posteriormente cambia el estado del mismo activo.

2.8.1.4. Préstamo de activos.

- El sistema solicita los datos del usuario el para préstamo del activo.
- El sistema registra los datos del usuario.
- El sistema verifica el estado del activo.
- El responsable procede a realizar el préstamo del activo.

2.8.1.5. Generar reporte.

- El responsable solicita al sistema un reporte de los activos de TI, préstamos y estado de activos.

2.8.2. Administrador.

2.8.2.1. Creación, modificación y eliminación de activos.

- El administrador solicita al sistema una lista de los activos a modificar.
- El administrador procede a modificar, eliminar los activos seleccionados.

- El administrador procede a crear el activo en caso de no existir en la aplicación.

2.8.2.2. Mantemiento de la aplicación.

- El administrador tiene acceso al mantemiento preventivo del sistema.

2.8.3. Jefe de área.

2.8.3.1. Generar reporte.

- Tiene acceso unicamente a los reportes que genera el sistema.

2.9. Diagramas de caso de uso – Asignación y devolución de activos.

2.9.1. Asignar activo.

- El usuario solicita que se le asigne un activo.
- El sistema registra los datos del usuario al cual se le asignará el activo.
- Después de asignarle el activo al usuario se envía una notificación de asignación de activos.
- El usuario tendrá acceso a la notificación.

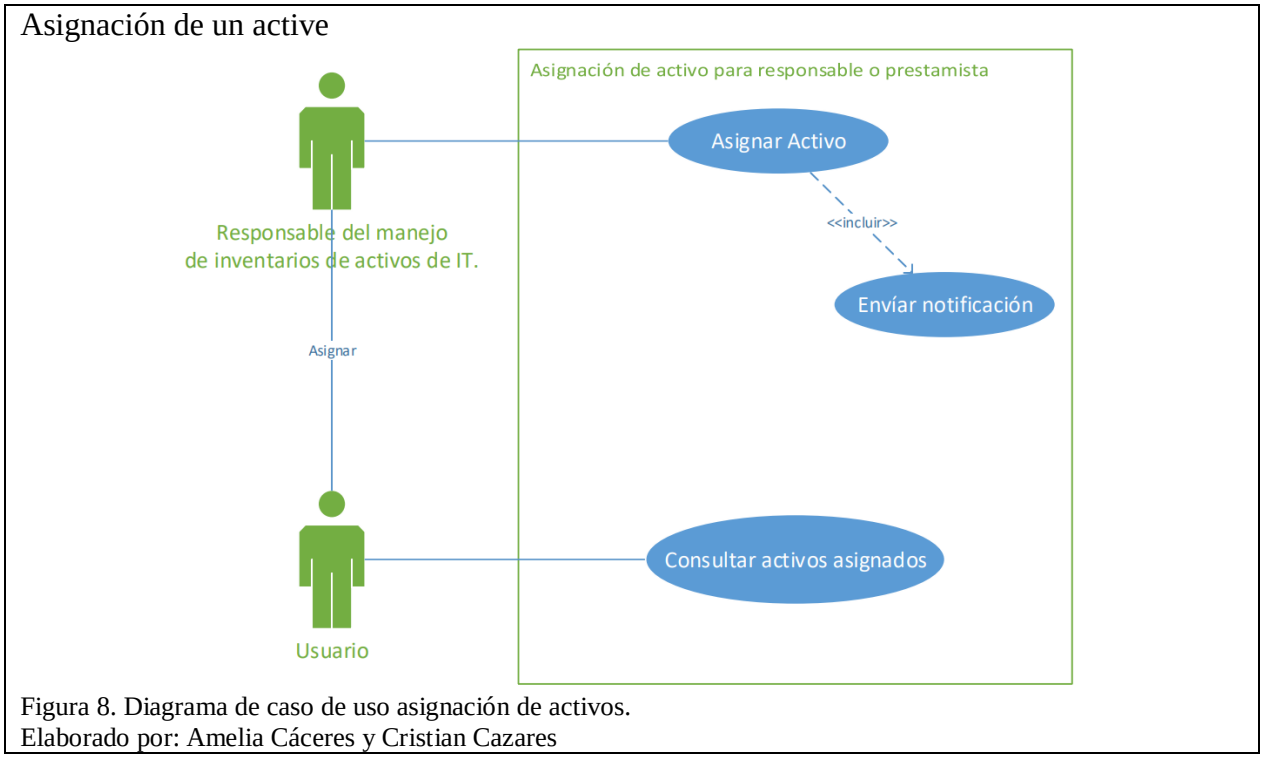


Figura 8. Diagrama de caso de uso asignación de activos.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.9.2. Alamcenado del activo

- El usuario solicita la devolución del activo.
- El responsable ingresa la devolución al sistema.
- El sistema registra observaciones en la devolución del activo.
- El sistema emite una notificación de devolución del activo al usuario el cual tiene acceso a la misma.

Almacenado del activo

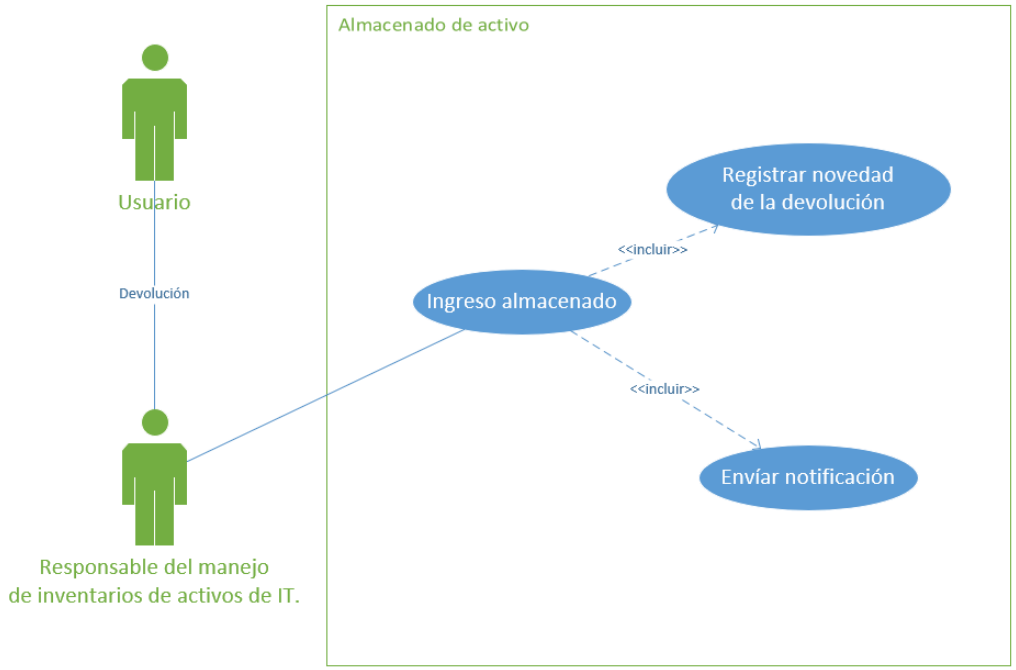


Figura 9. Diagrama de caso de uso almacenado del activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.10. Requerimientos Funcionales y no Funcionales

2.10.1. Requerimientos Funcionales

2.10.1.1. Login.

Tabla 3. Requerimiento funcional Login

Identificación	RF01
Nombre del requerimiento	Login.

Precondición	Creación del usuario en el módulo de gestión de usuarios.
Proceso	Un usuario debe ingresar el usuario y contraseña, proporcionados al crearlo en el módulo de administración, en la página de Inicio de Sesión.
Postcondición	Ninguna. Ingresar al sistema con el rol con el que fue creado el usuario.
Salida	Página principal del sistema de gestión de activos, donde se le da la bienvenida.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional Login.

2.10.1.2. Creación de activos de TI.

Tabla 4. Requerimiento funcional creación de activos

Identificación	RF02
Nombre del requerimiento	Crear nuevo activo.
Precondición	Contar con el rol y permiso necesario para realizar la acción de crear activos. Que no exista un registro con el mismo número de serie y tag, en la base de datos.
Proceso	El usuario debe ingresar todos los campos obligatorios y no obligatorios para la creación del activo en la base de datos.
Postcondición	El activo debe siempre iniciar con estado Almacenado.
Salida	El activo editado se mostrará en la lista de administración de activos y un mensaje de procesado satisfactoriamente.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional crear.

La aplicación estará en la capacidad de enviar notificaciones a los usuarios que han sido asignados como responsables de un equipo.

2.10.1.3. Cancelación de procesos.

Tabla 5. Requerimiento funcional cancelar

Identificación	RF03
Nombre del requerimiento	Cancelar.
Precondición	Ninguna.
Proceso	Dar clic sobre el botón Cancelar.

Postcondición	Ninguna.
Salida	El sistema lo regresará al a vista anterior de la acción que canceló.
Prioridad	Media.

Nota: Tabla de requerimiento funcional cancelar.

2.10.1.4. Eliminación de activos de TI.

Tabla 6. Requerimiento funcional eliminar

Identificación	RF04
Nombre del requerimiento	Eliminar activo.
Precondición	Contar con el rol y permiso necesario para realizar la acción de eliminar activos, ingresar al módulo de Activos.
Proceso	El usuario debe seleccionar el registro deseado con la opción de eliminar, identificada con el botón color rojo y una x.
Postcondición	El activo se eliminará permanentemente.
Salida	Mensaje de procesado satisfactoriamente.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional eliminar.

2.10.1.5. Reportes.

Tabla 7. Requerimiento funcional reporte

Identificación	RF05
Nombre del requerimiento	Exportar.
Precondición	Contar con el rol y permiso necesario para ingresar a las pantallas de reportes de activos e ingresar al módulo de Reportes.
Proceso	Ingresar los datos necesitados para aplicar el filtro de activos. Seleccionar el botón aplicar para visualizar los resultados en la tabla de reportes. Seleccionar la opción de descargar en formato Excel o PDF.
Postcondición	------
Salida	Documento con los datos filtrados de la tabla de reportes.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional reporte.

2.10.1.6. Ingreso de un nuevo activo de TI.

Tabla 8. Requerimiento funcional ingreso de un nuevo activo

Identificación	RF06
Nombre del requerimiento	Ingreso de un nuevo activo de TI (sería igual que crear activo).
Proceso	El usuario debe ingresar todos los campos obligatorios y no obligatorios para la creación del activo en la base de datos.
Postcondición	El activo debe siempre iniciar con estado Almacenado.
Salida	El activo editado se mostrará en la lista de administración de activos y un mensaje de procesado satisfactoriamente.
Prioridad	Alta.
Proceso	El usuario debe ingresar todos los campos obligatorios y no obligatorios para la creación del activo en la base de datos.

Nota: Tabla de requerimiento funcional ingreso de un nuevo activo.

2.10.1.7. Asignación de activos de TI.

Tabla 9. Requerimiento funcional asignación de un activo

Identificación	RF07
Nombre del requerimiento	Asignación de activos de TI.
Precondición	Contar con el rol y permiso necesario para realizar la acción de cambiar de estado al activo. Que el estado del activo sea Almacenado.
Proceso	Ingresar al módulo de activos. Seleccionar el activo con estado Almacenado para asignarlo a un usuario. Ingresar el responsable y la Ubicación del activo.
Postcondición	El estado del equipo cambia a Operativo. Envío de correo al responsable del manejo de inventarios y al usuario a quien se le asignó el activo.
Salida	El sistema lo regresará al listado de administración de activos, y un mensaje de procesado satisfactoriamente.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional ingreso de un nuevo activo.

2.10.1.8. Almacenamiento de activos de TI.

Tabla 10. Requerimiento funcional almacenamiento de un activo

Identificación	RF08

Nombre del requerimiento	Almacenamiento de activos de TI.
Precondición	Contar con el rol y permiso necesario para realizar la acción de cambiar de estado al activo. Que el estado no se chatarrizado, ni extraviado.
Proceso	Ingresar al módulo de activos. Seleccionar el activo para cambiarlo a estado Almacenado. Ingresar el Estado del equipo y la Ubicación del activo.
Postcondición	-----
Salida	El sistema lo regresará al listado de administración de activos, y un mensaje de procesado satisfactoriamente.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional almacenamiento de un activo.

2.10.1.9. Prestación de activos de TI.

Tabla 11. Requerimiento funcional prestación de activos

Identificación	RF09
Nombre del requerimiento	Prestación de activos de TI.
Precondición	Contar con el rol y permiso necesario para realizar la acción de cambiar de estado al activo. Que el estado del activo sea almacenado.
Proceso	Ingresar al módulo de activos. Seleccionar el activo para cambiarlo a estado Prestado. Ingresar el responsable y la Ubicación del activo.
Postcondición	----
Salida	El sistema lo regresará al listado de administración de activos, y un mensaje de procesado satisfactoriamente.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional prestación de activos.

2.10.1.10. Gestión de usuarios (creación).

Tabla 12. Requerimiento funcional crear usuarios

Identificación	RF10
Nombre del requerimiento	Creación de usuarios.

Precondición	Contar con el rol y permiso necesario para realizar la acción de crear usuarios. Existencia de al menos un ***Rol.***
Proceso	Ingresar al módulo de ***Mantenimiento/Usuarios***. Seleccionar la opción de ***Crear nuevo***. El usuario debe ingresar todos los campos obligatorios y no obligatorios para la creación del usuario en la base de datos.
Postcondición	-----
Salida	El sistema lo regresará al listado de administración de usuarios, y un mensaje de ***procesado satisfactoriamente***.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional crear usuarios.

2.10.1.11. Gestión de perfiles/roles.

Tabla 13. Requerimiento funcional gestión de perfiles/roles

Identificación	RF011
Nombre del requerimiento	Creación de perfiles/roles.
Precondición	Contar con el rol y permiso necesario para realizar la acción de crear ***roles***.
Proceso	Ingresar al módulo de ***Mantenimiento/Roles***. Seleccionar la opción de ***Crear nuevo***. El usuario debe ingresar el ***nombre*** del rol y las ***pantallas***(acciones) a las que tendrá permiso el rol.
Postcondición	----
Salida	El sistema lo regresará al listado de administración de roles, y un mensaje de ***procesado satisfactoriamente***.
Prioridad	Alta.

Nota: Tabla de requerimiento funcional gestión de perfiles/roles.

2.10.2. Requerimientos no Funcionales.

2.10.2.1. Interfaz de usuario del sistema.

Tabla 14. Requerimiento no funcional interfaz de usuario del sistema

Identificación	RNF01
Nombre del requerimiento	Interfaz de usuario del sistema.
Descripción	Diseño de interfaz con características de fácil uso y comprensión.

	Debe ser compatible con los navegadores Chrome, Internet Explorer 11+, Mozilla. La aplicación tendrá un menú de navegación, con los diferentes módulos para el acceso de los usuarios, ubicado en la parte superior de la pantalla. La aplicación deberá indicar que usuario este logueado actualmente.
Prioridad	Media.

Nota: Tabla de requerimiento no funcional interfaz de usuario del sistema.

2.10.2.2. Arquitectura del sistema.

Tabla 15. Requerimiento no funcional arquitectura del sistema

Identificación	RNF02
Nombre del requerimiento	Arquitectura del sistema.
Descripción	Compatible con SQL Server 2012. Herramienta de Desarrollo (IDE): Visual Studio .NET 2015 o superior, con ASPX o C#. Debe cumplir con el estilo de arquitectura MVC.
Prioridad	Media.

Nota: Tabla de requerimiento no funcional arquitectura del sistema.

2.10.2.3. Seguridad del sistema.

Tabla 16. Requerimiento no funcional seguridad del sistema

Identificación	RNF03
Nombre del requerimiento	Seguridad del sistema.
Descripción	El usuario debe iniciar sesión con las credenciales proporcionadas por la aplicación. La aplicación deberá almacenar el LOG de eventos para creación, bajas y modificaciones de registros. Los usuarios con rol de Administrador, tendrán acceso a todos los módulos y acciones de la aplicación.
Prioridad	Alta.

Nota: Tabla de requerimiento no funcional seguridad del sistema.

2.10.2.4. Rendimiento y disponibilidad del sistema.

Tabla 17. Requerimiento no funcional rendimiento y seguridad del sistema

Identificación	RNF04
Nombre del requerimiento	Rendimiento y disponibilidad del sistema.
Descripción	Los usuarios podrán tener acceso a la aplicación desde todo tipo de dispositivos que cuenten con acceso a internet en cualquier lugar. La aplicación estará alojada en los servidores de Microsoft Azure, garantizando así, una disponibilidad del 99% de los datos.
Prioridad	Alta.

Nota: Tabla de requerimiento no funcional rendimiento y seguridad del sistema.

2.11. Estados de un activo.

2.11.1. Estado Asignado.

- El responsable genera un registro con el nuevo activo que ha llegado a la compañía.
- El responsable ingresa todos los datos necesarios.
- Cuando el estado del activo es "Asignado", el estado funcional pasa automáticamente a Operativo.
- Selecciona el botón "Guardar", se genera un mail automáticamente hacia el responsable del manejo de inventarios, usuario final siempre y cuando el estado del activo sea Asignado.

El mail contendrá información básica como:

- Estimado usuario se te ha asignado el siguiente activo:
- Datos del activo: marca, modelo, número de serie, fecha de asignación, ubicación.
- Estado del activo.
- Responsable de asignación.

Asignación de un activo de TI

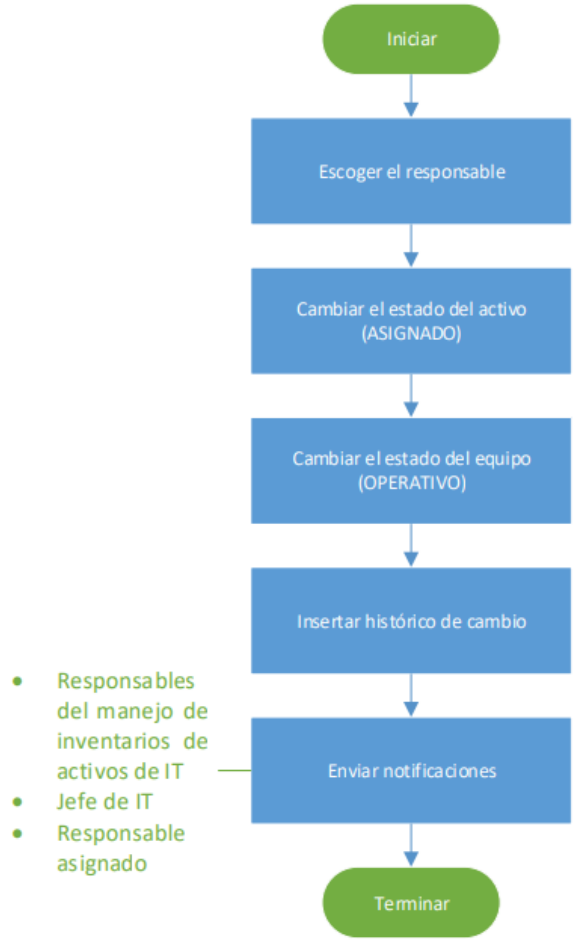


Figura 10. Diagrama de proceso estado asignado.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.11.2. Estado Almacenado

- El responsable genera un registro con el nuevo activo que ha llegado a la compañía.
- El responsable ingresa todos los datos necesarios.
- En este estado, el estado funcional del equipo me deberá permitir escoger de un listado la opción “operativo” o “no operativo”.
- Selecciona los botones “Guardar”, se genera un mail automáticamente hacia el responsable del manejo de inventarios.

El mail contendrá información básica como:

- Estimado usuario se te ha asignado el siguiente activo:
- Datos del activo: marca, modelo, número de serie, fecha de asignación, ubicación.
- Estado del activo.
- Responsable de asignación.

Almacenamiento de un activo de TI

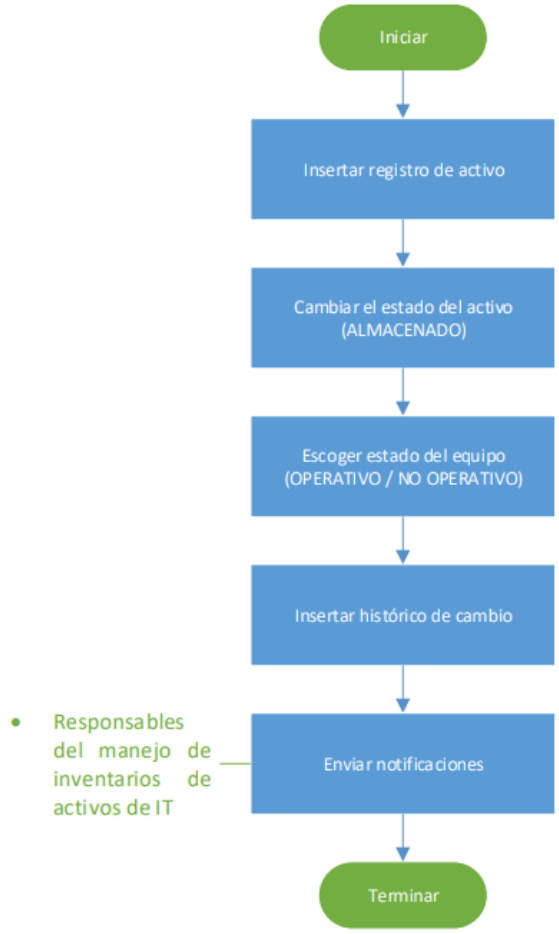


Figura 11. Diagrama de proceso estado almacenamiento.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.11.3. Estado Chatarrizado

- El responsable buscará el activo que va a cambiar su estado a Chatarrizado.
- Seleccionará el activo encontrado, escoger la opción cambiar a estado “Chatarrizado”.

- El estado funcional del activo pasará automáticamente a No Operativo.
- El sistema deberá borrar la información los campos Responsable, Ubicación y Observaciones.
- Selecciona el botón “Guardar”, y se genera un mail automáticamente hacia el responsable del manejo de inventarios y jefe de área TI.

El mail contendrá información básica como:

- Estimado usuario se ha Chatarrizado el siguiente activo:
- Datos del activo: marca, modelo, número de serie.
- Estado del activo.

Chatarrizado de un activo de TI

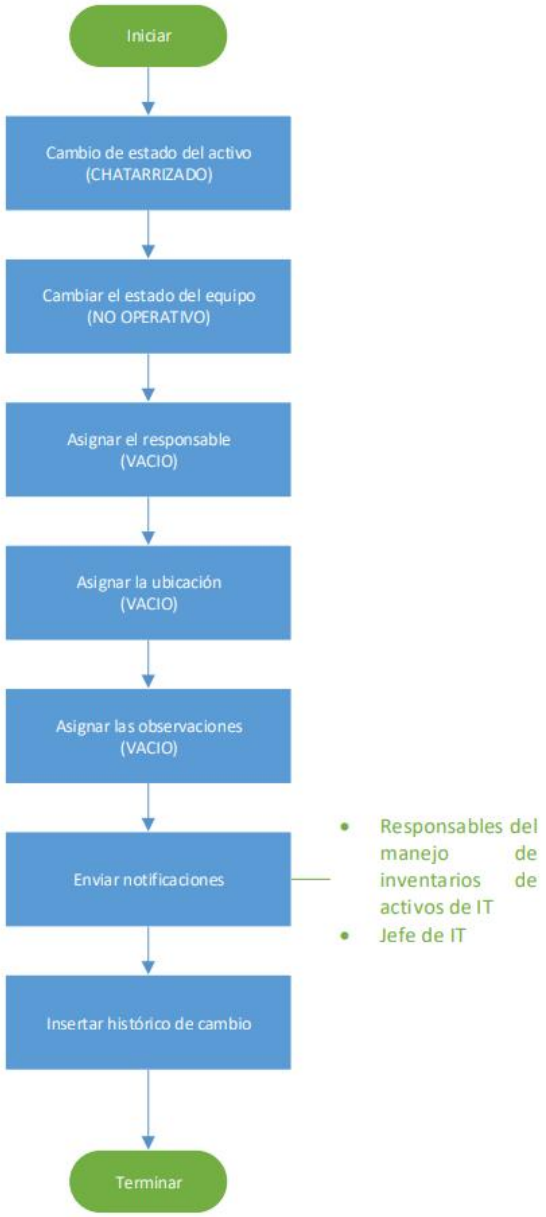


Figura 12. Diagrama de proceso estado Chatarrizado.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.11.4. Estado Préstamo

- El responsable buscará el activo a modificar.
- Selecciona el activo encontrado y escogerá la opción cambiar a estado "préstamo" y se deberá habilitar solo los campos Responsable y Ubicación para modificarlos.

- Selecciona el botón “Guardar” y se genera un mail automáticamente hacia el responsable del manejo de inventarios y el usuario final.

El mail contendrá información básica como:

- Estimado usuario, se te ha asignado en Préstamo el siguiente activo:
- Datos del activo: marca, modelo, número de serie, fecha de asignación, ubicación.
- Estado del activo.
- Responsable de asignación.

Préstamo de un activo de TI

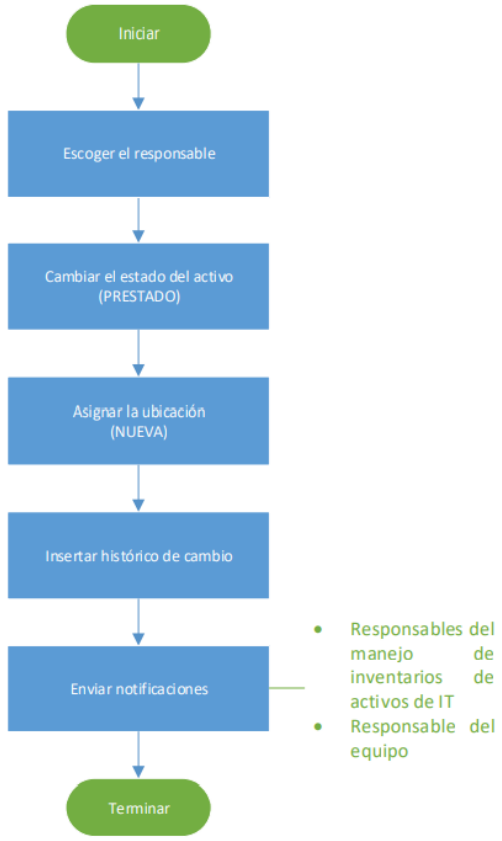


Figura 13. Diagrama de proceso estado préstamos.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.11.5. Estado Extraviado

- El responsable buscará el registro a modificar.

- Selecciona el activo encontrado y escogerá la opción cambiar a estado “Extraviado” se cambia el estado del equipo a No Operativo, automáticamente se blanquean los campos Responsable y Ubicación.
- Selecciona los botones “Guardar”, se genera un mail automáticamente hacia el responsable del manejo de inventarios.

El mail contendrá información básica como:

- Estimado usuario, se ha reportado como extraviado el siguiente activo:
- Datos del activo: marca, modelo, número de serie, fecha de asignación, ubicación.
- Estado del activo.
- Responsable de asignación.

Estado extraviado de un activo de TI

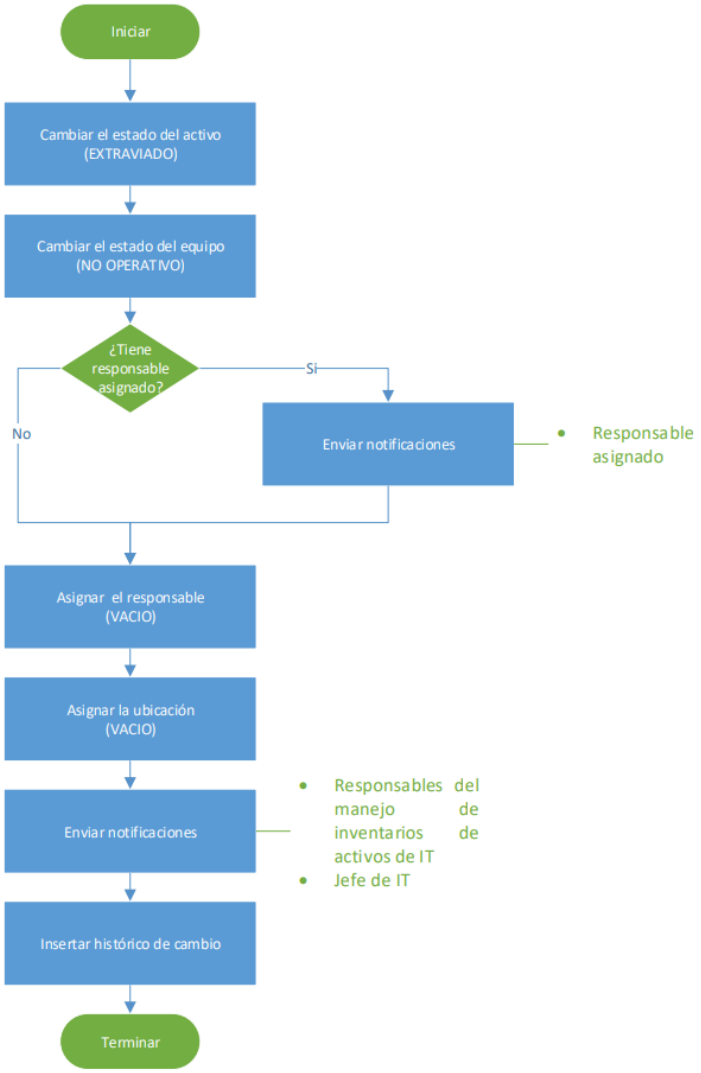


Figura 14. Diagrama de proceso estado extraviado.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.11.6. Estado Reparación

- El responsable busca el registro a modificar.
- Selecciona el activo y escogerá la opción cambiar a estado “Reparación”, se cambiará el estado del equipo a No Operativo, se habilita el campo Ubicación.
- Selecciona el botón “Guardar”, se genera un mail automáticamente hacia el responsable del manejo de inventarios y el usuario que tenía asignado el Activo.

El mail contendrá información básica como:

- Estimado usuario, el activo que se te ha asignado; está en reparación:
- Datos del activo: marca, modelo, número de serie, fecha de asignación, ubicación.
- Estado del activo.
- Responsable de asignación.

Reparación de un activo de TI

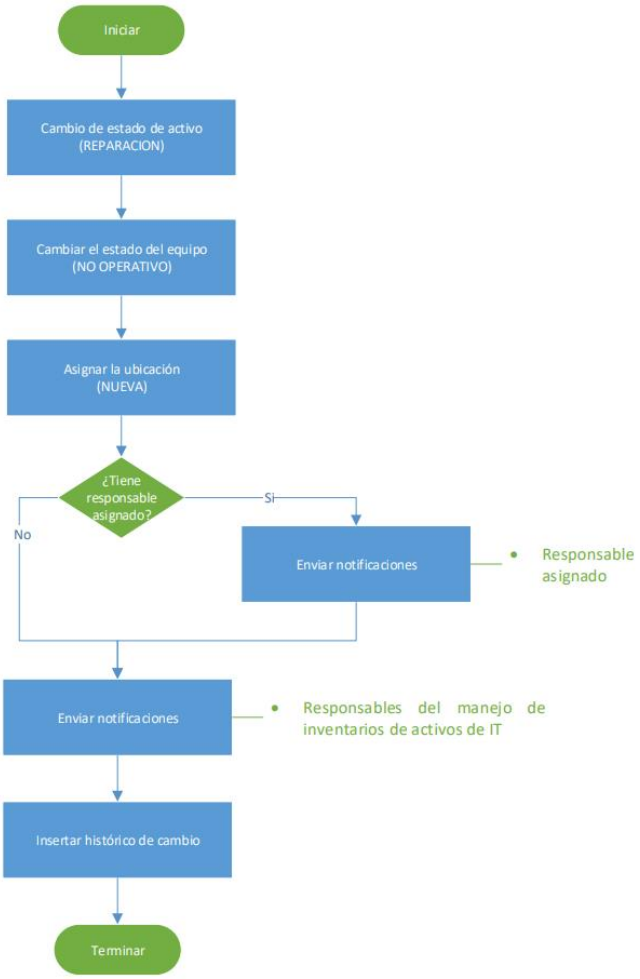


Figura 15. Diagrama de proceso estado reparación.
Elaborado por: Amelia Cáceres y Cristian Cazares

2.11.7. Reportes

- Reporte General con filtros en la cabecera de este y con opciones de exportar a Excel.
- Los campos para mostrar serán:
    - Nro. De serie

    - Tipo de equipo
    - Responsable
    - Área
    - Ubicación
    - Estado del activo
    - Estado funcional
    - TAG
    - MAC
    - Año
    - Etiqueta
    - Orden de Compra
    - Responsable
    - Observaciones
- El reporte solo debe sacar los datos actuales de los activos.
- Los reportes solo tendrán acceso los usuarios con el rol y permiso para reportes.

Reportes

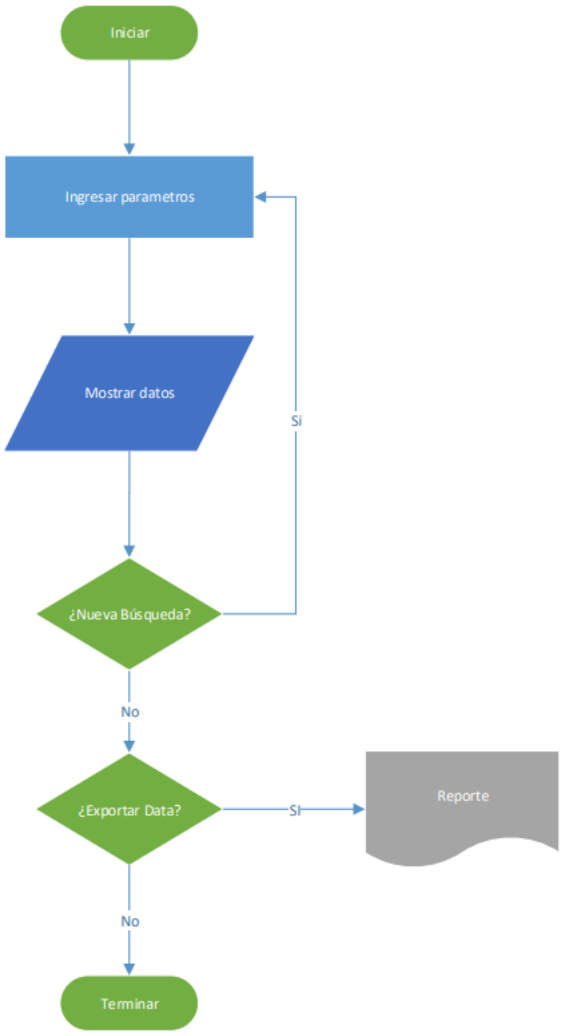


Figura 16. Diagrama de procesos reportes.
Elaborado por: Amelia Cáceres y Cristian Cazares

Capítulo 3

Análisis y desarrollo

3. Desarrollo del sistema

3.1. Tecnología y arquitectura. El sistema para desarrollar debe ser construida de acuerdo con los estándares corporativos y ser compatible como se indica en la tabla 18.

Tabla 18. Estándares corporativos y de compatibilidad.

Estándares corporativos y de compatibilidad	
Base de datos	SQL Server 2012
Herramienta de Desarrollo (IDE)	Visual Studio .NET 2015 o superior, con ASPX o C#
Navegador	Chrome 45+
	Internet Explorer 11+
Estándares de desarrollo corporativo	Almacenamientos de LOG de eventos
	Creaciones, bajas, modificaciones
	Cambios en la app

Nota: Estándares corporativos y de compatibilidad.

3.2. Bases de datos.

3.2.1. Contexto de la base de datos. Se detalla enseguida, la secuencia para crear la base de datos en el sistema a través de Code First y Visual Studio. Estas herramientas permiten guardar datos que no existen es decir Code First establece como destino una base de datos vacía a la cual se le agregará nuevas tablas y a la cual se le definirá un modelo mediante VB.net.

- Se crea el contexto que deriva de dbcontext.
- Se usa “Default” como nombre de la base de datos y crea una cadena para conexión de esta base de datos.

Contexto de la base de datos

```
1   using Datos.Modelos;
2   using System.ComponentModel.DataAnnotations.Schema;
3   using System.Data.Entity;
4
5   namespace Datos
6   {
        30 referencias | Amelia, Hace 3 días | 2 autores, 2 cambios
7       public class Contexto : DbContext
8       {
            9 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
9           public Contexto() : this("Default")
10          {
11          }
12
            2 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
13          public Contexto(string connectionString) : base(connectionString)
14          {
15              Database.SetInitializer(new MigrateDatabaseToLatestVersion<Contexto, Migraciones.Configurador>());
16          }
17
```

Figura 17. Líneas de código del contexto de la base de datos.
Elaborado por: Amelia Cáceres y Cristian Cazares

- Modelbuilder para la entidad Activos

Modelbuilder

```
87              modelBuilder.Entity<Activo>()
88                  .Map(m => m.ToTable("Activos"))
89                  .HasKey(m => m.Id).Property(m => m.Id).HasDatabaseGeneratedOption(DatabaseGeneratedOption.Identity);
90
91              modelBuilder.Entity<Activo>().Property(m => m.EstadoActivo).IsRequired();
92              modelBuilder.Entity<Activo>().Property(m => m.EstadoEquipo).IsRequired();
93              modelBuilder.Entity<Activo>().Property(m => m.Serie).IsRequired();
94              modelBuilder.Entity<Activo>().Property(m => m.Tag).IsRequired().HasMaxLength(200);
95              modelBuilder.Entity<Activo>().HasIndex(m => m.Tag).IsUnique();
96              modelBuilder.Entity<Activo>().Property(m => m.Etiqueta).IsOptional();
97              modelBuilder.Entity<Activo>().Property(m => m.MAC).IsOptional();
98              modelBuilder.Entity<Activo>().Property(m => m.EstadoActivo).IsRequired();
99              modelBuilder.Entity<Activo>().Property(m => m.EstadoEquipo).IsRequired();
100             modelBuilder.Entity<Activo>().Property(m => m.OrdenCompra).IsOptional();
101             modelBuilder.Entity<Activo>().Property(m => m.Año).IsOptional();
102             modelBuilder.Entity<Activo>().Property(m => m.Observaciones).IsOptional().HasMaxLength(200);
103
104             modelBuilder.Entity<Activo>()
105                 .HasMany(m => m.Historicos)
106                 .WithRequired(m => m.Activo)
107                 .WillCascadeOnDelete(true);
108
109             modelBuilder.Entity<Activo>()
110                 .HasOptional(m => m.Ubicacion)
111                 .WithMany(m => m.Activos);
```

Figura 18. Modelbuilder para la entidad Activos.
Elaborado por: Amelia Cáceres y Cristian Cazares

- Define el modelo de activos para el contexto que se está usando.

- DBSets de las entidades

DBSets de las entidades

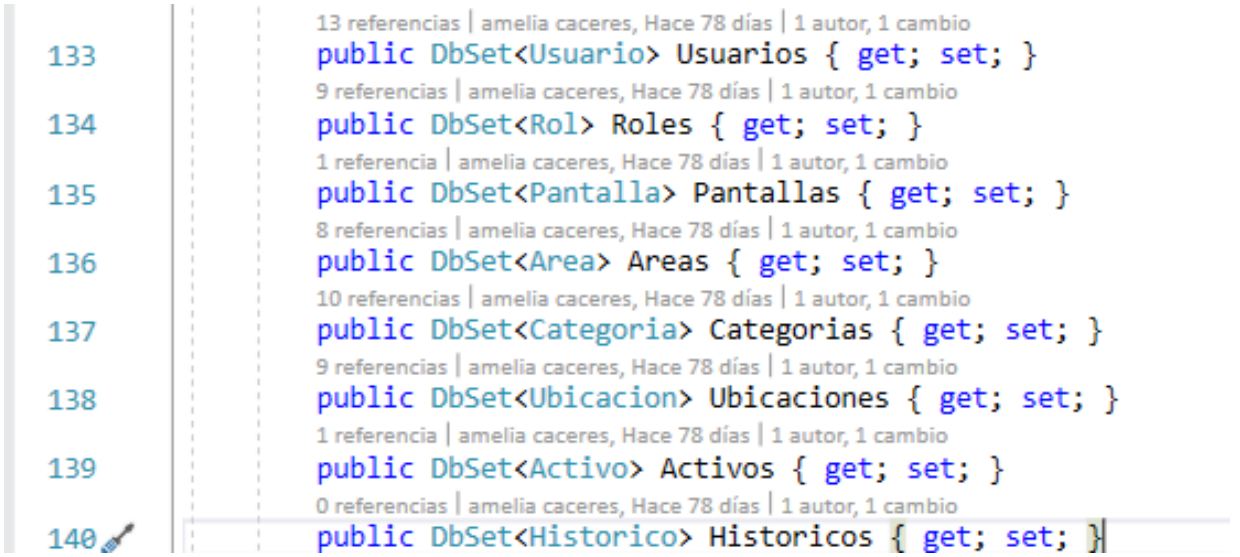


Figura 19. DBSets para entidades.
Elaborado por: Amelia Cáceres y Cristian Cazares

- Permiten acceder a las operaciones CRUD de las entidades.

3.2.2. Estándar para tablas. A continuación, se muestra un ejemplo de cómo están estructuradas las relaciones de la base de datos. Por siguiente, la tabla desplegada es la de activos, el nombre de las tablas iniciará con mayúscula, cada campo de las tablas tendrá un nombre que iniciará con mayúscula y serán nombres relacionados al entorno de la empresa y el tamaño de caracteres no será mayor a 35, además se debe especificar el tipo de datos (string, int, etc).

Tabla activos

```
14
                27 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
15              public string Serie { get; set; }
                6 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
16              public string Etiqueta { get; set; }
                20 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
17              public string Tag { get; set; }
                37 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
18              public virtual Categoria Categoria { get; set; }
                6 referencias | Amelia, Hace 3 días | 2 autores, 2 cambios
19              public string MAC { get; set; }
                22 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
20              public EstadoActivo EstadoActivo { get; set; }
                21 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
21              public EstadoEquipo EstadoEquipo { get; set; }
                6 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
22              public string OrdenCompra { get; set; }
                10 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
23              public int Año { get; set; }
                20 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
24              public virtual Area Area { get; set; }
                31 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
25              public virtual Ubicacion Ubicacion { get; set; }
                33 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
26              public virtual Usuario Responsable { get; set; }
                8 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
27              public string Observaciones { get; set; }
                0 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
28              public DateTime? UltimoCambio { get; set; }
                10 referencias | amelia caceres, Hace 78 días | 1 autor, 1 cambio
29              public virtual ICollection<Historico> Historicos { get; set; }
```

Figura 20. Estándar para tablas.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.3. Diagrama Entidad – Relación. El diagrama del sistema de gestión de activos entidad – relación representa las entidades y relaciones que se utilizaron para la estructura y creación de la base de datos.

Diagrama entidad – relación

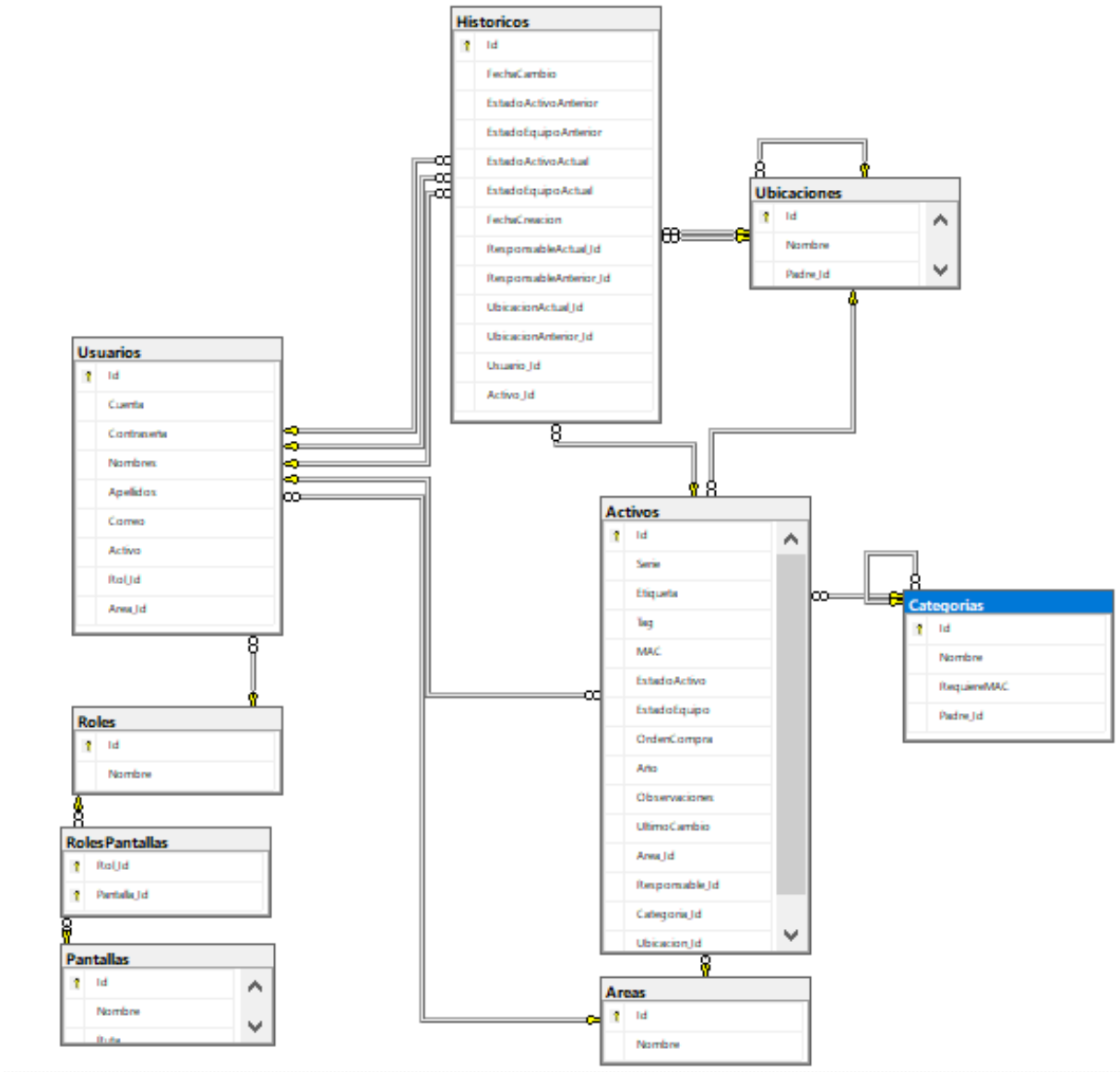


Figura 21. Diagrama entidad – relación del sistema.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.4. Diccionario de datos. A continuación, se detalla la descripción de cada atributo de las entidades de la base de datos.

3.4.1. Entidad MigrationHistory. En la tabla 19, se presenta el histórico de las migración o cambios que sufre la base de datos ya que al momento de trabajar con .net Framework para el desarrollo, tiene la herramienta de migraciones.

Tabla 19. Entidad MigrationHistory

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
MigrationId	nvarchar	300	NO	NO	Identificador único para la migración.
ContextKey	nvarchar	600	NO	NO	Cadena para distinguir las migraciones de otras configuraciones de migraciones que utilizan la misma base de datos.
Model	varbinary	-1	NO	NO	Cadena XML comprimida con el modelo de destino de la migración.
ProductVersion	nvarchar	64	NO	NO	Es el número de versión de Entity Framework utilizado para hacer la migración.

Nota: Tabla de la entidad MigrationHistory.

3.4.2. Entidad de Activo. En la tabla 20, se presenta todos los atributos que se necesitan para almacenar un activo en el sistema.

Tabla 20. Entidad de Activo

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único para el activo
Serie	nvarchar	-1	NO	NO	Código alfanumérico que representa el número de serie de fábrica del activo
Etiqueta	nvarchar	-1	SI	NO	Texto para indicar una clasificación o valoración del activo

Tag	nvarchar	400	NO	NO	Código alfanumérico que sirve para identificar al activo relacionado con la empresa
MAC	nvarchar	-1	SI	NO	Identificador que corresponde al activo
EstadoActivo	smallint	2	NO	NO	Enumeración que sirve para representar el estado en que se encuentra el activo
OrdenCompra	nvarchar	-1	SI	NO	Número de la orden de compra con la que se adquirió el activo
Año	int	4	SI	NO	Año del activo
Observaciones	nvarchar	400	SI	NO	Texto para indicar algún tipo de observación relacionada al activo
UltimoCambio	datetime	8	SI	NO	Fecha del último cambio q tuvo el activo
Area_Id	int	4	SI	NO	Clave secundaria del área a la que pertenece el activo
Responsable_Id	int	4	SI	NO	Clave secundaria del responsable que tiene a cargo el activo
Categoria_Id	int	4	NO	NO	Clave secundaria que identifica a qué tipo de categoría (equipo) pertenece el activo
Ubicación_Id	int	4	SI	NO	Clave secundaria de la ubicación de donde se encuentra el activo
EstadoFuncional	smallint	2	NO	NO	Enumeración que sirve para representar el estado funcional del activo

Nota: Tabla de la entidad de activo.

3.4.3. Entidad Áreas. En la tabla 21, se presenta todos los atributos que se necesitan para las áreas envueltas en el desarrollo de la gestión de activo de TI.

Tabla 21. Entidad Áreas

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único para el área
Nombre	nvarchar	-1	SI	NO	Texto con el nombre del área

Nota: Tabla de la entidad Áreas.

3.4.4. Entidad Categorías. En la tabla 22, se presenta todos los atributos que se necesitan para las categorías de cada activo en el sistema.

Tabla 22. Entidad Categorías

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único para las categorías
Nombre	nvarchar	-1	SI	NO	Texto con el nombre de la categoría
RequiereMAC	bit	1	NO	NO	Indica si el activo debe tener MAC
Padre_Id	int	4	SI	NO	Indica si la categoría está inmersa en otra categoría

Nota: Tabla de la entidad Categorías.

3.4.5. Entidad Históricos. En la tabla 23, se presenta todos los atributos que se necesitan para los históricos de los activos al momento de cambiar de estado en el sistema.

Tabla 23. Entidad Históricos

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único del histórico de cambios

FechaCambio	datetime	8	NO	NO	Fecha del último cambio q tuvo el activo
EstadoActivoAnterior	smallint	2	NO	NO	Enumeración que representa el estado anterior del activo
EstadoActivoActual	smallint	2	NO	NO	Enumeración que representa el estado actual del activo
FechaCreacion	datetime	8	NO	NO	Fecha de cuando se creó el histórico de cambio
ResponsableActual_Id	int	4	SI	NO	Clave secundaria del usuario que es responsable del activo
ResponsableAnterior_Id	int	4	SI	NO	Clave secundaria del usuario que fue responsable del activo en el anterior estado
UbicacionActual_Id	int	4	SI	NO	Clave secundaria de la ubicación donde se encuentra el activo
UbicacionAnterior_Id	int	4	SI	NO	Clave secundaria de la ubicación anterior del activo
Usuario_Id	int	4	NO	NO	Clave secundaria que identifica al usuario que realizó el último cambio
Activo_Id	int	4	NO	NO	Clave secundaria que identifica al activo del histórico
EstadoFuncionalAnterior	smallint	2	NO	NO	Enumeración que representa el estado funcional anterior del activo
EstadoFuncionalActual	smallint	2	NO	NO	Enumeración que representa el estado funcional actual del activo

Nota: Tabla de la entidad Históricos.

3.4.6. Entidad Pantallas. En la tabla 24, se presenta todos los atributos que se necesitan para la configuración de las pantallas al momento de administrar los roles en el sistema.

Tabla 24. Entidad Pantallas

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único de registro de pantallas
Nombre	nvarchar	-1	SI	NO	Texto con el nombre de la pantalla
Ruta	nvarchar	-1	SI	NO	Texto con la ruta navegacional de la pantalla

Nota: Tabla de la entidad Pantallas.

3.4.7. Entidad Roles. En la tabla 25, se presenta todos los atributos que se necesitan los roles que se le asigna a cada usuario en el sistema.

Tabla 25. Entidad Roles

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único de registro de roles
Nombre	nvarchar	-1	SI	NO	Texto con el nombre del rol

Nota: Tabla de la entidad Roles.

3.4.8. Entidad RolesPantallas. En la tabla 26, se presenta la relación varios a varios entre roles y pantallas para poder asignar pantallas a cada rol en el sistema.

Tabla 26. Entidad RolesPantallas

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Rol_Id	int	4	NO	NO	Clave secundaria que identifica el rol
Pantalla_Id	int	4	NO	NO	Clave secundaria que identifica la pantalla

Nota: Tabla de la entidad Roles.

3.4.9. Entidad Ubicaciones. En la tabla 27, se presenta todos los atributos que se necesitan las ubicaciones que se le asigna a cada activo en el sistema.

Tabla 27. Entidad Ubicaciones

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único de registro de ubicaciones
Nombre	nvarchar	-1	SI	NO	Texto con el nombre de la ubicación
Padre_Id	int	4	SI	NO	Indica si la ubicación está inmersa en otra ubicación

Nota: Tabla de la entidad Ubicaciones.

3.4.10. Entidad Usuarios. En la tabla 28, se presenta todos los atributos que se necesitan los usuarios que se le asigna a cada activo y utilizan el sistema.

Tabla 28. Entidad Usuarios

Columna	Tipo de dato	Tamaño	Permite Nulos	Es identidad	Descripción
Id	int	4	NO	SI	Identificador único de registro de usuario
Cuenta	nvarchar	-1	SI	NO	Texto con la cuenta del usuario con la que inicia sesión en el sistema
Contraseña	nvarchar	-1	SI	NO	Texto cifrado con la contraseña que se le asignó al usuario
Nombres	nvarchar	-1	SI	NO	Texto con el nombre del usuario
Apellidos	nvarchar	-1	SI	NO	Texto con el apellido del usuario
Correo	nvarchar	-1	SI	NO	Texto con el correo del usuario a donde le llegarán las notificaciones
Activo	bit	1	NO	NO	Indica si el usuario está activo o no, para poder ingresar al sistema
Rol_Id	int	4	NO	NO	Clave secundaria que identifica el rol que tiene el usuario

Area_Id	int	4	NO	NO	Clave secundaria que identifica a que área pertenece el usuario dentro de la empresa

Nota: Tabla de la entidad Usuarios.

3.5. Diagramas para la implementación.

3.5.1. Diagrama de clases. La arquitectura en que está desarrollado nuestro proyecto es por medio de capas donde consta con la capa de datos, lógica y web. Los diagramas de clases de la aplicación web son los siguientes.

3.5.1.1. Capa de datos. En la Figura 22., se presenta el diagrama de clases en cuya capa de datos están las clases de los modelos de datos que nos ayuda a interactuar la aplicación con la base de datos.

Diagrama de clases capa de datos

Figura 22. Diagrama de clases capa de datos.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.5.1.2. Capa lógica. En la Figura 23., se presenta el diagrama de clases cuya capa lógica tiene las clases del Core de la aplicación con los métodos más importantes donde realiza todo el

procesamiento de la información en la aplicación, esta capa se conecta con la capa de datos para poder procesar la información que se tiene en la base de datos.

Diagrama de clases capa lógica

Figura 23. Diagrama de clases capa lógica.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.5.1.3. Capa de web. En la Figura 24., se presenta el diagrama de clases cuya capa web tiene las clases principales que son los controladores de cada vista para ser presentado en la web, esta

capa es en la que se relaciona el usuario final con la aplicación, además se conecta con la capa lógica para procesar lo que ingresa el usuario.

Diagrama de clases capa web

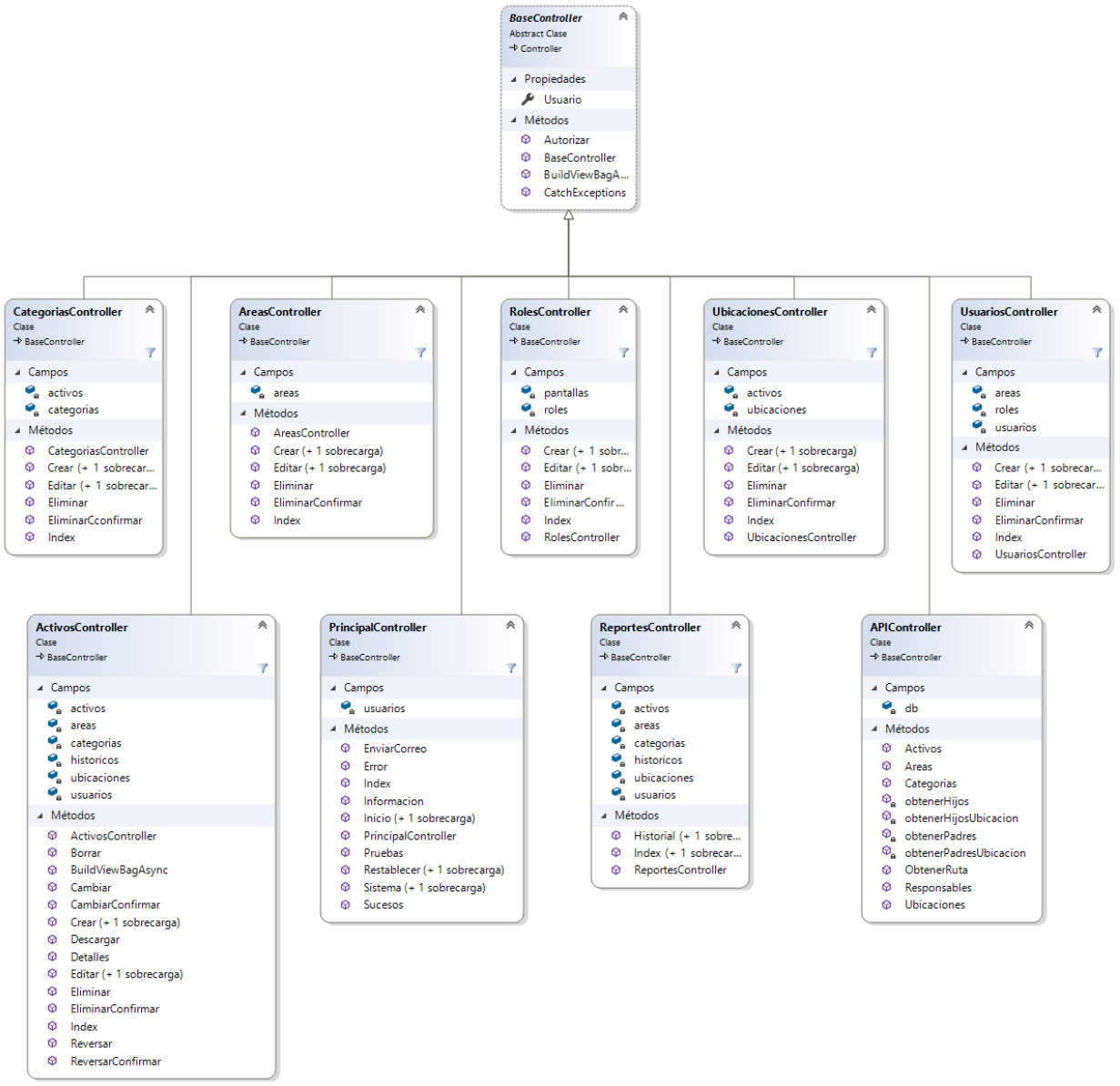


Figura 24. Diagrama de clases capa web.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.5.1.4. Diagrama de navegación. La navegación que cuenta nuestra aplicación web es de una pantalla de inicio donde se tiene 4 menús principales y 8 submenús, donde el acceso total tiene el

administrador y, como la aplicación cuenta con un módulo de gestor de usuarios y roles, se le puede asignar cada una de las pantallas del menú y submenús a cada usuario de acuerdo a sus necesidades.

Diagrama de navegación aplicación web

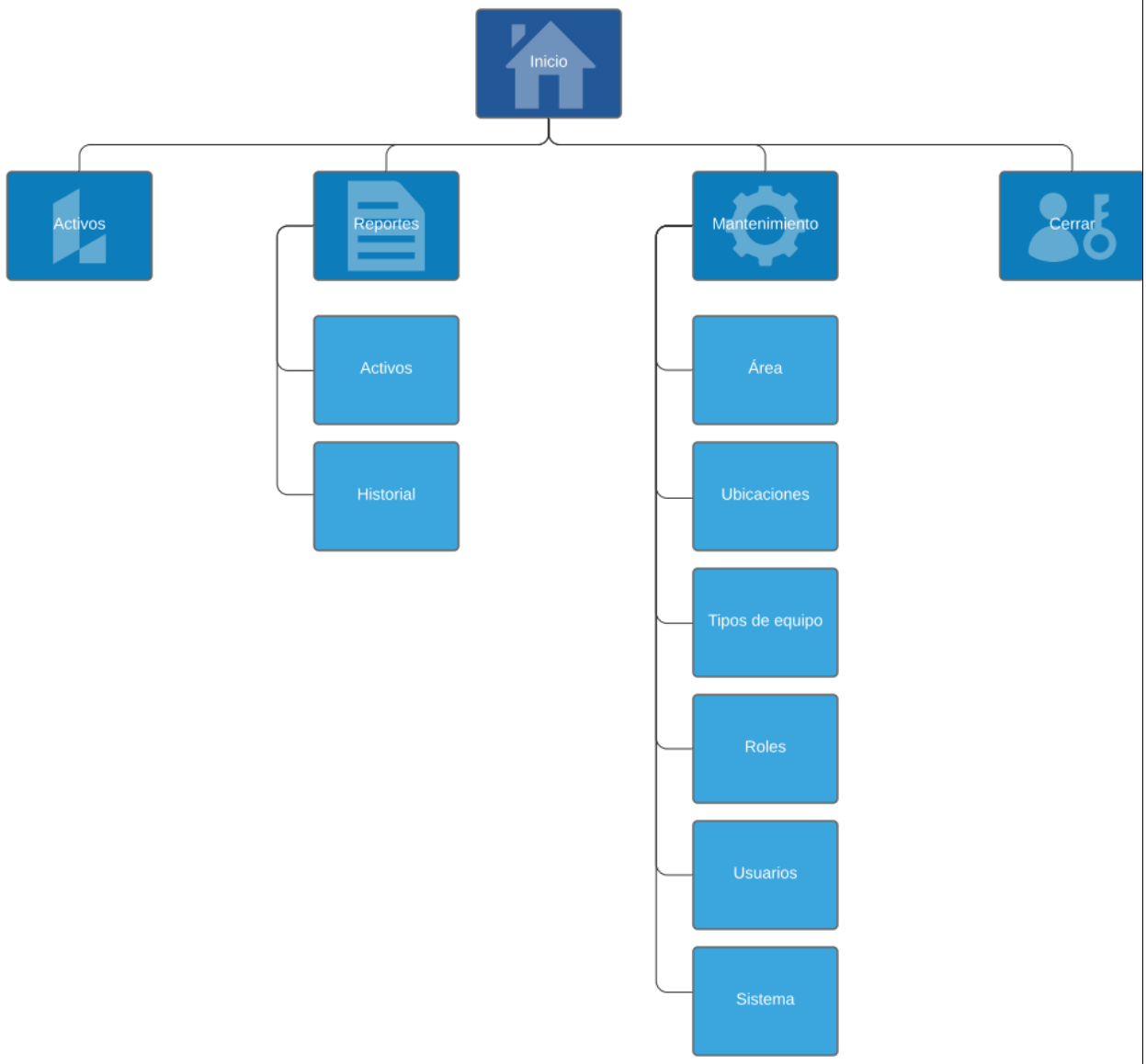


Figura 25. Diagrama de navegación aplicación web.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.5.2. Diagrama de secuencias. A continuación, los diagramas a detallar en esta parte son de dos acciones importantes que realiza el usuario que son crear un activo y modificar el estado del activo.

3.5.2.1. Crear un activo. En la figura 26, tenemos la representación en un diagrama de secuencia cómo se realiza el ingreso de un nuevo activo y representar la interacción entre la vista, controlador y la base de datos.

Diagrama de secuencia crear un activo

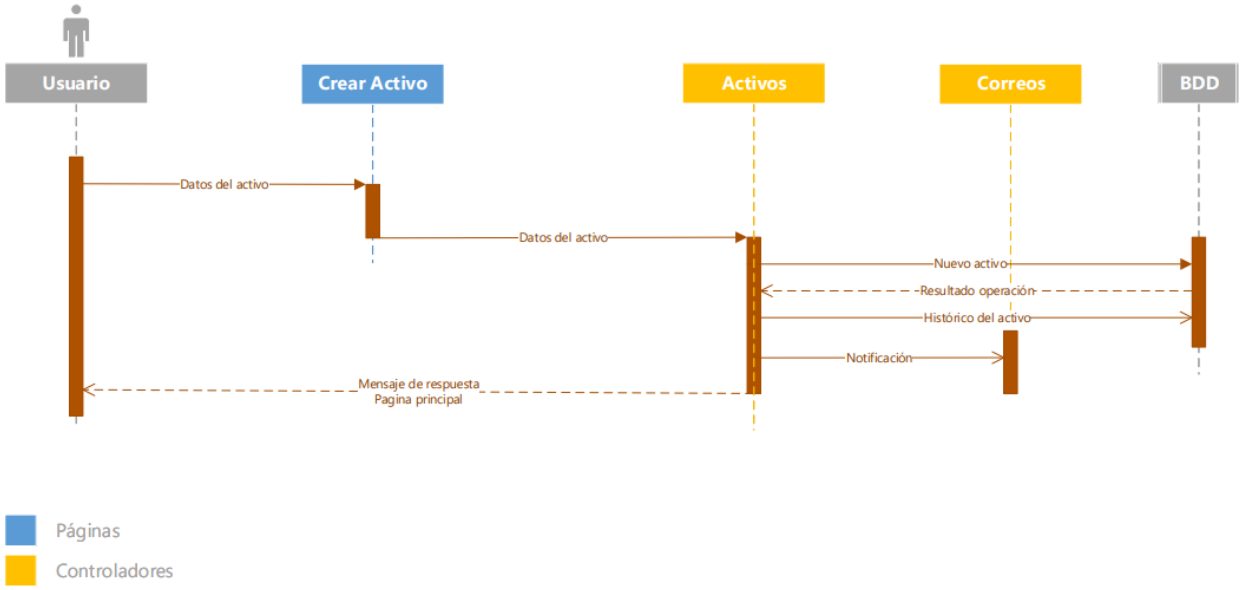


Figura 26. Diagrama de secuencia crear un activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.5.2.2. Modificar estado de un activo. En la figura 27, tenemos la representación en un diagrama de secuencia cómo se realiza el ingreso de un nuevo activo y representar la interacción entre la vista, controlador y la base de datos.

Diagrama de secuencia crear un activo

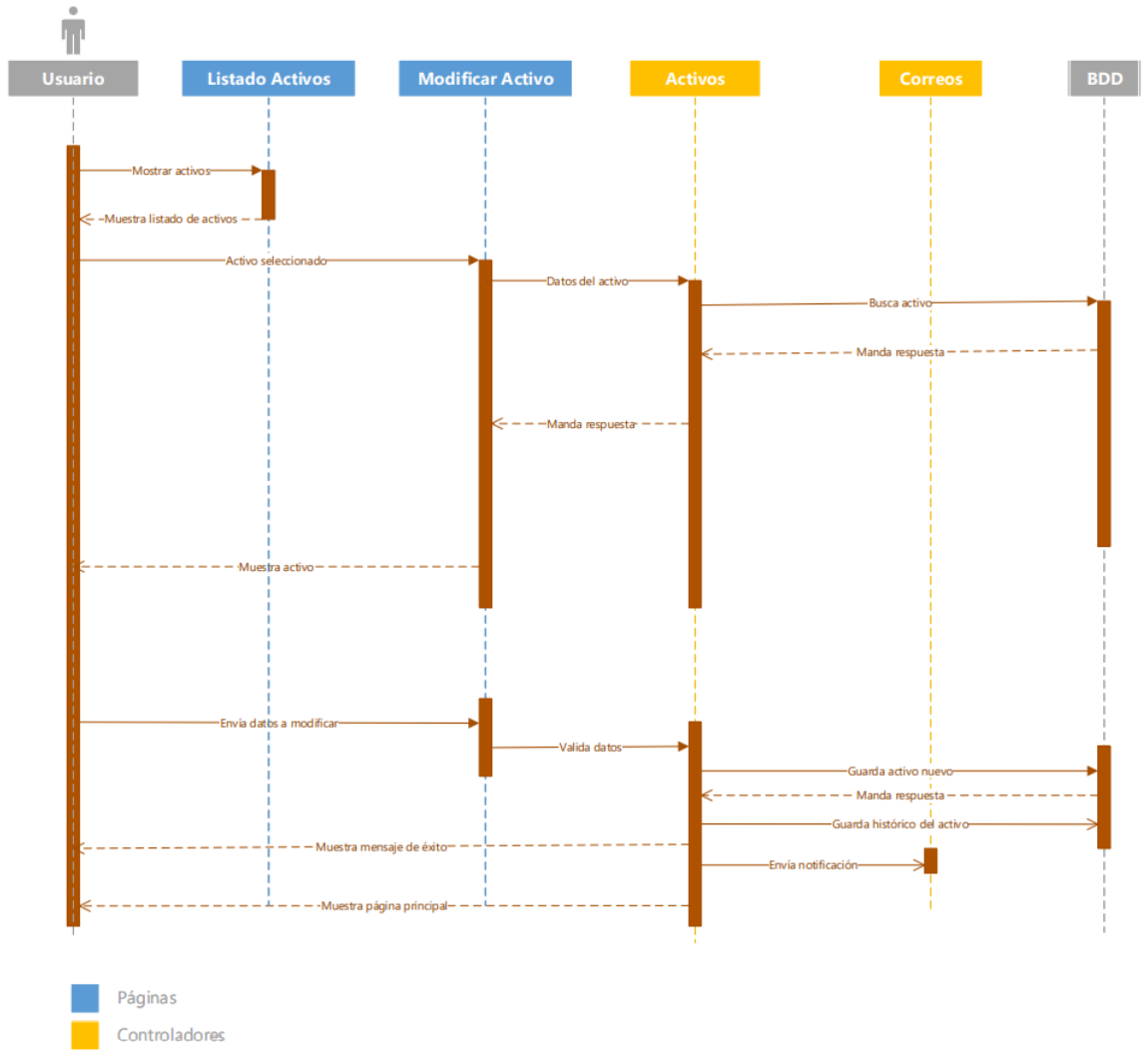


Figura 27. Diagrama de secuencia crear un activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.6. Código del sistema. A continuación, se muestra el código para la construcción del sistema tanto en la capa lógica como en la capa web.

3.7. Crear Usuarios.

3.7.1. "Crear". Método Post para la creación de un usuario (Capa Web). En este método la información llega proporcionada por el usuario desde el controlador en la interfaz de usuario, como el nombre, apellidos, cuenta, correo, área, rol, y si el usuario está activo o no. En la Figura 28, se describen en las líneas 109,110,111 la asignación al usuario una contraseña hash aleatoria, en la

línea 113 se hace una solicitud a la capa lógica para aplicar las reglas de creación de usuario, en la línea 114 se hace una solicitud a la capa lógica para aplicar las reglas de envío de correo para la creación de usuario y en la línea 117 me redirección a la vista de administración de usuarios.

Método para crear usuarios

```
90          [HttpPost]
91          [ValidateAntiForgeryToken]
            0 referencias | amelia caceres, Hace 6 horas | 1 autor, 2 cambios
92          public async Task<ActionResult> Crear(UsuarioViewModel modelo)
93          {
94              base.Autorizar();
95              try
96              {
97                  if (ModelState.IsValid)
98                  {
99                      var registro = new Usuario
100                     {
101                         Nombres = modelo.Nombres,
102                         Apellidos = modelo.Apellidos,
103                         Cuenta = modelo.Cuenta,
104                         Correo = modelo.Correo,
105                         Activo = modelo.Activo,
106                         Area = await areas.FindAsync(modelo.IdArea),
107                         Rol = await roles.FindAsync(modelo.IdRol),
108                     };
109                     var contraseña = Guid.NewGuid().ToString().Replace("-", "").Substring(0, 10);
110                     var pasHas = System.Web.Helpers.Crypto.HashPassword(contraseña);
111                     registro.Contraseña = pasHas;
112
113                     await usuarios.AddAsync(registro);
114                     usuarios.EnviarCorreo(registro, contraseña);
115
116                     Session["Success"] = true;
117                     return RedirectToAction("Index");
```

Figura 28. Método para crear usuarios.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.7.1.1. "usuarios.AddAsync". Método con las reglas de validación para agregar un usuario (Capa Lógica). Este método valida que no exista otro usuario con la misma cuenta ingresada, que el usuario tenga un rol asignado, que el usuario tenga un área asignada y que tenga nombre y apellido.

En la línea 191 se hace una solicitud a la capa datos para la inserción del usuario en la base de datos.

Método para crear reglas de validación de usuarios

```
25 referencias | amelia caceres, Hace 77 días | 1 autor, 1 cambio
178     public override async Task AddAsync(Usuario registro)
179     {
180         if (await AnyAsync(m => m.Cuenta.Trim() == registro.Cuenta.Trim()))
181             throw new Exception($"Ya se encuentra registrada la cuenta {registro.Cuenta}");
182         if (registro.Rol == null || registro.Rol.Id == 0)
183             throw new Exception($"Debe seleccionar un rol para el usuario");
184         if (registro.Area == null || registro.Area.Id == 0)
185             throw new Exception($"Debe seleccionar un área para el usuario");
186         if (string.IsNullOrWhiteSpace(registro.Nombres))
187             throw new Exception($"El campo Nombres es obligatorio");
188         if (string.IsNullOrWhiteSpace(registro.Apellidos))
189             throw new Exception($"El campo Apellidos es obligatorio");
190
191         await base.AddAsync(registro);
192     }
```

Figura 29. Método para crear reglas de validación de usuarios.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.8. Eliminar Usuario.

3.8.1. "Eliminar". Método Post para la eliminación de un usuario (Capa Web). A este método del controlador llega el Id del usuario a eliminar, en la línea 210 se realiza una solicitud a la capa lógica para buscar el usuario en la base de datos según su Id, en la línea 211 se hace una solicitud a la capa lógica para aplicar las reglas de eliminación de usuario.

Método para eliminar usuarios

```
202     [HttpPost, ActionName("Eliminar")]
203     [ValidateAntiForgeryToken]
        0 referencias | amelia caceres, Hace 77 días | 1 autor, 1 cambio
204     public async Task<ActionResult> EliminarConfirmar(int id)
205     {
206         base.Autorizar();
207
208         try
209         {
210             var registro = await usuarios.FindAsync(id);
211             await usuarios.RemoveAsync(registro);
212             Session["Success"] = true;
213         }
```

Figura 30. Método para eliminar usuarios.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.8.1.1. "usuarios.RemoverAsync". Método con las reglas de validación para eliminar un usuario (Capa Lógica). Este método valida que no se elimine el usuario con Id = 1, que no se pueda eliminar un usuario que tenga un activo a su nombre, que no tenga historial de activos, y que no se eliminen todos los usuarios. En la línea 243 se hace una solicitud a la capa datos para la eliminación del usuario en la base de datos.

Método con reglas para eliminar usuarios

```
19 referencias | amelia caceres, Hace 77 días | 1 autor, 1 cambio
227 public override async Task RemoveAsync(Usuario registro)
228 {
229     if (registro.Id == 1)
230         throw new Exception($"No se puede eliminar el usuario {registro.Cuenta} por que está siendo utilizado por el sistema.");
231
232     var activos = await db.Set<Activo>().AnyAsync(m => m.Responsable.Id == registro.Id);
233     if (activos)
234         throw new Exception($"No se puede eliminar un usuario que tenga activos asignados.");
235
236     var historicos = await db.Set<Historico>().AnyAsync(m => m.ResponsableAnterior.Id == registro.Id || m.ResponsableActual.Id == registro.Id);
237     if (historicos)
238         throw new Exception($"No se puede eliminar un usuario que tenga historial de activos asignados.");
239
240     if (await db.Set<Usuario>().CountAsync() == 1)
241         throw new Exception("No se pueden eliminar todos los usuarios.");
242
243     await base.RemoveAsync(registro);
244 }
```

Figura 31. Método con reglas para eliminar usuarios.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.9. Crear registro de un activo

3.9.1. "Crear". Método Post para la creación de un activo (Capa Web). El usuario le proporciona a este método la información necesaria para la creación de un activo en la interfaz de creación de un activo. En la línea 243 se hace una solicitud a la capa lógica para aplicar las reglas de creación de activo, en la línea 245 se hace una solicitud a la capa lógica para aplicar las reglas de almacenar un activo, en la línea 247 a 275 verifica que el formulario tenga un archivo para poder guardarlo en una ruta física.

Método para el registro de activos

```
243                 await activos.AddAsync(registro);
244
245                 activos.Almacenar(registro.Id, registro.EstadoEquipo, registro.Ubicacion.Id, User.GetId(), true);
246
247                 if (Request.Files.Count > 0)
248                     for (int i = 0; i < Request.Files.Count; i++)
249                     {
250                         var file = Request.Files.Get(i);
251                         if (file != null && file.ContentLength > 0)
252                         {
253                             var fileName = Path.GetFileName(file.FileName);
254                             var path = Server.MapPath($"{Application.RutaAdjuntos}{registro.Id}");
255                             try
256                             {
257                                 if (!Directory.Exists(path))
258                                     try
259                                     {
260                                         Directory.CreateDirectory(path);
261                                     }
262                                     catch (Exception ex)
263                                     {
264                                         Logica.Programa.Logger.Error(ex);
265                                     }
266                                 path = Path.Combine(path, fileName);
267                                 file.SaveAs(path);
268                             }
269                             catch (Exception ex)
270                             {
271                                 Logica.Programa.Logger.Error(ex);
272                             }
273                         }
274                     }
275
```

Figura 32. Método para el registro de activos.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.9.1.1. "activos.AddAsync". Método con las reglas de validación para agregar un activo (Capa Lógica). Este método valida que no exista otro activo con el mismo Tag, con el mismo número de serie, que tenga ubicación asignada, área asignada, que pertenezca a una categoría, y que si el estado es asignado que tenga un responsable. En la línea 44 se hace una solicitud a la capa datos para la inserción del activo en la base de datos.

Método con reglas para validación del registro de un activo

```
25 referencias | amelia caceres, Hace 77 días | 1 autor, 1 cambio
24  public override async Task AddAsync(Activo registro)
25  {
26      if (registro.EstadoActivo == Datos.Tipos.EstadoActivo.Asignado && registro.Responsable?.Id == 0)
27          throw new Exception($"Debe seleccionar un responsable.");
28
29      if (await db.Set<Activo>().AnyAsync(m => m.Tag == registro.Tag))
30          throw new Exception($"Ya se encuentra registrado un activo con el TAG {registro.Tag}.");
31
32      if (await db.Set<Activo>().AnyAsync(m => m.Serie == registro.Serie))
33          throw new Exception($"Ya se encuentra registrado un activo con la serie {registro.Serie}.");
34
35      if (registro.Ubicacion == null || registro.Ubicacion.Id == 0)
36          throw new Exception($"Debe seleccionar una ubicación.");
37
38      if (registro.Area == null || registro.Area.Id == 0)
39          throw new Exception($"Debe seleccionar un área.");
40
41      if (registro.Categoria == null || registro.Categoria.Id == 0)
42          throw new Exception($"Debe seleccionar tipo de equipo.");
43
44      await base.AddAsync(registro);
45      Programa.Logger.Info($"**********Activo {registro.Tag} creado**********");
46  }
```

Figura 33. Método con reglas de validación para registro de un activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.10. Eliminar un activo

3.10.1. "Eliminar". Método Post para la eliminación de un activo (Capa Web). A este método, del controlador llega el Id del activo a eliminar. En la línea 498 se hace una solicitud a la capa lógica para buscar el activo en la base de datos según su Id, en la línea 499 se hace una solicitud a la capa lógica para aplicar las reglas de eliminación de activo, en la línea 501 a 510 se elimina el archivo del activo en caso de que existiera.

Método para la eliminación de un activo

```
490         [HttpPost, ActionName("Eliminar")]
491         [ValidateAntiForgeryToken]
492         [AuthorizeRoute("Activos", "Crear")]
            0 referencias | amelia caceres, Hace 77 días | 1 autor, 1 cambio
493         public async Task<ActionResult> EliminarConfirmar(int id)
494         {
495             base.Autorizar();
496             try
497             {
498                 var registro = await activos.FindAsync(id);
499                 await activos.RemoveAsync(registro);
500                 Session["Success"] = true;
501                 var path = Server.MapPath($"~/App_Data/Adjuntos/{id}");
502                 if (Directory.Exists(path))
503                     try
504                     {
505                         Directory.Delete(path, true);
506                     }
507                     catch (Exception iex)
508                     {
509                         Logica.Programa.Logger.Error(iex);
510                     }
511             }
```

Figura 34. Método para la eliminación de un activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.10.1.1. "activos.RemoveAsync". Método con las reglas de validación para eliminar un activo (Capa Lógica). En la línea 81, se hace una solicitud a la capa datos para la eliminación del activo en la base de datos y en la línea 82 se guarda en el archivo de log un mensaje informativo sobre la eliminación del registro.

Método con reglas para eliminar un activo

```
            19 referencias | amelia caceres, Hace 77 días | 1 autor, 1 cambio
79          public override async Task RemoveAsync(Activo registro)
80          {
81              await base.RemoveAsync(registro);
82              Programa.Logger.Info($"***************Activo {registro.Tag} eliminado***************");
83          }
84
```

Figura 35. Método con reglas para eliminar un activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.11. Cambiar estado del activo. Los estados que podrá tener el activo son: Almacenado, Asignado, Chatarrizado, Préstamo, Extraviado, Reparación.

3.11.1. Cambios de Estado. Se ha definido que algunos estados para ser cambiados deben cumplir ciertas condiciones.

Tabla 29. Cambio de estado

Estado	Condición	Observación

Préstamo	Solo cuando el activo este en estado Almacenado.	Para poder cambiar a préstamo y el activo no esté en almacenado, se requerirá hacer un doble cambio de estado, guardando en cada caso los históricos.

Nota: Tabla estado del equipo.

3.11.2. Información necesaria para la gestión de activos. A continuación, se muestra un esquema del sistema requerido, en donde existen campos de llenado por los usuarios.

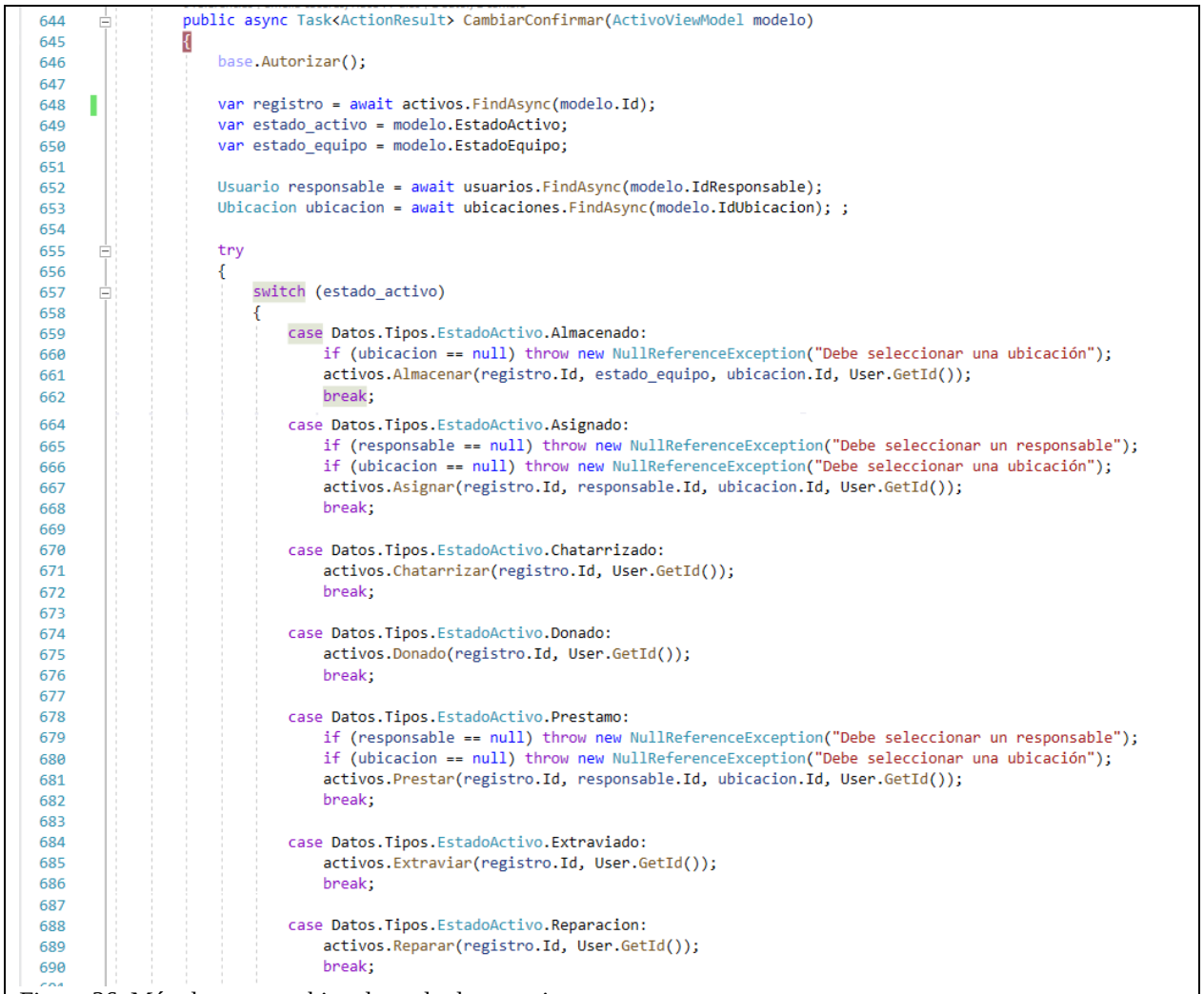
Tabla 30. Información para ingresar como activos

Campos de llenado	**Observaciones**
Información del activo	
Nro. de serie	Código alfanumérico único de cada activo. OBLIGATORIO
Tag	Código alfanumérico único de cada activo. OBLIGATORIO
Etiqueta	Código alfanumérico. NO Indispensable
Tipo Equipo	Alfanumérico, Indispensable (Lista desplegable).
Ubicación	Campo Indispensable (Lista desplegable).
MAC	Código alfanumérico, no obligatorio.
Estado Activo	Campo Indispensable (Lista desplegable).
Orden de Compra	Numérico No Obligatorio
Estado Funcional	Campo Indispensable (Lista desplegable).
Responsable	Campo Indispensable (Lista desplegable).
Adjuntos	Carga de documentos No Obligatorio
Área Responsable	Campo Indispensable (Lista desplegable).
Observaciones	Alfanumérico 200 caracteres. No obligatorio.

Nota: Tabla de información a ingresar como activos.

3.11.2.1. "Estado Activo". Método Post para la cambiar de estado de un activo (Capa Web). En la línea 648, se hace una solicitud a la capa lógica para buscar el activo en la base de datos según su Id, en la línea 652 se hace una solicitud a la capa lógica para encontrar aquel usuario en la base de datos según su Id, en la línea 653 se hace una solicitud a la capa lógica para buscar la ubicación en la base de datos según su Id, en la línea 657 a 694 según la opción de estado de activo ingresada por el usuario, se hará una solicitud a la capa de lógica para aplicar las reglas de cambio de estado.

Método para cambiar el estado de un activo



```
644         public async Task<ActionResult> CambiarConfirmar(ActivoViewModel modelo)
645         {
646             base.Autorizar();
647
648             var registro = await activos.FindAsync(modelo.Id);
649             var estado_activo = modelo.EstadoActivo;
650             var estado_equipo = modelo.EstadoEquipo;
651
652             Usuario responsable = await usuarios.FindAsync(modelo.IdResponsable);
653             Ubicacion ubicacion = await ubicaciones.FindAsync(modelo.IdUbicacion); ;
654
655             try
656             {
657                 switch (estado_activo)
658                 {
659                     case Datos.Tipos.EstadoActivo.Almacenado:
660                         if (ubicacion == null) throw new NullReferenceException("Debe seleccionar una ubicación");
661                         activos.Almacenar(registro.Id, estado_equipo, ubicacion.Id, User.GetId());
662                         break;
664                     case Datos.Tipos.EstadoActivo.Asignado:
665                         if (responsable == null) throw new NullReferenceException("Debe seleccionar un responsable");
666                         if (ubicacion == null) throw new NullReferenceException("Debe seleccionar una ubicación");
667                         activos.Asignar(registro.Id, responsable.Id, ubicacion.Id, User.GetId());
668                         break;
669
670                     case Datos.Tipos.EstadoActivo.Chatarrizado:
671                         activos.Chatarrizar(registro.Id, User.GetId());
672                         break;
673
674                     case Datos.Tipos.EstadoActivo.Donado:
675                         activos.Donado(registro.Id, User.GetId());
676                         break;
677
678                     case Datos.Tipos.EstadoActivo.Prestamo:
679                         if (responsable == null) throw new NullReferenceException("Debe seleccionar un responsable");
680                         if (ubicacion == null) throw new NullReferenceException("Debe seleccionar una ubicación");
681                         activos.Prestar(registro.Id, responsable.Id, ubicacion.Id, User.GetId());
682                         break;
683
684                     case Datos.Tipos.EstadoActivo.Extraviado:
685                         activos.Extraviar(registro.Id, User.GetId());
686                         break;
687
688                     case Datos.Tipos.EstadoActivo.Reparacion:
689                         activos.Reparar(registro.Id, User.GetId());
690                         break;
```

Figura 36. Método para cambiar el estado de un activo
Elaborado por: Amelia Cáceres y Cristian Cazares

3.11.2.2. "activos.Prestar". Método con las reglas de validación para cambiar a estado Prestado a un activo (Capa Lógica). En la línea 424 a 425 se guarda en el archivo de log un mensaje informativo sobre el inicio del proceso de donado, en la línea 427 se hace una solicitud a la capa de datos para buscar el activo según el id, en la línea 428 se hace una solicitud a la capa de datos para buscar la ubicación según el id, en la línea 429 se hace una solicitud a la capa de datos para buscar el usuario responsable según el id, en la línea 433 a 449 se aplica la lógica para cambiar de estado al activo con su histórico de cambios y su nueva ubicación, responsable, estado del activo, estado del equipo.

En la línea 450 a 451 se hace una solicitud a la capa datos para la actualización del activo en la base.

Método de comprobar el cambio de estado de un activo

```
424             Programa.Logger.Info($"**********Prestar**********");
425             Programa.Logger.Info($"Proceso iniciado.");
426
427             var registro = db.Set<Activo>().Find(modelo);
428             var _ubicacion = db.Set<Ubicacion>().Find(ubicacion);
429             var _responsable = db.Set<Usuario>().Find(responsable);
430
431             Programa.Logger.Info($"Preparando activo.");
432
433             registro.Historicos.Add(new Historico()
434             {
435                 EstadoActivoAnterior = registro.EstadoActivo,
436                 EstadoActivoActual = EstadoActivo.Prestamo,
437                 EstadoEquipoActual = registro.EstadoEquipo, //no segura, no se especifica
438                 EstadoEquipoAnterior = registro.EstadoEquipo,
439                 UbicacionAnterior = registro.Ubicacion,
440                 UbicacionActual = _ubicacion,
441                 FechaCambio = DateTime.Now,
442                 ResponsableAnterior = registro.Responsable,
443                 ResponsableActual = _responsable,
444                 Usuario = db.Set<Usuario>().Find(usuario),
445             });
446
447             registro.Responsable = db.Set<Usuario>().Find(responsable);
448             registro.EstadoActivo = EstadoActivo.Prestamo;
449             registro.Ubicacion = _ubicacion;
450             db.Set<Activo>().AddOrUpdate(registro);
451             result = db.SaveChanges() > 0;
```

Figura 37. Método para validar el cambio de estado de un activo.
Elaborado por: Amelia Cáceres y Cristian Cazares

3.12. Pruebas del sistema. A continuación, se describen los resultados de las pruebas de carga y stress realizadas al sistema web, con la finalidad de comprobar el correcto funcionamiento y rendimiento del sistema.

3.13. Pruebas de carga y stress. Para realizar estas pruebas se utilizó el software "Jmeter" que trabaja bajo el entorno de Java, este software utiliza la dirección URL al cual se le realizará las pruebas, el tipo de petición (get), el número de peticiones, tiempo de cada petición, cantidad de usuarios que van a realizar las peticiones, entre otros.

3.13.1. Pruebas de carga. Se utilizaron dos casos para las pruebas tanto de carga como stress, en el primer caso se realizó una petición con 10 usuarios cada 5 segundos sobre un determinado

requerimiento, en el segundo caso se lo realizo con 35 usuarios. Todo esto con la finalidad de comparar los dos casos y ver el comportamiento del sistema.

Tabla 31. Pruebas de carga caso 1

Caso 1			
Prueba	**Tiempo (s)**		**Paquetes Enviados (bytes)**
	Mínimo	**Máximo**	
Login	0,855	0,978	115
Crear Usuarios	1,498	1,949	335
Registro de activos	0,419	0,706	115
Cambio de estado de activos	0,753	2,613	240
Eliminar Activo	1,653	3,075	232
Reporte	0,932	1,224	248

Nota: Tabla de pruebas de carga caso 1.

Tabla 32. Pruebas de carga caso 2

Caso 2			
Prueba	**Tiempo (s)**		**Paquetes Enviados (bytes)**
	Mínimo	**Máximo**	
Login	1,442	4,790	417
Crear Usuarios	1,893	3,572	252
Registro de activos	1,875	5,341	355
Cambio de estado de activos	2,636	6,071	466
Eliminar Activo	1,219	2,187	376
Reporte	1,691	1,894	368

Nota: Tabla de pruebas de carga caso 2.

3.13.2. Análisis de resultados pruebas de carga. En los dos casos mostrados anteriormente se detectó que el tiempo máximo de respuesta de las peticiones es de 6,071 segundos con un total de 466 bytes enviados por segundo, para el proceso de cambio de estado de activos, teniendo en cuenta que la prueba se la hizo para 35 usuarios, además que, este proceso es uno de los más pesados debido a las diferentes validaciones que posee, se puede decir que este tiempo es aceptable. Esto nos muestra que el sistema responde satisfactoriamente a las pruebas realizadas, además no se detectó errores durante la ejecución de las pruebas realizadas al sistema.

3.13.3. Pruebas de stress. Las pruebas de stress se hicieron en un único caso ya que el objetivo es ver el comportamiento del sistema bajo una gran carga de datos. Se realizaron 500 peticiones de usuarios por segundo durante un tiempo indefinido. Las pruebas de stress permiten comprobar el rendimiento del sistema en situaciones en las cuales el sistema pueda colapsar por la sobrecarga de peticiones de usuarios o de información.

Tabla 33. Pruebas de stress

Prueba	**Tiempo (s)**		**Paquetes Enviados (bytes)**
	Mínimo	**Máximo**	
Login	0,398	3,603	279
Crear Usuarios	13,078	16,217	252
Registro de activos	7,607	9,822	248
Cambio de estado de activos	12,193	18,258	355
Eliminar Activo	1,671	2,626	352
Reporte	0,208	3,771	236

Nota: Tabla de pruebas de stress.

3.13.4. Análisis de resultados pruebas de stress. Tomando en cuenta las 500 peticiones concurrentes, se concluyó que el sistema no colapsa y cumple con las expectativas de funcionar correctamente. Las pruebas de stress reflejaron que el tiempo máximo de respuesta en el peor de los escenarios es de 18,258 segundos en el proceso de cambio de estado de activos, tiempo aceptable para que un usuario no perciba mucho tiempo de espera, además, todos los paquetes se enviaron correctamente lo que refleja que el sistema no tiene ningún error al momento de enviar o desplegar su información.

El tiempo que lleva al sistema procesar el login varía entre 0,398 a 3,603 segundos y es menor al tiempo de registrar un activo, eliminar un activo o desplegar el reporte de activos. Todo esto se debe a la cantidad de registros y datos que va a desplegar la pantalla.

3.14. Plan de pruebas.

3.14.1. Creación de nuevo activo de TI.

Tabla 34. Prueba de creación de un activo

Nombre de la prueba:		Creación de Nuevo activo de TI		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Guardar sin ningún valor en la tabla	Se muestra un mensaje indicando: Debe llenar todos los campos requeridos.	1	0
2	Guardar con un valor de serie previamente usado en otro activo	Se muestra un mensaje indicando: Ya se encuentra registrado un activo con la serie x.	1	0
3	Guardar con un valor de TAG previamente usado en otro activo	Se muestra un mensaje indicando: Ya se encuentra registrado un activo con el TAG x.	1	0
4	Guardar el registro con el estado del activo “asignado” y sin responsable	Se muestra un mensaje indicando: Debe seleccionar un responsable.	1	0
5	Guardar el registro con los campos obligatorios y sin adjuntos	Se regresa al listado de activos y muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
6	Guardar el registro con los campos obligatorios 1 documento adjunto	Se regresa al listado de activos y muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1

7	Guardar el registro con los campos obligatorios 4 documento adjunto	Se regresa al listado de activos y muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: Se han probado todas las validaciones posibles al momento de guardar un nuevo activo de TI y la aplicación muestra un mensaje de error cuando el activo no se pueda crear.				
Conclusiones: Es posible crear un nuevo activo sin ningún problema.				

Nota: Tabla de prueba de creación de un activo.

3.14.2. Edición de activo de TI.

Tabla 35. Pruebas de edición de un activo

Nombre de la prueba:		Edición de activo de TI		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Guardar sin ningún valor en la tabla	Se muestra un mensaje indicando: Debe llenar todos los campos requeridos.	1	0
2	Guardar con un valor de serie previamente usado en otro activo	Se muestra un mensaje indicando: Ya se encuentra registrado un activo con la serie x.	1	0
3	Guardar con un valor de TAG previamente usado en otro activo	Se muestra un mensaje indicando: Ya se encuentra registrado un activo con el TAG x.	1	0
4	Guardar el registro con los campos obligatorios y sin adjuntos	Se regresa al listado de activos y muestra un mensaje indicando:	1	1

		Correcto: Procesado satisfactoriamente.		
5	Guardar el registro con los campos obligatorios 4 documento adjunto	Se regresa al listado de activos y muestra un mensaje indicando: Correcto: Procesado satisfactoriamente. El activo ahora cuenta los adjuntos que tenía, más los nuevos adjuntos.	1	1
Observaciones: Se han probado todas las validaciones posibles al momento de guardar un nuevo activo de TI y la aplicación muestra un mensaje de error cuando el activo no se pueda crear. No se muestra habilitados los campos Estado del activo, Estado funcional y Responsable.				
Conclusiones: Es posible editar un activo sin ningún problema.				

Nota: Tabla de pruebas de edición de un activo.

3.14.3. Eliminar activo

Tabla 36. Pruebas de eliminación de un activo

Nombre de la prueba:		Eliminación de activo de TI		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Eliminar un activo de TI creado con anterioridad	Se muestra un mensaje de confirmación de eliminación y posterior un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: No existe ninguna validación para la eliminación del activo.				
Conclusiones: Al eliminar el activo, se elimina de forma permanente de la aplicación.				

Nota: Tabla de pruebas de eliminación de un activo.

3.14.4. Ver archivos adjuntos

Tabla 37. Pruebas para ver archivos adjuntos

Nombre de la prueba:		Descargar archivos adjuntos de activos de TI		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Descargar archivo por archivo.	Al presionar sobre la opción de descargar archivo, se descarga el documento seleccionado.	1	1
2	Descargar todos los archivos del activo	Se descargan todos los documentos del activo comprimidos.	1	1
Observaciones: Ninguna.				
Conclusiones: Se puede descargar los documentos relacionados al activo que se han adjuntado previamente, para luego proceder a visualizarlos.				

Nota: Tabla de pruebas para ver archivos adjuntos.

3.14.5. Cambiar de estados

Tabla 38. Pruebas estado asignado

Nombre de la prueba:		Cambiar a estado “asignado” al activo		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Cambiar a estado “asignado” a un activo en estado “almacenado” sin ingresar un responsable	Se muestra un mensaje indicando: Debe seleccionar un responsable.	1	0
2	Cambiar a estado “asignado” a un activo en estado “almacenado” sin ingresar una ubicación	Se muestra un mensaje indicando: Debe seleccionar una ubicación.	1	0
3	Cambiar a estado “asignado” a un activo en estado “almacenado” seleccionando un	Se muestra un mensaje indicando:	1	1

	responsable y una ubicación	Correcto: Procesado satisfactoriamente.		
Observaciones: La aplicación muestra un mensaje informativo acorde a la situación dada, ya se para indicar q hubo error o hubo éxito.				
Conclusiones: Se puede cambiar de estado del activo a asignado solo si se ingresa el responsable y ubicación. El estado funcional del activo siempre será definido como operativo.				

Nota: Tabla de pruebas estado asignado.

Tabla 39. Pruebas estado préstamo

Nombre de la prueba:		Cambiar a estado "préstamo" al activo		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Cambiar a estado "préstamo" a un activo en estado "almacenado" sin ingresar un responsable	Se muestra un mensaje indicando: Debe seleccionar un responsable.	1	0
2	Cambiar a estado "préstamo" a un activo en estado "almacenado" sin ingresar una ubicación	Se muestra un mensaje indicando: Debe seleccionar una ubicación.	1	0
3	Cambiar a estado "préstamo" a un activo en estado "almacenado" seleccionando un responsable y una ubicación	Se muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: La aplicación muestra un mensaje informativo acorde a la situación dada, ya se para indicar q hubo error o hubo éxito.				
Conclusiones: Se puede cambiar de estado del activo a préstamo solo si se ingresa el responsable y ubicación.				

Nota: Tabla de pruebas estado préstamo.

Tabla 40. Pruebas estado de reparación

Nombre de la prueba:	Cambiar a estado "reparación" un activo
Responsable de la prueba:	Francisco Cotacachi

No	Operación	Resultado	Intentos	Éxito
1	Cambiar a estado “reparación” un activo	Se muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: No es necesario ingresar ningún campo para cambiar a este estado.				
Conclusiones: Al cambiar a estado reparación, el estado funcional siempre será no operativo.				

Nota: Tabla de pruebas estado de reparación.

Tabla 41. Pruebas estado de chatarrizado

Nombre de la prueba:		Cambiar a estado “chatarrizado” al activo		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Cambiar a estado “chatarrizado” a un activo	Se muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: No es necesario ingresar ningún campo para cambiar a este estado.				
Conclusiones: Al cambiar a estado chatarrizado, el estado funcional siempre será no operativo, no tendrá ni responsable ni ubicación.				

Nota: Tabla de pruebas estado de chatarrizado.

Tabla 42. Pruebas estado de extraviado/robado

Nombre de la prueba:		Cambiar a estado “extraviado/robado” al activo		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Cambiar a estado “extraviado/robado” a un activo	Se muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: No es necesario ingresar ningún campo para cambiar a este estado.				

Conclusiones: Al cambiar a estado chatarrizado, el estado funcional siempre será no operativo, no tendrá ni responsable ni ubicación.

Nota: Tabla de pruebas estado de extraviado/robado.

3.14.6. Creación de usuarios

Tabla 43. Pruebas creación de usuarios

Nombre de la prueba:		Creación de usuario		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Guardar con todos los campos en blanco	Se muestra un mensaje indicando: Debe llenar todos los campos requeridos.	1	0
2	Guardar con el correo en blanco	Se muestra un mensaje indicando: El campo Correo es obligatorio. Debe llenar todos los campos requeridos.	1	0
3	Guardar con una cuenta ya existente	Se muestra un mensaje indicando: Ya se encuentra registrada la cuenta x.	1	0
4	Guardar con todos los campos llenos	Se muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: Se han probado todas las validaciones posibles al momento de guardar un nuevo usuario.				
Conclusiones: Es posible crear un nuevo usuario sin ningún problema al llenar todos los campos requeridos.				

Nota: Tabla de pruebas creación de usuarios.

3.14.7. Eliminar usuarios

Tabla 44. Pruebas eliminación de usuarios

Nombre de la prueba:		Eliminación de usuario		
Responsable de la prueba:		Francisco Cotacachi		
No	**Operación**	**Resultado**	**Intentos**	**Éxito**
1	Eliminar un usuario con activos asignados	Se muestra un mensaje indicando: No se puede eliminar un usuario que tenga activos asignados.	1	0
2	Eliminar un usuario con historial de activos	Se muestra un mensaje indicando: No se puede eliminar un usuario que tenga historial de activos asignados.	1	0
3	Eliminar un usuario	Se muestra un mensaje indicando: Correcto: Procesado satisfactoriamente.	1	1
Observaciones: No se pueden eliminar usuarios que tenga o hayan tenido activos a su cargo.				
Conclusiones: Es posible eliminar un usuario solo si no tiene relación al algún activo.				

Nota: Tabla de pruebas eliminación de usuarios.

CONCLUSIONES

- Al término del trabajo se puede concluir que la metodología XP es de gran ayuda, ya que facilita la construcción del software al tener varias interacciones con el cliente y con ello tener un análisis de cada módulo a desarrollar y que en cada entregable no se tenga cambios fuertes que perjudicarían los tiempos de entrega del sistema.
- Conjuntamente, con el desarrollo de la aplicación web para la gestión de activos de TI, el personal involucrado en dicho proceso podrá acceder fácilmente al uso de la aplicación desde cualquier computador con acceso a internet y así llevar a cabo las diferentes tareas de almacenamiento, asignación, préstamo, reparación, chatarrización y extravío de los activos.
- Así, la empresa consolidará los datos relacionados a los activos de TI en una sola base de datos, que anteriormente se encontraban en Excel, y a través de ello, mejorar la organización de dichos datos para su futuro uso evitando perdidas y redundancia de datos de los activos de TI que posee la empresa.
- Del mismo modo, el módulo de reportes permitirá filtrar los activos según los criterios de búsqueda conforme a la necesidad del interesado, y poder así, tener acceso a la información detallada e histórica de dichos activos de TI, requerida en cualquier lugar y momento, con la finalidad de tomar decisiones adecuadas y solventar necesidades emergentes que se susciten dentro de la ejecución de las actividades en la institución.
- El sistema ayudará en el control de los responsables asignados a los activos de TI, mediante notificaciones automáticas vía correo, a los diferentes actores involucrados en el proceso de cambio de estado, tales como: jefe de TI, responsable del manejo de inventario y usuario final. Indicándoles así, su responsabilidad sobre el correcto uso de dicho activo desde la fecha en cual fue asignado hasta el momento de su devolución o nuevo cambio de estado del activo de TI.

RECOMENDACIONES

- Después de construir la aplicación web para la gestión de activos de TI para la empresa Garancheck.Cía.Ltda, se recomienda siempre que se quiera automatizar un proceso manual, realizar un diagrama de proceso para ver qué pasos se puede o no se puede excluir al momento de automatizarlo.
- Para tener un mejor seguimiento de los activos de TI, se propone una futura versión que lleve a cabo la generación de códigos de barras y lectura de estos códigos para el etiquetado de cada activo ingresado en el sistema.
- Además, se aconseja implementar un módulo para lograr una gestión de mantenimientos preventivos de cada activo para aumentar la vida útil de cada activo.
- Por último, para un futuro proyecto es recomendable realizar la versión móvil del sistema ya que en la actualidad está tomando más acogida dichos dispositivos y mejorara la experiencia del usuario.

GLOSARIO DE TÉRMINOS

TI: Tecnología de información.

XP: Programación extrema o Extreme Programming.

SCRUM: Metodología para desarrollo de software.

UML: Lenguaje unificado de modelado.

UWE: Ingeniería Web Basada en UML.

GitKraken: Aplicación para gestionar código fuente.

Fork: Aplicación para gestionar código fuente.

Tag: Código alfanumérico único de cada activo.

HTML: Lenguaje de marcado para crear páginas web.

Css: hojas de estilo para crear presentaciones de páginas web.

RF01: Requerimiento funcional.

RNF01: Requerimiento no funcional.

LISTA DE REFERENCIAS

Alvarez, M. A. (01 de Enero de 2001). *Qué es HTML*. Obtenido de https://desarrolloweb.com/articulos/que-es-html.html

Chisaguano, C. D., & Ludeña, V. M. (Febrero de 2019). *Análisis, diseño e implementación de un sistema de gestión de inventarios de activos de TI para los laboratorios de la carrera de Ingeniería en Ciencias de la Computación de la Universidad Politécnica Salesiana sede Quito campus sur.* Obtenido de dspace: http://dspace.ups.edu.ec/handle/123456789/16969

Freshservice. (20 de Diciembre de 2020). *¿Qué es una herramienta de gestión de activos?* Obtenido de Freshservice: https://freshservice.com/latam/it-asset-management-software/

Garancheck Cía. Ltda. (2020). *Desarrollamos sus proyectos*. Obtenido de Garancheck: http://www.garancheck.com.ec/main/

Gonzales, J. A. (2018). *Sistema automatizado de gestión de activos de TI basado en la norma ISO/IEC 19770-3.* Lima: Universidad Privada Norbert Wiener.

Guajardo, P. (22 de Mayo de 2020). *Bootstrap: guía para principiantes de qué es, por qué y cómo usarlo*. Obtenido de Rock Content: https://rockcontent.com/es/blog/bootstrap/

ISSI. (12 de Nov de 2003). *Metodologías Ágiles en el Desarrollo de Software.* Obtenido de Ingeniería del Software y Sistemas de Información): http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf

jQuery. (2020). *¿Qué es jQuery?* Obtenido de La Fundación jQuery : https://jquery.com/

La Rosa, A. (14 de Junio de 2018). *Gestión de activos de TI: Desafíos de la implementación*. Obtenido de pandorafms: https://pandorafms.com/blog/es/gestion-de-activos-de-ti/

Maldonado, D. (05 de Enero de 2019). *¿Qué es la Gestión de Activos de TI?* Obtenido de Icorp : http://www.icorp.com.mx/blog/que-es-la-gestion-de-activos-de-ti/

Microsoft. (Noviembre de 2020). *Pruebe SQL Server on-premises o en el cloud*. Obtenido de Microsoft: https://www.microsoft.com/es-es/sql-server/sql-server-downloads

Microsoft. (Noviembre de 2020). *Trabaja de forma visual. Diagramación de forma sencilla.* Obtenido de https://www.microsoft.com/es-ww/microsoft-365/visio/flowchart-software

Microsoft. (Noviembre de 2020). *Visual Studio |Las mejores herramientas para cualquier desarrollador*. Obtenido de https://visualstudio.microsoft.com/es/

Oracle. (Noviembre de 2020). *¿Qué es una base de datos relacional?* Obtenido de Oracle Argentina: https://www.oracle.com/ar/database/what-is-a-relational-database/

Ortego, D. D. (29 de Marzo de 2017). *¿Qué es C#?* Obtenido de Openwebinars : https://openwebinars.net/blog/que-es-c-introduccion/

Pellicer, P. (2017). *¿Qué es el .NET? ¿Para qué sirve?* Obtenido de emagister: https://www.emagister.com/blog/que-es-el-net-para-que-sirve/

Schwabe, D., Rossi, G. H., Olsina, L., & Pastor, Ó. (2008). *Web Engineering: Modelling and Implementing Web Applications.* Londres: Springer.

Tecon. (Nov de 2020). *¿Qué es Azure?* Obtenido de Soluciones Informáticas Tecon: https://www.tecon.es/que-es-microsoft-azure-como-funciona/